



Ubiquitous Robotics System for Knowledge-based Auto-configuration System for Service Delivery within Smart Home Environments

PhD Thesis

Mustafa Awwad Al-Khawaldeh

Mechatronics Research Group
Department of Engineering - Faculty of Technology
Leicester - United Kingdom

This thesis is submitted in partial fulfilment of the requirements of De Montfort
University for the award of Doctor of Philosophy (PhD)

April 2014

Declaration

No part of the material described in this thesis has been submitted for the award of any other degree or qualification in this or any other university or college of advanced education.

Mustafa Al-Khawaldeh

Dedication

To my father, Mr Awwad Al-Khawaldeh and my mother, Mrs Aawayed Al-Khawaldeh for their endless love, encouragement, inspiration, support and prayers, which were candles in my life in spite of the distance. They have inculcated in me a love of pursuing higher studies.

To the soul of my sister Mariam, who dreamt of seeing me being awarded my doctorate. Her faithfulness and prayers were a ray of hope for my success. You brought me a smile. You are always in my thoughts and prayers.

To my lovely wife Nisreen and my little princesses, Merah, Hala and Noor, for their endless support, sacrifice and for always being by my side. Their prayers always act as a catalyst in my academic life. I am especially grateful to Nisreen for being a dedicated wife and mother. She was a beacon of hope in troubled times. I wish you all a prosperous future.

To my brothers and sisters, Mohammad, Ahmad, Mahmood, Issa, Houriah, Fada, Anam, for their continuous love and support throughout my studies.

To all of these dear ones, I dedicate this work, which I hope will attract your attention.

Acknowledgements

First and foremost, I am grateful to Almighty Allah for enabling me to complete this research work. Without him, nothing is possible.

This thesis marks the end of my PhD journey, for which I feel indebted to many supportive people. I would like to express my sincere gratitude for my supervisory team: Dr. Xi Chen and Prof. Philip Moore, at De Montfort University, United Kingdom, for being exceptionally helpful. It has been an honour to be their PhD student. They motivated me and encouraged me through tough times by simplifying issues and providing me with much-needed support, thus directing me onto the right path. The general atmosphere of working with the team was comfortable. In addition, the research team was never hesitant in providing me with clear and constructive feedback on my progress as my research was part of a collaborative group project. Their sincere interest in my research development was just overwhelming and kept me animated and energised throughout my studies. The invaluable experience which they shared with me, preventing me from taking wrong steps and saving me a lot of time, will be cherished forever. Furthermore, they always willingly and promptly tolerated and accepted my mistakes, considering them a natural component of my learning experience, while showering me with praise for my good performance.

I acknowledge and extend my genuine gratitude to Dr. Chi Biu Wong, to whom I am indebted for his support, encouragement and constructive criticism, and Dr. Haibo Jia for his supportive ideas. I am really thankful for the knowledge and experience he

shared with me and his friendly support in negotiating the various hurdles I have encountered. I also thank Dr. Seng Chong for sharing his academic experience in the Mechatronics Research Group.

I gratefully acknowledge the funding source that facilitated my PhD work. I would like to express my deep gratitude to Philadelphia University for providing the scholarship to support my research study at De Montfort University. In addition, special thanks are made to Prof. Marwan Kamal, the former President of the Philadelphia University and to the current President Prof. Mohammad Awwad, to Prof. Qasem Obaidi, to Dr. Ibrahim Badran and to all my colleagues at Mechatronics Engineering at Philadelphia University for their advice, help and cooperation. Special thanks go to Prof. Omar Badran for his encouragement.

I also want to extend my thanks to my colleagues and friends at De Montfort University and University of Leicester specially my office mate Engineer Ghazi Al-Qaruiti, for their academic discussions. I owe Mr Shadan Khattak and Mr. Abhishek Gupta a debt of gratitude for their encouragement and academically useful discussions. I specially appreciate the help of my friends, Ayedh Al-Ajmi and Abdullah Al-Ajimi; I will never forget the good times we spent at De Montfort University. They have been a source of friendship as well as good advice and collaboration. I very much appreciate their enthusiasm and willingness to support me through tough times.

I am particularly indebted to my brother Dr Ahamd Al-Khawaldeh for his steadfast support and unstinting help, which was greatly needed and deeply appreciated. I do not have enough words to say how thankful I am to you, Abu Bashar.

I must also thank my family for all their love and encouragement, particularly my parents, who have inculcated a love of science in me and supported me in all my academic pursuits. I am deeply indebted to my wife, without whose patience, constant support, love and continued enthusiasm throughout the project, this work would not have been achieved. The joy and enthusiasm she has for her PhD research was motivational for me, even during tough times in my journey. I am also grateful for the excellent example she has provided as a successful wife, mother and student. Finally, special thanks are due to my parents-in-law, who have been a constant source of boundless love, support and encouragement. I know I can never repay their favour, but I would like to seize the opportunity to thank them all for what they have given me.

Thank you all

Mustafa Al-Khawaldeh

Synopsis

The future smart home will be enhanced and driven by the recent advance of the Internet of Things (IoT), which advocates the integration of computational devices within an Internet architecture on a global scale [1, 2]. In the IoT paradigm, the smart home will be developed by interconnecting a plethora of smart objects both inside and outside the home environment [3-5]. The recent take-up of these connected devices within home environments is slowly and surely transforming traditional home living environments. Such connected and integrated home environments lead to the concept of the smart home, which has attracted significant research efforts to enhance the functionality of home environments with a wide range of novel services.

The wide availability of services and devices within contemporary smart home environments make their management a challenging and rewarding task. The trend whereby the development of smart home services is decoupled from that of smart home devices increases the complexity of this task. As such, it is desirable that smart home services are developed and deployed independently, rather than pre-bundled with specific devices, although it must be recognised that this is not always practical. Moreover, systems need to facilitate the deployment process and cope with any changes in the target environment after deployment.

Maintaining complex smart home systems throughout their lifecycle entails considerable resources and effort. These challenges have stimulated the need for dynamic auto-configurable services amongst such distributed systems. Although

significant research has been directed towards achieving auto-configuration, none of the existing solutions is sufficient to achieve auto-configuration within smart home environments. All such solutions are considered incomplete, as they lack the ability to meet all smart home requirements efficiently. These requirements include the ability to adapt flexibly to new and dynamic home environments without direct user intervention. Fulfilling these requirements would enhance the performance of smart home systems and help to address cost-effectiveness, considering the financial implications of the manual configuration of smart home environments.

Current configuration approaches fail to meet one or more of the requirements of smart homes. If one of these approaches meets the flexibility criterion, the configuration is either not executed online without affecting the system or requires direct user intervention. In other words, there is no adequate solution to allow smart home systems to adapt dynamically to changing circumstances, hence to enable the correct interconnections among its components without direct user intervention and the interruption of the whole system. Therefore, it is necessary to develop an efficient, adaptive, agile and flexible system that adapts dynamically to each new requirement of the smart home environment.

This research aims to devise methods to automate the activities associated with customised service delivery for dynamic home environments by exploiting recent advances in the field of ubiquitous robotics and Semantic Web technologies. It introduces a novel approach called the *Knowledge-based Auto-configuration Software*

Robot (Sobot) for Smart Home Environments, which utilises the Sobot to achieve auto-configuration of the system.

The research work was conducted under the Distributed Integrated Care Services and Systems (iCARE) project, which was designed to accomplish and deliver integrated distributed ecosystems with a homecare focus. The auto-configuration Sobot which is the focus of this thesis is a key component of the iCARE project. It will become one of the key enabling technologies for generic smart home environments. It has a profound impact on designing and implementing a high quality system. Its main role is to generate a feasible configuration that meets the given requirements using the knowledgebase of the smart home environment as a core component.

The knowledgebase plays a pivotal role in helping the Sobot to automatically select the most appropriate resources in a given context-aware system via semantic searching and matching. Ontology as a technique of knowledgebase representation generally helps to design and develop a specific domain. It is also a key technology for the Semantic Web, which enables a common understanding amongst software agents and people, clarifies the domain assumptions and facilitates the reuse and analysis of its knowledge.

The main advantages of the Sobot over traditional applications is its awareness of the changing digital and physical environments and its ability to interpret these changes, extract the relevant contextual data and merge any new information or knowledge. The Sobot is capable of creating new or alternative feasible configurations to meet the system's goal by utilising inferred facts based on the smart home ontological model, so that the system can adapt to the changed environment. Furthermore, the Sobot has the

capability to execute the generated reconfiguration plan without interrupting the running of the system.

A proof-of-concept testbed has been designed and implemented. The case studies carried out have shown the potential of the proposed approach to achieve flexible and reliable auto-configuration of the smart home system, with promising directions for future research.

Table of Contents

Declaration.....	i
Dedication	ii
Acknowledgements.....	iii
Synopsis.....	vi
List of Tables	xv
List of Figures.....	xvi
List of Abbreviations and Acronyms	xviii
Chapter 1: Introduction	1
1.1 Research Motivation.....	1
1.2 Research Aim and Objectives	5
1.2.1 Aim	5
1.2.2 Objectives	5
1.3 Research Scope.....	6
1.4 Thesis Outline.....	7
Chapter 2: Smart Home Environments	11
2.1 Introduction	11
2.2 The Concept of the Smart Home Environment	12
2.3 Applications in Smart Home Environments.....	14
2.4 The Necessity of Auto-configuration of Smart Home Environments	18
2.5 Summary	22
Chapter 3: Auto-configuration within Smart Home Environments	23
3.1 Introduction	23
3.2 Types of Auto-configuration	24
3.3 Configuration Systems Solutions	26
3.3.1 Configuration Systems Based on Knowledge Representation Technology.....	28
3.3.2 Configuration Systems Based on Autonomous Robotics Technology.....	30
3.3.3 Configuration System Solutions Comparison.....	33
3.4 Summary	39

Chapter 4: Enabling Technologies for Auto-configuration within Smart Home Environments	41
4.1 Introduction	41
4.2 Evolution of Robotics Technology.....	42
4.3 Software Robots (Sobots).....	46
4.4 Robot Ethics	52
4.5 Knowledge Representation Technologies	54
4.5.1 Definition of Knowledgebase	54
4.5.2 Ontologies	57
4.5.2.1 The Ontology Concept	57
4.5.2.2 Ontology Features	58
4.5.2.3 Ontology Languages	61
4.5.2.4 Ontology Models for Smart Homes	67
4.6 Summary	70
Chapter 5: Knowledge-based Auto-configuration Sobot Architecture for Smart Home Environments	72
5.1 Introduction	72
5.2 Overview of the iCARE Project.....	74
5.2.1 iCARE Actors and Context	75
5.2.2 iCARE Reference Architecture	78
5.2.2.1 Supporting Layer.....	78
5.2.2.2 Device Interface Layer.....	80
5.2.2.3 Harmonisation Layer.....	81
5.2.2.4 Service Layer	82
5.2.2.5 Service Execution Layer	83
5.3 Service Delivery Schema within the iCARE Architecture	84
5.4 Requirements and Specifications of the Proposed Approach.....	87
5.5 The Belief-Desire-Intention (BDI) Model	90
5.6 Configuration Sobot Conceptual Solution	94
5.6.1 The Resource Handler Component.....	95
5.6.2 The Local Knowledgebase.....	96

5.6.2.1 Context Model.....	96
5.6.2.2 Device Model	97
5.6.2.3 Service Model	98
5.6.2.4 Configuration Model.....	98
5.6.3 Configuration Plan Generation Engine Component	99
5.6.4 Configuration Plan Executor	100
5.7 Summary	100
Chapter 6: Knowledgebase Model for Auto-configuration Sobot within a Smart Home Environment.....	102
6.1 Introduction	102
6.2 Ontology-based Knowledge Representation	103
6.3 Ontology Modelling Method.....	106
6.4 Smart Home Ontology Design	110
6.4.1 Domain, Purpose, Source and Scope	111
6.4.2 Existing Domain Ontologies.....	111
6.4.2.1 Context Ontology Model	114
6.4.2.2 Device Ontology Model.....	116
6.4.2.3 Service Ontology Model	117
6.4.2.4 Configuration Ontology Model.....	117
6.4.3 Brainstorming Important Terms	117
6.4.4 The Modelling Approach.....	119
6.4.5 Identification of Concepts and their Properties	119
6.5 Building a Knowledgebase Model Using Protégé.....	122
6.5.1 Development Languages.....	122
6.5.1.1 Web Ontology Language (OWL).....	122
6.5.1.2 Semantic Web Rule Language (SWRL)	123
6.5.2 Building the Smart Home Ontology	124
6.5.2.1 Context Model.....	124
6.5.2.2 Device Model	127
6.5.2.3 Service Model	132
6.5.3 Configuration Rule Design	135

6.6 Summary	141
Chapter 7: Sobot Implementation	143
7.1 Introduction	143
7.2 Sobot Development Environment	144
7.2.1 Ontology Development Environment.....	144
7.2.2 Reasoning Engine	145
7.3 The OSGi Framework	146
7.4 Implementation of the Sobot Prototype in the OSGi Framework	150
7.4.1 The Resource Handler	151
7.4.2 The Local Knowledgebase Model	153
7.4.3 Configuration Plan Generation Engine.....	154
7.4.4 Configuration Plan Executor	156
7.5 Workflow for Sobot Auto-configuration.....	156
7.6 Summary	159
Chapter 8: Validation of the Sobot Prototype.....	160
8.1 Introduction	160
8.2 The Testbed	161
8.3 The Scenarios for Case Studies	164
8.3.1 Early Start and Stop Scenarios.....	164
8.3.2 Non-occupancy Setback Scenario	166
8.4 Resource Instantiation	166
8.5 The Usefulness and Effectiveness of the Inference Mechanism	174
8.6 Results, Performance Metrics and Evaluation Criteria	182
8.7 Summary	188
Chapter 9: Conclusion and Recommendations	190
9.1 Research Summary	190
9.2 Research Findings	192
9.3 Contributions to Knowledge	194
9.4 Recommendations for Future Work	196
References	198
Appendices.....	222

A.1: A Ontology Result Classified Using Pellet	222
A.2 : OWL/XML Rendering of the Whole Ontology Including the Individuals	Error! Bookmark not defined.
A.3: Ontology Metrics.....	Error! Bookmark not defined.

List of Tables

Table 3.1: A Comparison of Configuration-related Research	36
Table 6.1 : Identified Concepts of Temperature Regulation Service Object	121
Table 6.2 : OWL Ontology Reasoning Rules [102].....	139
Table 6.3 : SWRL Rule Atoms [101].....	140
Table 8.1: Sensors and Actuators in the Testbed	163
Table 8.2: Individuals by Type	170
Table 8.3: Symbolic Location	173

List of Figures

Figure 4.1: Evolution of Robotics	44
Figure 4.2: Ubibot Components	47
Figure 4.3: Semantic Web Language Architecture [195]	62
Figure 4.4: An is-a hierarchy(taxonomy) [195]	64
Figure 5.1: Integrated iCARE Reference Architecture	75
Figure 5.2: iCARE Configuration Process Architecture	86
Figure 5.3: The BDI-based Structure of Sobot [230].....	93
Figure 5.4: Sobot Architecture	95
Figure 5.5: Knowledge Package Model	97
Figure 6.1: Knowledgebase of the Smart Home Domain [30].....	105
Figure 6.2: Ontology Development Process [241].....	107
Figure 6. 3: Knowledge Package Model	113
Figure 6.4: Knowledge Layer Model	114
Figure 6.5: Temperature Regulation Service Object.....	120
Figure 6. 6: Location Model.....	125
Figure 6. 7: Subclasses of IndoorArea	125
Figure 6.8: Subclasses of OutdoorArea	126
Figure 6.9: Subclasses of the RoomArea	126
Figure 6.10: Physical Device Model	127
Figure 6.11: Device Model	128
Figure 6. 12: Subclasses of Parameter	129

Figure 6.13: Subclasses of Measurand, EnergyMeasurand and TemperstureMeasurand	130
Figure 6.14: Subclasses of Event	130
Figure 6.15: Subclasses of DataPort	131
Figure 6.16: Subclasses of Action and ActionStatus	132
Figure 6.17: Service Model.....	133
Figure 6.18:Subclasses of Alarm	133
Figure 6.19: Main Classes of the iCARE Ontology Model	134
Figure 6.20: Object Property Hierarchy and Data Property Hierarchy.....	135
Figure 7.1: OSGi Architecture [268]	150
Figure 7.2: Sobot Architecture within the OSGi Framework	152
Figure 7.3: Workflow for Sobot Auto-configuration.....	158
Figure 8.1: Testbed Floor Layout.....	162
Figure 8.2: Early Start Strategy.....	165
Figure 8.3: Early Stop Strategy	166
Figure 8.4: Transforming Diagram	168
Figure 8.5: The Functional Block Presentations of the Services and Devices Involved	169
Figure 8.6: Syntax Symbol-based Linkage	175
Figure 8.7: Ontology-based Linkage.....	176
Figure 8. 8: The Result of Executing the Parameter Propagation Rules.....	181
Figure 8. 9: The Result of the Sobot Configuration Process – GardenTemperatureSensor is connected to HeatingControlService.....	185

List of Abbreviations and Acronyms

AI	Artificial Intelligence
AmI	Ambient Intelligence
AODA	Autonomic Ontology Driven Architecture
ASyMTRe	Automated Synthesis of Multi-Robot Task Solutions through Software Reconfiguration
ASyMTRe-D	Automated Synthesis of Multi-Robot Task Solutions through Software Reconfiguration-Distribution
BDI	Belief-Desires-Intentions
CoBrA	Context-Broker Architecture
COBRA-ONT	Context Broker Architecture ONTology
CONON	CONtext Ontology
CPS	Cyber-Physical System
DAML+OIL	DARPA Agent Markup Language+ Ontology Inference Layer
DogOnt	Ontology Modelling for Intelligent Domotic Environments

ECA	Event-Condition-Action
EHS	European Home System
Embot	Embedded Robot
Env	Environment
FIPA	Foundation for Intelligent Physical Agents
IBM	International Business Machines
iCARE	Distributed Integrated Care Services and Systems
IFR	International Federation of Robotics
IoE	Internet of Everything
IoT	Internet of Things
JESS	Java Expert System Shell
JVM	Java Virtual Machine
Mobot	Mobile Robot
NS	Name Space
O-A-V	Object-Attribute-Value
OIL	Ontology Inference Layer
OSGi	Open Services Gateway initiative

OWL	Web Ontology Language
OWL DL	Web Ontology Language Description Logic
OWL-S	Ontology Web Language based framework of the Semantic Web
PEIS	Physically Embedded Intelligent System
RDF	Resource Description Framework
RDF(S).	Resource Description Framework Schema
RFID	Radio-Frequency Identification
SGML	Standard Generalized Markup Language
SHOM	Smart Home Ontology Model
SIS	Symbiotic Information System
SOA	Service Oriented Architecture
Sobot	Software Robot
SOCAM	Service-Oriented Context-Aware Middleware
SOUPA	Standard Ontology for Ubiquitous and Pervasive Applications
SW	Semantic Web
SWRL	Semantic Web Rule Language
Ubibots	Ubiquitous Robotics

UFAM	Ubiquitous Functions Activation Module
UML	Unified Modelling Language
URI	Uniform Resource Identifier
U-Space	Ubiquitous Space
W3C	World Wide Web Consortium W3C
WSDL	Web Services Description Language
XML	eXtensible Markup Language
Xmlscheme	eXtensible Markup Language Scheme

Chapter 1: Introduction

1.1 Research Motivation

The smart home environment (also known as the automated home or intelligent home) is defined as a digital environment which is made sensitive, responsive and adaptive to the presence of people by coordinating various resources to adjust the functions or behaviours of the home according to household needs [6-11]. The realisation of the smart home promises users a lived environment with convenience, security, comfort and reduced energy demand [9, 12, 13].

Although the goals of the smart home have changed little, its implementation and contents have evolved, following technological advances in recent decades. It started with the implementation of standalone home automation systems and now comprise

ecosystems [6] where small computing devices are embedded inside physical commodities (smart objects) and personal services on these ubiquitous devices [14].

The future smart home will be enhanced and driven by the recent advance of the Internet of Things (IoT), which advocates the integration of computational devices within Internet architecture on a global scale [1, 2]. In the IoT paradigm, the smart home will be developed by interconnecting a plethora of smart objects both inside and outside the home environment [3, 5]. This type of network of smart objects is designed to sense and manipulate the physical environment on an unprecedented scale [15, 16]. Smart home applications such as energy management, environmental monitoring, security and health care can benefit from these smart objects [17-20]. For example, smart home applications (e.g. heating control, patient monitoring) can obtain accurate local weather observations and forecast information from the Met Office, using data from numerous weather stations.

However, existing smart home systems are typically created by a tight coupling of application functionality and hardware. The functionality of applications (or services) is fully dependent on hardware platforms and application domains [21]. The IoT paradigm will lead to a separation of application functionality and devices (smart objects) to create the future smart home environment [1]. It will break the application domain barrier, so that applications can access the available smart objects freely, regardless of their application domain [22]. Thereby, the direct dependence between hardware platform and application functionality will disappear.

Service-Oriented Architecture (SOA) is a feasible approach to achieving such large-scale decoupling which supports the rapid development of evolvable, interoperable and easily assembled applications. Services are identified, in the context of this thesis, as software entities provided to end users to deliver necessary functions by utilising available smart objects [23-25]. These characteristics are especially important for smart home systems. The service development effort can be significantly reduced as developers concentrate on the implementation of application logic, rather than being distracted by underlying hardware platforms and available smart objects. This will allow service development to be simplified and make low-cost smart home implementation possible for a broad market.

However, each home environment is different in terms of available resources and individual needs. Additionally, available devices keep changing, because devices can join it or leave the home environment randomly. The connection between services and devices may need to be adjusted according to users' preferences and the devices available at a given time. Given their variety, it would be impossible to manage services and devices separately and efficiently, because each one may have its own transport protocol, language, specifications and management data [22, 26].

Currently, services are deployed either via a predefined equipment bundle or by an expert, according to customised preferences. Maintaining such complex systems is very costly and requires great effort, time and expertise, thus failing to meet the essential requirements for a smart home, which are ease of use and affordable cost. Consequently, the full potential of the smart home environment has not yet been achieved [6, 9, 11, 26, 27].

This raises the desirability of developing dynamic, autonomous and configurable infrastructure (or middleware) to compose the services and devices for such distributed systems, to meet users' requirements, satisfy their needs and deal elegantly with the complexity of the smart home environment [21, 28, 29]. Thus, auto-configuration becomes necessary in any smart home to deal with the complexities that emerge from the interactions among services and devices, which are beyond the human capacity to manage while ensuring the essential standards of flexibility and robustness.

In practice, auto-configuration represents a major challenge for the developers of smart home environments, mainly because the installation and configuration of large, dynamic and complex systems and their integration into the existing home environment are time-consuming, intricate and error-prone processes, even for experts [14, 30]. Despite the attention that configuration technologies have received from industry and academia for some time, their development is still at an early stage in the field of smart home service application [30-33].

The smart home environment infrastructure should be flexible enough to support various sequences of activities leading to a customized service delivery. A smart home auto-configuration system will configure infrastructure by following intelligent techniques such as ontology based on a set of semantic descriptions and rules for the applications. For instance, when a new resource is introduced to the environment, the whole system will automatically adapt to its presence by connecting it to appropriate services. Thus, the auto-configuration process can simplify the intricacy and reduce the cost of managing and deploying these services and devices in smart spaces.

A review of the recently published literature indicates that this problem has not been tackled systematically. This implies that there is a knowledge gap in the development of a smart home model which provides an auto-configuration technique that is agile, adaptive and sufficient and requires no direct user intervention. The solutions that have been proposed so far (see Section 3.3) are confined to laboratory simulations and demonstration homes. They are inadequate in that they lack the ability to meet the requirements of the smart home environments, such as ease of networking, high accuracy and cost effectiveness. Therefore, it is significant to develop an efficient, adaptive, agile and flexible system that adapts dynamically to the requirements of the smart home environments. In other words, a system is required to automate the activities of customised service delivery for dynamic home environments without direct user intervention [7, 30, 34].

1.2 Research Aim and Objectives

1.2.1 Aim

The main aim of this research is to devise methods to automate the activities associated with customised service delivery for smart home environments by exploiting the recent advancements in the field of ubiquitous robotics and Semantic Web technologies.

1.2.2 Objectives

The main objectives of this research are:

1. To formulate the specifications and requirements of the configuration process, taking into consideration the needs of users, the demands of services and environmental conditions.

2. To formulate a conceptual framework for the proposed ubiquitous robotics auto-configuration system.
3. To identify service delivery or integration activities that might benefit from automation.
4. To design and implement an ontological knowledge-based auto-configuration proof-of-concept testbed, mainly deriving ubiquitous robotics, to realise the automation of the service delivery or integration activities related to identified cases and scenarios.
5. To assess and validate the efficiency and effectiveness of the proposed approach by implementing a laboratory testbed.

1.3 Research Scope

This research study investigates the use of the latest generation of robotics and Semantic Web technology in the context of smart home environments. Ubiquitous robotics, which merges ubiquitous computing technology with robotic technology, offers potential solutions for smart home auto-configuration. It attempts to automate the integration of services within these environments by employing artificially intelligent software robot (Sobot), enabling the services to auto-configure by adopting ontology-based integration. The Sobot is used to simplify the management and deployment of services and devices in the smart home environment with the help of the auto-configuration concept.

The research work is conducted under the Distributed Integrated Care Services and Systems (iCARE) project which is designed to accomplish and deliver an integrated

distributed home care system. The auto-configuration Sobot is a key part of the iCARE project, helping to achieve auto-configuration of various activities within that project; more details of the iCARE system architecture are provided in Chapter 5.

The Service development, environmental variables and the communication protocols, both wired and wireless are beyond the scope of this work. Similarly, the devices, whether simple or complex, are not the focus of this research; they are simply used for receiving commands and extracting data.

1.4 Thesis Outline

The thesis is structured as follows:

Chapter 2 reviews the state-of-the-art of smart home environments. The smart home environment is a great evolution of the conventional home environment to fulfil its main role of supporting various human activities by offering flexibility and adaptation to deal with the various needs of its users, different contexts and diverse activities. The applications of the smart home environments (e.g. energy management, healthcare, etc.) which build upon and benefits from advances in ambient intelligence (AmI), ubiquitous computing technologies have been discussed. The necessity of auto-configuration of smart home environments is highlighted.

Chapter 3 reviews the state-of-the-art of auto-configuration systems for service delivery within smart environments. It summarises and categorises the approaches adopted by previous researchers in the field of auto-configuration systems for smart environments and robotics technology. The existing approaches have been analysed

against the proposed criteria and discusses their advantages and disadvantages. The chapter highlights the importance of formulating a new approach integrating new technologies namely the ubiquitous robotics and knowledge representation to enhance the smart home automation systems.

Chapter 4 discusses the enabling technologies (ubiquitous robotics and knowledge representation) adopted in the present research to achieve automatic configuration within the smart home systems. It critically reviews the evolution of the three generations of robotics technology, namely industrial robotics, service robotics and ubiquitous robotics. Ubiquitous robotics is particularly discussed in terms of ubiquitous computing, ubiquitous environment. It extensively reviews the features of the Ubibot particularly the Sobot (the focus of this research). The robot ethics is discussed as there is a risk of robots' interference into the people's life in an uncontrolled way.

The chapter also presents knowledge representation (KR) as an area of artificial intelligence (AI) which aims to represent knowledge in symbols to facilitate inferring new elements of knowledge from the existing knowledge. Knowledgebase is defined and its technologies are represented. Ontology which plays a central role in the Semantic Web is also defined. Its importance to the present research is further illustrated as it provides formal models of domain knowledge which can be used by intelligent agents and in this thesis (Sobot). This review leads to the formulation of a new approach whereby the integration of new technologies promises improvements in auto-configuration.

Chapter 5 presents the overall architecture adopted to address the major research challenge. In particular, it offers a detailed description of the conceptual framework of the proposed methodology, which uses new techniques for automating home service delivery and providing coordination among services from different providers with differing objectives and goals. The new approach formulated here benefits from advances in ubiquitous robotics and knowledge representation technologies.

Ubiquitous robotics, which merges ubiquitous computing technology with robotic technology, offers potential solutions for smart home auto-configuration. It automatically delivers services in a cooperative fashion by achieving the rapid configuration of smart home systems without direct user intervention, by means of an ontology-based auto-configuration software robot (Sobot). The BDI is also presented as the model adopted to inspire the formulation of the auto-configuration framework and for the implementation of the Sobot's reasoning mechanism.

Chapter 6 illustrates the design and building of the knowledgebase model using Protégé software. Protégé successfully served as ontology design environment [35-41]. It is an ontology editor as well as a knowledgebase framework. The ontology design, including identification of the main concepts underlying the project is explained. The classifications of general home devices provided by different vendors are discussed. The proposed ontology encompasses knowledge of these devices and their related properties, contexts and relationships, in order to address their current limited or non-existent interoperability.

The Sobot infers logical consequences from a set of facts asserted about the smart home environment, utilizing the ontological knowledge-based model, and then takes the most appropriate auto-configuration decision in light of the given requirements, thus facilitating the achievement of the smart home objectives.

Chapter 7 gives an account of the implementation of the Sobot, whose conceptual design, explained in the Chapter 5, was realised by adopting the OSGi framework to achieve auto-configuration of a smart home system. The development environment of the proposed approach having been established, the proposed auto-configuration Sobot, including a specially designed ontology model, was designed using Protégé (Version 4.1) software. The workflow for Sobot auto-configuration is also provided.

Chapter 8 presents an evaluation of the Sobot prototype within a smart home environment, by reporting experimental case studies which were carried out to evaluate its viability and capabilities. The test cases involved using the Sobot to generate configurations to allow various services exploring available devices to deliver the maximum functions claimed by services. The prototype is assessed in terms of a number of attributes: the flexibility, awareness and adaptability of the Sobot and the usefulness and effectiveness of the ontology and the inference mechanism.

Chapter 9 summarises of the research findings, highlights the research contribution and makes recommendations for future work.

Chapter 2: Smart Home Environments

2.1 Introduction

The smart home is developing rapidly to support various human activities and fulfil their needs by offering a flexible and adaptable environment. A clear definition of the concept of the smart home environment is provided in Section 2.2. The chapter critically reviews the state-of-the-art of the smart home environments. The applications in the smart home environments (e.g. energy management, healthcare, etc.) which build upon and benefit from advances in ambient intelligence (AmI) and ubiquitous computing technologies are discussed in Section 2.3. The development and operation of these applications has led to the necessity of auto-configuration of smart home environments, which is highlighted in Section 2.4. The chapter ends with a summary.

2.2 The Concept of the Smart Home Environment

The smart home environment is an evolving concept driven by advances in technologies such as the Internet of Things, pervasive computing, data mining and the increased availability of robust sensors and actuators [15, 42-44]. The conventional home environment is often unable to fulfil its main role of supporting various human activities because of its rigid and static nature: it offers very limited flexibility and adaptation to deal with the various needs and activities of its users. To overcome these limitations, the concept of the smart home environment has been proposed to improve quality of life by enhancing users' daily activities, supporting their various needs automatically, by altering its behaviour without deliberate human intervention as well as facilitating independent living for their special needs (e.g. elderly, disabled, and patient) [26, 45, 46].

The smart home environment is also known as the automated home or intelligent home. It is a digital, adaptive environment, sensitive and responsive to its residents' needs [6, 11]. It is explicitly defined by Aldrich [47] as "a residence equipped with computing and information technology which anticipates and responds to the needs of the occupants, working to promote their comfort, convenience, security and entertainment through the management of technology within the home and connections to the world beyond". Smart home is described as conjuring up the idea of an imminent future in which people are surrounded by very many fine-grained distributed networks [48]. These networks consist of sensors, actuators and various computational electronic

devices that are inconspicuously used to enhance daily objects such as clothes and furniture, thus collectively constitute digital environments that are adaptive, sensitive and responsive to the inhabitants' presence [25].

Thus, the smart home environment should be able to collect and analyse information about residents' daily lives and use it to enhance their living environment. This entails that it flexibly and securely controls home appliances, monitors residents' health status, advises them on actions (in case of any irregularities occurring) in the home environment, and enables users to easily access the required information [11]. Recent advances in sensor technology, embedded systems and robotics indicate that the realisation of the smart home concept is now practicable, particularly if essential security and privacy concerns are addressed [49].

One of the crucial challenges faced by smart homes is to provide intelligent support reliably within the home environment. Intelligence is an important feature of the smart home. It refers to self-awareness, problem-solving and the ability to memorize, learn, understand, plan and reason about the environment [11]. The smart home is required to be imbued with an awareness of its physical context (e.g. temperature, lighting, house layout), its occupants' context (e.g. preferences, location, activities), as well as temporal context (hour of day, day, week, season, year).

Such an environment imbued with context-aware reasoning makes it conceivable for users to obtain customized services such as temperature and lighting settings based on their preferences and the monitoring of their health status [50]. In the smart home environment, computer software acts as an intelligent agent capable of perceiving the

states of the physical context and its users by means of sensors, reasoning about these states using AI techniques, making decisions and taking actions to accomplish certain goals, such as maximizing users' safety, maintaining their comfort and minimizing consumption of resources [9].

2.3 Applications in Smart Home Environments

Research efforts to make the environment 'smart' have been actively carried out across various domains under the umbrella of AmI[51, 52]. A smart home is an example of an environment enriched with AmI [53]. AmI builds on the idea of pervasive computing to deliver services which are imperceptibly integrated into people's lives through connected devices that are progressively embedded into the environment [49, 54, 55]. Typical service applications in the home environment are energy management and health monitoring [56].

Energy management is no longer a luxury in the domestic environment, but essential for normal householders, who are beginning to seek easy ways to reduce their energy consumption because of high energy prices and environmental concerns [57]. Smart homes equipped with smart energy management automate energy conservation activities to reduce consumption without compromising users' quality of life, by easing their operational burden [18]. Energy management systems within smart home environments have attracted much research interest as an application of smart grid technologies, to extend their capabilities (e.g. automation) into the home [57-59]. The smart home environment plays a vital role in the interaction between the grid and the consumer [60]. Automation technology is a key feature of the smart grid which enables

utility companies to adjust and control each individual device from a central location [61-63].

The smart home also facilitates the assessment of its users' health [17, 64, 65]. For instance, researchers have tried to create a linkage between alterations in users' motion patterns and the commencement of dementia symptoms, benefiting from motion sensors and the evaluation of specific parameters such as walking speed and distance covered [66]. Smart home technologies have also been used for early-childhood autism screening [67] and performing accustomed routines for initiating new health behaviours [65, 68-71].

The smart home provides essential infrastructure to enable healthcare services to deliver care to people in their own homes. Lifestyle monitoring, which is a typical telecare application, potentially offers a new technique for providing a safe assistive home environment for old and vulnerable people in case unexpected events occur [50]. It is based on the potential to determine and manage peoples' care or health status through remote monitoring of their behavioural characteristics and parameters concerning their interactions with and within their local environment [72]. Thus, any changes in peoples' normal activity over time, detected through sensors embedded in the home environment, can be recognised as unusual and responded to [72]. In addition, computer algorithms have been generated to envisage and detect users' activities in the home and to identify their behaviours, emotions and even gestures [42].

However, [50] argue that there is little comprehensive understanding of how to deliver complete and efficient lifestyle monitoring systems, notwithstanding the great numbers

of marketable worldwide installations. In other words, although all these health monitoring technologies offer considerable opportunities for the smart home, they also come with some risks and challenges concerning users' privacy and security [9, 73]. Many people are still hesitant to use sensing technologies in their homes, being afraid of allowing others to monitor their digital trails and make use of them [74]. However, they might be well received if they are properly introduced and used, cost-effective and properly resourced [75]. In addition, such monitoring might be acceptable if it enhances peoples' feelings of security and safety in the home, increases the available care options and supports the carers' role [76]. Thus, an effective realization of the smart home environment essentially requires such challenges to be addressed.

To dispel users' concerns, researchers should address such challenges by providing alternative ways to identify users without compromising their privacy and safety, such as by using digital representations with users' consent, instead of using video footage. Users should be ensured of their decisive authority to control the system by imposing restraints to prevent their home from making undesired decisions and taking harmful actions.

This further indicates that it is essential to benchmark the attitudes and opinions of telecare users, informal carers and care professionals before introducing any telecare system to their environment [77-79]. Research shows that the deployment of telecare is expected to yield important consequences [80, 81]. Examples of these consequences are electronic monitoring of patients, accurate and secure electronic patient records, reduction of cost and time by providing virtual consultations, thus eliminating the need for patients to go to hospital, as well as fewer admissions to residential care units.

Both energy management and health monitoring applications have attracted considerable attention from academic and industry researchers, mainly because of increasing healthcare costs and energy consumption, and recent technological advances in miniature devices, smart textiles and wireless communications [11] However, the existing smart home systems are not intelligent enough, for three main reasons:

- The wide availability of various services and devices in contemporary networked smart environments makes the management and deployment of dynamic smart home systems difficult without the intentional involvement of human actors, because devices have been innovatively used beyond their original design across several application domains (e.g. energy consumption for daily activity monitoring) and because different types of devices can achieve the same functionalities (e.g. user location can be obtained via GPS, RFID, WiFi) [82]. It has also been observed that the development of smart home services has begun to decouple from the development of smart home devices. Rather than being pre-bundled with smart home devices, smart home services have been developed and deployed independently, with the consequence that the systems demand significant manual works to create efficient linkages between services and appropriate devices, because of the large number of possibilities and combinations [22, 82].
- These service applications require a large number of sensors and actuators. In the context of the IoT, there will be even more of these [83], coexisting within one physical area (e.g. the home). Several sensor types (e.g. PIR) are found to be used across application areas such as home security, telecare and energy

management services. The same type of sensor and actuator is also used to detect and create different phenomena, depending on its context and application area. Different types of sensor and actuator are used to detect and create the same phenomena, thus augmenting the complexity of smart home systems.

- These systems lack the elasticity and scalability needed to deal with users' different requirements, needs and changing states [84]. They are unable to deal efficiently with dynamic contexts, where devices may be added or removed during runtime. This highlights the need for an open, intricate, agile and flexible architecture to deal efficiently with all the changing circumstances [26].

In summary, a large number of possible combinations of services and devices, as well as diverse user preferences, have made the management and deployment of dynamic systems difficult without the intentional involvement of human actors. This has led to interoperability issues and prevents various smart home components from communicating with each other unless certain gateways or adapters are utilized. This has in turn highlighted the necessity of having auto-configurable systems to connect the correct devices to application services and deliver smart home services appropriately and autonomously.

2.4 The Necessity of Auto-configuration of Smart Home Environments

Auto-configuration has become essential in any smart home environment, due to the complexity of the interactions among applications and devices [33, 85]. This complexity and the dynamic nature of smart home environments make it impossible, at the time of design, to take account of which components will interact in what ways in the

environment. Thus, it is essential that they can be configured during the deployment of the system, or auto-configured at runtime. Besides changing devices and/or their parameters, one important aspect of configuration is to adjust the connections among them using multiple communication options such as Ethernet, Wi-Fi and power-line, in line with the actual tasks to be undertaken or unexpected changes or failures in the physical environment.

The complexity of such management tasks is beyond the capability of most households. Thus, auto-configuration becomes essential to ensure the flexibility and robustness of smart home environments. In detail, it will be actively necessary to identify appropriate components for service applications and create appropriate connections in each situation without direct human intervention.

Compared to manual configuration, which requires intensive human effort and knowledge, in auto-configuration the correct interconnections among the components are decided automatically. Auto-configuration has also been defined as the ability to adjust the system's configuration dynamically to adapt to altered circumstances, hence to enable the adding, removing or modifying of entities, without interrupting the normal operation of the whole service [86]. Thus, significant efforts are required to maintain linkages between services and devices so as to cope with any changes of requirements of the target environment. This in turn requires considerable expenditure of money, effort and time for experts to maintain such complex systems.

Most of the endeavours to make environments 'intelligent' have addressed mainly the technical features of building elements, but lack of studying on the dynamic

interrelationships between the designed environment and its users. This has in turn made it difficult to address communication problems among users, their activities and smart systems, which may cause user dissatisfaction [29]. This does not achieve the flexibility and agility of smart home systems, which are intended to adapt to changes in environment and users' requirements.

Overall, the abovementioned issues have stimulated the need to design dynamic auto-configurable applications to provide communication models among these distributed systems autonomously. Recent research in smart home technology aims to enable interaction between smart home devices and the networking infrastructure with no obvious user control [14, 31].

The smart home systems must support the ubiquity of their services' operation. This means that any smart home service must always be available, notwithstanding any changes in the service environment. More specifically, a smart system needs to be automatically interoperable with resources in its current service environment, rather than statically pre-programmed for its environment. Manual configuration requires major inputs of money, effort and time for experts to maintain such complex systems. The smart home vision highlights the need of auto-configuration in smart home environments by indicating that users are not meant to manage the smart home system themselves, as professional administrators do with conventional distributed system[9].

The integration of automatic configuration technology into smart homes ensures peace of mind, increased comfort, health monitoring, safety and security, as well as economic

benefits (e.g. lower energy consumption) [6, 15, 87]. The necessity of the auto-configuration within smart home environments is further illustrated in Chapter 3.

This literature review shows that the full potential of smart home environments has not yet been realised, due to the complexity of the essential automation systems [9, 26, 49]. Achieving the auto-configuration of smart home systems is a complex task[88]. Adjusting the smart home system accurately to its users' requirements requires profound expert knowledge. Thus,[68] argue that much work is needed to establish the efficiency of automatic monitoring functions and automatic adaptation to users' changing needs.

To deal effectively with these issues, smart home environments require auto configuration to enable services to utilise any required device through any network within a given smart home environment, automatically, anywhere and at any time. It helps to deal with the intricacy of the interactions among smart home applications as it reaches levels beyond the human ability to control them, while ensuring the required quality of flexibility and robustness of smart home systems. This would allow them to integrate and cooperate continuously in order to meet inhabitants' objectives and fulfil their changing needs throughout the environment. Overall, this will serve the global trend towards providing flexible and agile services or products for users at low prices and at the appropriate times [25]. The configuration problem within smart home environments and the related work are further identified and reviewed in the following chapter.

2.5 Summary

This chapter has discussed the state-of-the-art of smart home environments which builds upon and benefits from advances in AmI, ubiquitous computing and home automation. The promise of artificial intelligence (AI) is that people will live in a networked environment that is adaptive, sensitive and responsive to their presence. To achieve such networked environment vision, the smart home environment system is required to consider different and obscure protocols that can be employed for different purposes.

One of the main challenges faced by developers of smart home environments is auto-configuration, mainly because the installation, configuration and integration of large and complex systems into the existing home environment is an intricate, time-consuming, and error-prone process, even for the experts. The smart control over home environments stimulates users to incorporate automation technology into their homes to increase comfort and economic benefits such as reduced energy consumption [6]. The vision of smart home systems aims to make the home environment aware, adaptable and able to appropriately collect and analyse a range of contextual information, responding to information requirements, to dynamic circumstances and to users' needs [11, 89]. This review has revealed the need for efficient automatic service delivery related to ubiquitous systems, due to the widespread availability of service applications.

Chapter 3: Auto-configuration within Smart Home Environments

3.1 Introduction

Auto-configuration technologies are required to enable semi-autonomous and/or autonomous service delivery and its operation within smart home environments. The main aim is to enable services to utilise any networked device within such environments anywhere and at any time.

The types of automatic configuration are characterized on the basis of target entities and presented in Section 3.2. The configuration solutions are categorized by the enabling technology and further analysed in terms of the identified criteria set out in Section 3.3.

These benchmarks were determined by the main aim of the present research, which is to achieve auto-configuration within smart home environments at runtime with no direct user intervention. Previous work is analysed qualitatively, as to whether these benchmarks were achieved or not.

This analysis identifies the technologies applied to achieving automatic configuration and highlights the advantages and disadvantages of approaches previously developed. This leads to the identification of a knowledge gap and the research problem. The chapter ends with a summary.

3.2 Types of Auto-configuration

Research into auto-configuration was mainly carried out in the web services area, followed by work on AmI and distributed robotics, either with help from web services or directly employing traditional AI techniques [85]. Within these research areas, the term ‘configuration’ refers to patterns of cooperation, collaboration and information exchange amongst multiple software components to learn and reason about the environment in order to execute actions when none is predefined [90-92].

Automatic configuration systems help to minimise the overheads of the existing configuration process with minimum user intervention [30]. Achieving auto-configuration in smart home environments could fulfil users’ need for an easy and flexible environment.

In the areas of AmI and network robotics, the concept of configuration is generally employed to refer to organisational and/or structural variance. Achieving configuration

is a key function of an autonomic system, defined as “a system that operates and serves its purpose by managing itself without external intervention even in the case of environmental changes” [48].

To clarify the scope of configuration in this research, it is essential to specify the targeted entities or components. The following tripartite classification is based on the entities that need to be configured. This analysis disambiguates the term ‘configuration’ as used in related research areas and clarifies the boundary of the present work: the configuration of software agents.

- Research into the configuration of *physical components* is concerned with robots assembled from numerous identical parts (modular robots) [93] configured according to a function within the swarm, forming a swarm of mini-robots whose capabilities exceed those of the individual units [94]. The collaboration of mini-robots in a swarm within an ecosystem to achieve the assigned tasks requires the properties of swarm intelligence systems, such as self-organization, flexibility and adaptation.
- Research into the configuration of *network nodes* [95-97] aims to achieve the optimal physical deployment of networked devices based on policies for the automatic generation of the most appropriate connections. An additional aim is to create routing paths amongst the networked components, particularly mobile devices such as smart phones. This helps to optimize the information pathway, the reliability and robustness of the network, as well as the quality of the service, with minimal energy consumption.

- Research into the configuration of *software agents* focuses on devising mechanisms for automatic role assignment and the coordination of agents in order to achieve shared, predefined goals [98, 99]. This configuration type represents the core concern of this thesis.

3.3 Configuration Systems Solutions

The evaluation process adopted here focuses on the ability of configuration approaches to meet the requirements of smart home environments and cope with their dynamics. The success of auto-configuration will enable the smart home environment to adapt quickly to changes in users' needs (e.g. because of changes in their preferences or medical conditions) in order to provide a comfortable and adaptable environment.

It is expected that the key gains from employing auto-configuration systems will be in the form of enhanced performance and reduced operating cost. The auto-configuration features will enhance the performance of the smart home system, thanks to the better adaptation to changing system characteristics. Thus, assessment criteria will focus on the methodologies to evaluate the gains that can be attained using auto-configuration systems: flexibility of automating smart home systems online with minimum or no user intervention and enhancing the overall performance of the smart home system.

Research has been done to achieve auto-configuration by concentrating on knowledge representation technology [14, 31, 100-103] and autonomous robotics technology [85, 104-108]. The criteria and performance metrics for the assessment are flexibility, reconfiguration type (offline or online) and configuration executor (machine or human). Flexibility indicates that the system deals elegantly with the complexity of smart home

environmental conditions [109]. Since the aim of this research work is to achieve auto-configuration within smart home environments without user intervention, we have added three other criteria: on-line configuration; the targeted environment to determine the domain of auto-configuration (whether it is a smart home or not); and the configuration executor, to determine whether the user should interfere to support the configuration process or whether it is a wholly machine-based process. These criteria are further illustrated in Section 3.3.3.

Although the user is certainly the final determinant of what the system does and how it does it, minimum or no user intervention is preferable, particularly in the case of dementia patients, for the sake of making their lives easy. The transfer of responsibility for an individual's wellbeing to software entails adopting safe, reliable and secure auto-configuration techniques. Appropriate levels of user privacy should be established, taking into account what is considered appropriate from their point of view. This could address the ethics of machine use in peoples' lives, as discussed in Section 4.4.

The identification of the criteria was based on the main aim of this research: to achieve auto-configuration within a smart home environment at runtime with minimum user intervention. Previous work is analysed in light of these criteria to determine whether these benchmarks were achieved or not. Fulfilling the smart home requirements will help to make the whole system more agile. Such auto-configuration ensures the up-to-date and smooth functioning of the smart home environments. This seamless operation of the smart home system leads to a calm home environment, which enhances the quality of the services provided within smart home environments. This in turn satisfies users' needs and reduces the overall system cost. All of the studies discussed above

have been analysed in terms of these benchmarks. The evaluation process adopted was also used for subsequent evaluation of the proposed approach. In general, auto-configuration studies can be categorised, according to the enabling technology employed, as follows:

3.3.1 Configuration Systems Based on Knowledge Representation Technology

Service-Oriented Context-Aware Middleware (SOCAM) architecture is proposed for the building and rapid prototyping of context-aware services[102]. SOCAM provides an effective support for discovering and accessing numerous contexts to construct context-aware services. A formal context model based on ontology using the Ontology Web Language (OWL) (discussed in detail in Section 6.5.1.1) has been built to tackle issues such as context classification, semantic representation, context reasoning and dependency. However, test results indicate that the ontology reasoning time is much longer than the user-defined rule-based reasoning time.

An architecture for auto-configuring system has been constructed using knowledge representation technology [101]. It was quite easy, taking a model-driven approach, to transform the behavioural models and message sequence charts into a framework of executable code to which the more specific functionality (including execution of the OWL-based framework of the SW (OWL-S) ontology) could be added. However, some of the subtests failed, so the architecture needs to be improved.

A new proposed approach to achieving fast reconfiguration of modular manufacturing systems relied on an ontology-based reconfiguration agent without human intervention [100]. However, the problem-solving skills of the proposed agents would need to be

improved to solve more intricate problems. This could possibly be achieved by extending the reasoning engine architecture or adding new components which could use entirely decidable reasoning capabilities and ontology web language description logic (OWL DL).

Autonomic communications architecture has also been proposed to solve the auto-configuration problem. Using the OSGi framework, an autonomic context-aware element was developed in order to identify and personalize the service offer for a particular user [110]. The autonomic element senses devices linked to the home network considering user preferences. The services are modelled in OWL ontology.

The autonomic element also uses Semantic Web Rule Language (SWRL) reasoning to infer suitable services and offer a personalized service to each user [111]. Although the autonomic element has provided successful results, it must evolve to a more mature state to deal with more complex issues.

An ontology-based infrastructure called Sixth-Sense has been proposed to facilitate quick prototyping of artefact-related applications [103]. This approach is based on Semantic Web technologies including ontology to represent significant aspects artefact-human communication and reasoning of high level context from data collected by sensors. Although experienced users found it easy to customize the ontology, it was difficult for less experienced users.

Another novel approach called the Smart Home Ontology Model has been proposed to build a smart home in conjunction with an autonomic system whose aim is to monitor the home and its residents [31]. In particular, it aims to provide the elderly with

intelligent support and assistance in any circumstances at any time using wireless sensor technologies to control devices and other smart home components as required. However, this model needs to be enhanced to make reasoning components more complete and to provide more intelligent decisions and actions.

Shen et al have proposed an ontology-based approach to represent Product Extension Services (PES) knowledge configuration and developed a PES configuration system [34]. This was achieved by implementing efficient configurations through unambiguous sharing and reuse of knowledge. Nevertheless, this approach can be considered only a partial solution to the configuration problem, as configuration optimisation and reconfiguration were not considered.

An Autonomic Ontology Driven Architecture (AODA) has been proposed for the auto-configuring of resources, utilizing service- and event-oriented communications [112]. The AODA follows a top-down approach and is implemented as an ecosystem-wide ontology intended to characterize the properties of services and events relevant to producers, entities and consumers, and to participate in the dynamic collaborative environment. However, more validation of the AODA in more complex scenarios is required to manage energy consumption in the smart metering business case, as it involves a larger number of connected heterogeneous machines.

3.3.2 Configuration Systems Based on Autonomous Robotics Technology

Recent research has investigated the principles and techniques needed for auto-configuration within the field of autonomous robotics. McKee has proposed a novel Task-Directed Configuration of Networked Robotic Agents [104]. This model

integrates the concepts of a task factory, which produces task modules, and a set of modules demonstrating robotic components distributed in the environment, such as sensors and actuators. It specifically focuses on auto-configuring networked robotic agents where robotic components are extracted from a common pool and assembled in relation to a high-level task description. In this method, ad hoc robotic architectures are automatically created with just the components needed for each specific task. Task descriptions include information about the kinds of components required, about their physical location and about their requisite connectivity. However, this model needs to be further improved to develop the capability to form more intricate robotic agent architectures.

A plan-based approach is proposed to control the smart environment and support interaction between it and its users [113]. Users in this approach need not learn how to operate all devices and control their functions, but simply communicate their needs to the environment. Planning is utilized to develop strategies on the ways in which each function can be performed on each device to accomplish the users' defined goal. This approach is very similar to traditional action planning, rather than configuration planning.

Automated Synthesis of Multirobot Task Solutions through Software Reconfiguration (ASyMTRe) is proposed to increase the autonomous capabilities of heterogeneous robot systems [105]. It enables a robot coalition to connect schemas dynamically within and across robots to perform a single-robot task by coalitions of multiple robots. The approach is based on a centralized ASyMTRe configuration algorithm and the ASyMTRe-D negotiation protocol. The distributed negotiation process provides a more

flexible and robust method for establishing coalitions. On the other hand, it introduces a trade-off between robustness and solution quality.

A reactive approach to auto-configuration based on the ecology of physically embedded intelligent systems (PEIS Ecology) has been proposed by [85]. This approach employs SW services to independently generate a configuration to execute a cooperative navigation task, automatically changing this configuration when any component fails. One disadvantage is that it may produce non-optimal configurations. Another is that it may fail to identify a configuration which exists.

Lundh et al. have proposed a plan-based approach to automatically generate a desired configuration of a robot ecology, set of resources and environment [106]. In particular, hierarchical task planning was applied to configuration generation [107]. Configuration methods have a body which lists all the functionalities which comprise them. These functionalities are the system's basic components, defined by their functioning parameters, their input and output signals, and the preconditions which must hold in the ecology for their correct functioning. Configuration planning is achieved by a best-first search of the set of functionality instances which conform to the functionality of the body of each configuration method. This type of search allows activation of the configuration with minimum cost for every planned action. The state of the ecology is automatically attained at design-time and is also monitored during implementation in order to reconfigure it if any functionality fails. They have validated their approach by showing that a scenario which was previously hand-coded can currently be run autonomously in the adopted PEIS-Ecology testbed.

3.3.3 Configuration System Solutions Comparison

This extensive analytical review of the literature shows that auto-configuration is still a challenging research problem, as having it conducted by experts is a time-consuming and costly process and there is still no satisfactory solution. Above all, no proposed solution is sufficient to achieve auto-configuration within the smart home environments. These solutions are deemed incomplete, as they lack the ability to meet the requirements of the smart home environment as listed below. Table 3.1 lists the contributions (previous approaches) to the research literature most closely related to the present study, using the identified criteria to compare them. These terms are defined as follows:

- **Flexibility:** The system should deal elegantly with the complexity of smart home environmental conditions and flexibly perform in more dynamic situations to meet its goals without disruption. This entails that the smart home system needs to be able to interpret any changes it detects and extract the relevant contextual data. It needs to validate any new information or knowledge and encode this data into an appropriate format.

By utilising available knowledge locally or remotely, it must be able to reason a configuration plan based on the new information. It must efficiently infer new facts about more complex situations to meet the system's goal, based on the smart home ontological model, so that the system can adapt to the changed environment. It should be able to avoid contaminating the knowledgebase, by isolating and eliminating any inconsistent situational information. According to its context, the system should be able to change its operation or behaviour, i.e. its configuration, state and functions.

This means that the system needs to be aware of the digital and physical environment. It needs to monitor its operational context and directly sense any changes in the digital or physical environments. These changes include new devices or services joining the environment, existing ones leaving it, and the context of any device or service changes.

- **Configuration type:** Configuration is either executed online, while the system is working, or else the system has to be halted to perform the configuration offline. Online configuration is executed without affecting the system, which continues running even when the configuration of any changes in the smart home environment is applied.
- **Targeted environment:** Since the aim of the present research is to achieve auto-configuration within smart home environments, the configuration environment indicates whether the area of auto-configuration is a smart home or not (e.g. a factory).
- **Configuration executor:** The system needs to auto-configure efficiently and generate a proper linkage between the devices and services without direct user intervention, so that the operation of the system is not disturbed. This helps when the configuration needed is beyond users' capabilities, making life easier for them.

The approaches reviewed have some similarities with the one presented in this thesis. However, our approach differs from them in that it integrates two enabling technologies to achieve better results. This section shows that the research can benefit from

ubiquitous robotics and knowledgebase technologies in the home environment. It can be considered an effective approach to achieving auto-configuration, because it is able to intermediate between services and users in order to satisfy their individual needs. It could satisfy users' needs by positively enhancing and facilitating their home activities without disturbing their peaceful lives or intruding into their privacy. It helps smart home applications to provide users with the necessary services, accurately and inconspicuously, when and where needed.

Although the approaches reviewed above have accomplished good results, none of them has met all the requirements of the smart home environment; each fails to meet one or more of them. As shown in Table 3.1, if the approach meets the flexibility criterion, it either requires user intervention or is not conducted at runtime. In other words, there is no adequate solution to allow the smart home system to adapt dynamically to changing circumstances, hence to enable the correct interconnections among its components without user intervention and without interrupting the whole service. In short, the above-mentioned technologies and approaches have not yet evolved to a sufficiently mature state.

Table 3.1: A Comparison of Configuration-related Research

Technology	Reference	Approach Name	Flexibility	Configuration Type (Online)	Targeted Env. (Home)	Configuration executor (No Direct User Intervention)
Knowledge Representation	[102]	“A service-oriented middleware for building context-aware services”	✓	X	✓	X
	[101]	“A Semantic Web-driven approach to self-Configuring Computer Systems”	X	✓	✓	✓
	[100]	“An ontology-based reconfiguration agent for intelligent mechatronic systems”	✓	X	X	✓
	[106]	“Dynamic auto-configuration of an ecology of robots”	X	X	✓	✓

Knowledge Representation	[110]	“An autonomic approach to offer services in osgi-based home gateways”	✓	✓	✓	X
	[103]	“An Ontology-based Programming Platform for Smart Artifact Systems	✓	✓	X	✓
	[31]	“POSTECH's U-Health Smart Home for elderly monitoring and support”	✓	✓	✓	X
	[34]	“Configuration of product extension services in servitisation using ontology”	✓	X	X	✓
	[112]	“AODA: An Autonomic and Ontology-Driven Architecture for service-oriented and event-driven systems”	X	✓	✓	X

Robotics	[104]	“Task-directed configuration of networked robotic agents”	✓	X	X	X
	[113]	“Smart environments and self-organizing appliance ensembles”	X	X	✓	X
	[105]	“Building multirobot coalitions through automated task solution synthesis”	X	✓	X	X
	[85]	“Reactive auto-configuration of an ecology of robots”	X	✓	✓	✓

To bridge these research gaps, we propose a Knowledge-based Auto-configuration Software Robot (Sobot) for Smart Home Environments as a way forward to enable auto-configuring systems. The powerful features of the Sobot (described in detail in Chapter 4, Section 4.3) allow the Sobot to perform auto-configuration and meet the requirements of the smart home.

The proposed approach should also be able to automate the activities associated with customised service delivery for dynamic home environments, reliably and precisely, to facilitate context-aware services. For instance, automating the activities associated with medical services accurately and reliably is essential in order to provide patients with user-friendly services and to reduce the overall complexity of controlling their homes. This in turn reduces or removes the financial consequences of frequent journeys to hospital.

3.4 Summary

Auto-configuration has been defined within the scope of the present research. The types of automatic configuration were characterized in terms of target entities, to specify scope of configuration applied to this research. The most closely related auto-configuration work within the fields of knowledge representation and robotics technologies were thoroughly reviewed. This extensive assessment identified the advantages and disadvantages of these approaches, allowing the identification of a knowledge gap and the research problem.

Based on the criteria identified, relevant auto-configuration approaches were evaluated. It is evident from this review that there is no adequate solution to allow the smart home system to adapt dynamically to changing circumstances, hence to enable the correct interconnections among its components without user intervention and without interrupting the whole service. No published study has so far tackled this problem using a combination of both ubiquitous robotics and knowledge representation technologies, which is considered to be a novel feature of the present research. These technologies will be discussed in Chapter 4.

Chapter 4: Enabling Technologies for Auto-configuration within Smart Home Environments

4.1 Introduction

The proposed auto-configuration solution in this research is mainly inspired by ubiquitous robotics systems and knowledge representation (KR) technology. The author considers these enabling technologies the way forward to achieve auto-configuration within smart home environments.

The ubiquitous robotics revolution has been accelerated by the development of autonomous robotics and intelligent environments, giving rise to the multidisciplinary search field of intelligent robotic environments or ubiquitous robotics, which merges ubiquitous computing and ubiquitous environments with robotic technology.

There has been a noticeable propensity recently to exploit ubiquitous robotics technology to deal effectively with the configuration issue within smart home environments (the aim of this research) and to enable services to utilise any required device through any network within a given smart home environment, automatically, anywhere and at any time[54, 114-118].

The evolution of robotics technology is discussed in Section 4.2, with particular consideration of three revolutionary generations of robotics, namely industrial robotics, service robotics and ubiquitous robotics. The features of the Sobot are discussed in Section 4.3 and robot ethics in Section 4.4.

The knowledgebase plays a pivotal role in assisting the Sobot to automatically select and configure the most appropriate resources within smart home systems, enabling it to share a common understanding amongst software agents as well as humans, to clarify the domain assumptions and to facilitate the reuse of established knowledge [119]. Section 4.5 is concerned with KR enabling technology. The definition of KR in the context of AI research is reviewed in Section 4.5.1, then Section 4.5.2 offers a more detailed review of ontology, one of the most suitable KR techniques for use by intelligent agents and by the Sobot which is the subject of this thesis. The chapter ends with a summary.

4.2 Evolution of Robotics Technology

During the last half century, in response to the development of the social and commercial needs of human beings, robotics technology has undergone a profound revolution which can be represented in terms of three generations [116]. A robot is

defined as “an autonomous machine able to process information elicited from sensors, upon which it can make its own decisions and act in the environment” [120]. Autonomy in robots is defined as “the capacity to operate in the real-world environment without any form of external control, once the machine is activated and at least in some areas of operation, for extended periods of time” [121].

The first generation was dominated by industrial robotics (e.g. manipulators performing tasks such as assembling, polishing, welding and painting) and the second by service robotics (personal robots, domestic robots, medical robots, humanoid robotics etc.), leading to the current exploitation of ubiquitous robotics. This third generation is the outcome of developments merging ubiquitous computing and robotic technologies [122]. Industrial robots were introduced into production lines in the early 1960s and remained dominant until the 1990s. The Robotic Industries Association defines an industrial robot as “an automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes which may be either fixed in place or mobile for use in industrial automation applications” [24]. The first one was manufactured by Unimate and was installed by General Motors in 1961.

They were used to free human beings from hazardous and harmful tasks, to enhance the speed and accuracy of tasks such as painting and welding, and to reduce the cost of various manufacturing processes [123]. The industrial robot is pre-programmed and has relatively simple tasks such as manipulating and moving objects and cooperating with the environment [116].

However, it has been recognized that there is a need to develop more flexible robotic systems for use in unstructured environments. Compared with industrial robots, service robots take advantage of recent advances in mobility, perception and algorithmic research, which have the control, sensing and decision-making abilities that are necessary for them to work in unstructured, three-dimensional environments. These features enable robots to localise in two-dimensional maps of the world and to navigate unknown two-dimensional environments. Since 1995, developments in service robotics have allowed the construction of animal-like robots and mobile robots. Figure 4.1 illustrates the evolution of robotics over the last five decades.

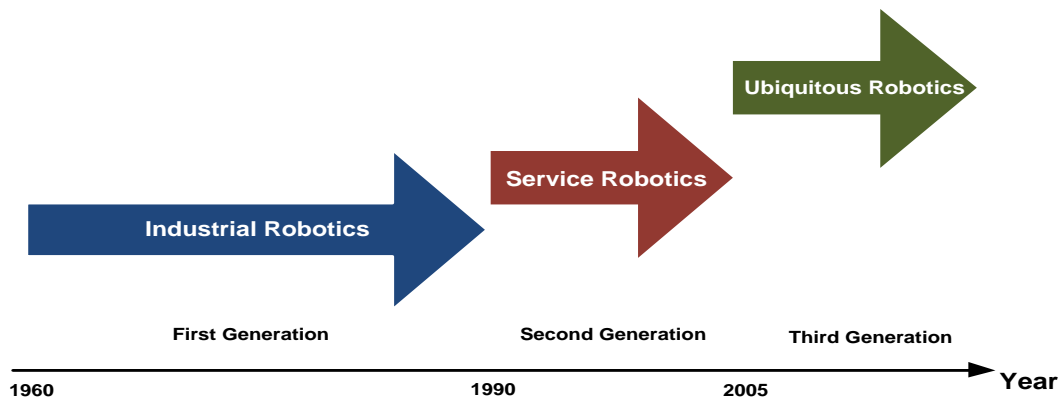


Figure 4.1: Evolution of Robotics

These technological advances and increasing social demands are the main driving forces of new applications emergence such as surgery assistance, automatic refuelling and rehabilitation, home cleaning and museum exhibitions [116]. The International Federation of Robotics identifies a service robot as one which operates fully or semi-autonomously to deliver services useful to the wellbeing of humans and equipment [124]. Service robots can execute daily tasks, assist people with disabilities and work as

companions or pets. Other types are surveillance robots, military robots, security robots, construction robots, field robots, medical robots, rehabilitation robots, and office and tour guide robots.

Ubiquitous robotics technology is the latest generation robotic technology utilising the ubiquitous devices (e.g. sensors, actuators) embedded everywhere in the inhabited environments and even inside human bodies via networks. Through these devices, the ubiquitous robots can gain better picture of the world and manipulate the world beyond its own capabilities [125]. The ubiquitous robotics revolution has been accelerated by the development of autonomous robotics and intelligent environments, giving rise to the multidisciplinary research field of intelligent robotic environments or ubiquitous robotics, which merges ubiquitous computing and ubiquitous environments with robotic technology [54].

It can be observed that the robotics revolution is driven by the more and more complex tasks and higher expectation such as able to communicate autonomously with users, understand their requests and provide them with services accordingly. The usage of knowledgebase has increased with each robotics generation [126]. Knowledge extraction, representation and usage can enable a grounded and shared model of the world appropriate for high-level tasks [127]. The main challenges are how to extract knowledge from natural languages, to benefit from several areas of knowledge in decision making and to recognize what knowledge is missing.

Bearing in mind that it would be virtually impossible to equip a robot with completely comprehensive knowledge before it was put to use, robots need to be developed to be

able to acquire missing knowledge automatically from somewhere at runtime, to achieve the tasks set by users. Thus, the DARPA Robotics Challenge (DRC) aims to improve semi-autonomous robots which are able to perform “complex tasks in dangerous, degraded, human-engineered environments” [128].

Robots should also be able to use various types of knowledge, since they might need to encompass several areas from more than one knowledge resource to achieve a single task. They should be able to identify gaps between the implicit forms of knowledge and experience. They already possess and that required for the task at hand, then be able to search for the pertinent knowledge from open sources.

These requirements have stimulated continual research efforts to develop intelligent robots, which are a part of ubiquitous robotics technology. In this respect, cognitive systems make an on-going contribution to increasing robots’ cognitive capabilities, enabling ubiquitous robots to automatically acquire and utilize shared knowledge and to reason based on decision-making mechanisms to achieve runtime tasks. Thus, ubiquitous robots are able to collectively comprehend the users’ needs or preferences, even when no direct orders are given, and provide a constant and seamless service [114].

4.3 Software Robots (Sobots)

Ubiquitous robotics builds on the top of the robotic technology and ubiquitous computing technology, and works within a ubiquitous environment. The main advantage of the Ubibot system lies in its ability to gather intelligence from the real world by extracting information benefiting from its action and perception abilities. A

ubiquitous robotics system, which offers services to users to satisfy their individual needs, is structured as a collective, integrating three different types of robots: Sobots, embedded robots (Embots) and mobile robots (Mobots) as shown in Figure 4.2. These abilities are manifested in its components, where the Sobot focuses on intelligence, the Embot on environmental perception and the Mobot on action execution [129].

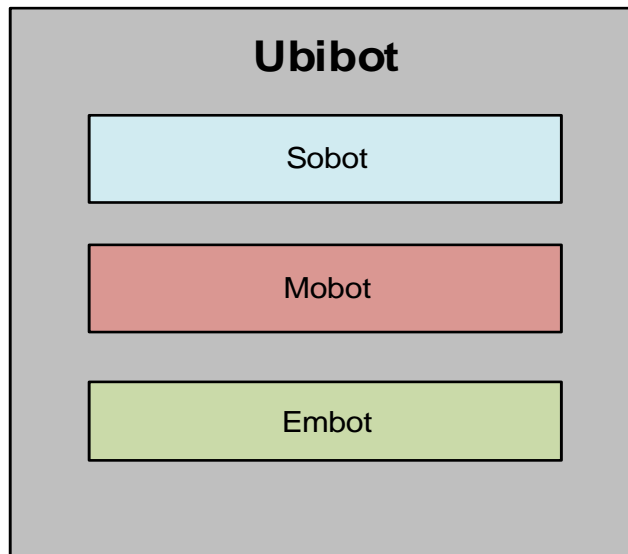


Figure 4.2: Ubibot Components

Each Ubibot has particular individual intelligence and roles, and can interact with the environment throughout networks. In this field, tasks are executed through the collaboration of various simple networked devices, providing services anywhere and at any time.

A robotic system providing services seamlessly can automatically utilise the sensors and actuators in surrounding environments [130, 131]. Ubibots are able to acquire missing knowledge automatically in order to fulfil the demands of users at runtime.

The aim of ubiquitous robotics is to provide seamless services, even in a changing environment, by means of utilising Sobots, Embots and Mobots [122, 132]. Ubibot system, encompassing these robotic elements in their various forms, can form a networked cooperative robot system.

However, the review will address only the main characteristics and functionalities of the Sobot, where the system's core intelligence is embedded, in accordance with the research aim of achieving automatic configuration within the smart home environments. The main reason is that the Sobot is able to recognize users' needs and provide a continuous and seamless service, without the need for direct commands [133]. This implies its ability to infer new knowledge about users' needs and their environment.

A Sobot is a virtual robot with embedded social and cognitive intelligence, which enables reactive behaviour driven by events, as well as proactive behaviour based on its collected intelligent. It acts as the brain of a real-world robot to control it, and interacts with human beings without the help of other types of Ubibot. The main features of a Sobot [134, 135] are context-aware intelligence, context-aware self-learning and calm seamless interaction.

The Sobot is context-aware, self-learning and autonomous, i.e. able to govern and control its behaviour without external orders. It can learn new skills and represent the user as a manager coordinating behaviours within the ubiquitous robotics environment in terms of its behaviour and communication with others. Context-awareness is defined by Dey and Abowd as "any information that can be used to characterise the situation of an entity, where an entity can be a person, place, or physical or computational object"

[136]. They add that context-awareness or context-aware computing is “the use of context to provide task-relevant information and/or services to a user, wherever they may be” without the need of explicit user intervention [132, 137, 138]. In addition, the Sobot is able to “learn” and assimilate motions, objects and situations. Its learning process can be continually developed, as it may need to alter its behaviour within the ubiquitous environment once a new object is introduced.

A Sobot has context-aware intelligence being a proactive software robot able to exhibit rational behaviour. It can operate with clear goals and well-designed plans which are continuously revised. Proactive computing was proposed to establish ways for programming computers to work on behalf of people [139]. This means that context-aware computing is driven by the need to use computational tools to compensate the weaknesses and deficits of human cognition, i.e. memory, comprehension, attention and decision-making. Thus, for such context-aware systems to guide users through particular activities there is a need to establish models of human behaviour based on rationality and predictability [140].

Diverse computational resources have been embedded in the environment to provide the required information and services at the right time and in the right place. These embedded sensors detect, identify and provide data about any changes in the environment and its users’ status. Besides that, algorithms have been developed to analyse the data provided and to make inferences about the required actions. This means that such context-aware systems need to be effective in a dynamic environment, as people’s everyday behaviours are unpredictable. Despite the great potential impact of

such computational systems' ability on peoples' lives, they have some significant ethical implications which will be discussed in Section 4.4.

The Sobot is also adaptive in the sense that it can adjust and respond to new situations instead of using a predetermined response to probable or anticipated ones. It can perform rationally, in that it does not merely repeat the same tasks, but alters each task as needed to generate an outcome appropriate to the context. The Sobot stays context-aware at all times so that it can adapt and adjust itself to the context whenever required. Whenever it perceives a change in the environment, it can detect and identify users automatically by observing their behaviour, gathering information about their status and the environment, as well as authenticating them, so that it can easily adjust itself to their preferences and interests [141-143].

In the context of the Sobot, calm sensing can be seen in terms of Weiser's observation that "the most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it" [54]. The age of calm technology is "when technology recedes into the background of our lives" [125] and "informs in detail but doesn't demand users' focus or attention"[144].

Calm technology provides users with a familiar, well-functioning environment, in which they are effortlessly tuned into everything that has happened, is happening or is going to happen around them, via numerous familiar details [144, 145]. Calm sensing helps greatly in developing assistive applications using Sobot systems to provide constant monitoring of vulnerable, elderly and disabled people. This vision of calm technology symbolizes a world of awareness, comfort and peace. In other words, users

will be kept permanently up-to-date with what has happened what is occurring in their environment and what will happen; crucially, however, such information will be available and evident to them only when needed and will simply disappear into the background when not needed. This further indicates that instead of making it an engaged living environment where people are proactive in their environment, it has become calm living, where computers rather than users are proactive [89].

The Sobot's features make it more powerful than an intelligent agent for auto-configuration. An intelligent agent is an autonomous entity that can observe via sensors and direct its responses accordingly using actuators towards accomplishing goals [146]. These have been used mainly to automate the activities of customized service delivery for dynamic home environments, since they encompass the features necessary for achieving auto-configuration.

In more detail, the Sobot is context-aware, providing pertinent information in the right form at the required time and place. Context-awareness helps in filtering environmental information and allows the Sobot to work automatically. It helps systems to become 'aware' and to make decisions intelligently, reacting to events as they occur.

The Sobot with decision-making abilities can determine suitable actions based on the collection of data from numerous sources in a complex way [147]. It can recognize the prevalent context and in light of this, make decisions based on its inference engine, relying on a knowledgebase, then implement them with no need for direct consultation of the user each time. Therefore, it is easy to update its knowledge base at runtime.

Sobots have been applied in a system called NewsAlert [148] to detect and alert managers by delivering a personalized electronic newspaper as an avatar able to talk to users, who by altering the agent's pitch, voice and speed can generate an attractive virtual persona. It has also been implemented as a virtual pet which can acquire actual information via the vision and voice system known as DogEar [149]

4.4 Robot Ethics

This expansion of computational systems' ability to make decisions affecting people's lives raises some ethical issues, because such systems have been evaluated only in terms of their efficiency in achieving the goals for which they were designed, regardless of any social implications. This issue has been termed 'machine ethics' or 'computer ethics' [150-152].

The most noteworthy ethical issue is the breach of users' privacy. There is a risk that such ubiquitous techniques may intervene in an uncontrolled way into the lives of disabled people, particularly in tracking, recording and monitoring too much of their private lives [153]. This could have negative social implications such as breaching vulnerable people's privacy by videoing them while they sleep [154]. It seems that the use of ubiquitous technology presents security and convenience, but at the expense of users' privacy. This means that appropriate levels of privacy should be established, taking into account what is considered appropriate by users. On the other hand, argue that issues such as social acceptance and human-robot interaction cannot be researched thoroughly.

Another social implication is the concern among some experts that people may become dependent on the technology which makes decisions, plans activities and does specific jobs on their behalf [152]. The long-term consequences of robots looking after the elderly, children and sick people have not attracted much attention [120]. Indeed, these concerns apply not only to vulnerable people but, also to people in general, as our increasing dependence on such technologies will influence our ability to learn, think and remember [89].

However, there are still no successful accomplishments in either AmI or context-aware computing, which indicates how challenging it can be to develop a machine to act like a human being [53]. This raises the concern of how effectively such telecare and eHealth systems could enable more precise data to be sensed, collected and used. It still requires great effort to understand, infer and adjust to changing environments and their impact on behaviour patterns in significant ways in real time, and thus to be accepted by people in their normal lives.

There is an argument as to whether robots can be implicitly or explicitly ethical [150, 155]. Robots can be made implicitly ethical by creating software designed to avoid unethical consequences and implicitly promote and support ethical behaviour [150]. In other words, computers could be considered implicitly ethical agents if they could differentiate between what actions are safe and permissible and what are unsafe and prohibited from transferring personal information and treating people well, thus allaying critical concerns over reliability and safety [156]. For example, instead of video footage, digital representations might be used to ensure users' anonymity [117]. Computers

should be limited to performing specific tasks. For instance, computers with detailed medication databases should prescribe medicines reasonably, so as not to cause harm.

Developing an ethical robot has been found to be challenging [157] perhaps because computers are purely syntactic machines, incapable of human understanding [158]. Research has revealed the difficulty of developing explicit, fully ethical computer agents, due to conflicting beliefs about what is considered ethical and the intricacy of programming computers to be fully ethical [120, 150, 157].

4.5 Knowledge Representation Technologies

4.5.1 Definition of Knowledgebase

Knowledge representation under AI research aims to represent knowledge in symbols to facilitate inference new elements of knowledge from the present knowledge elements [159-161]. The symbols are specifically concepts, facts and relations among them relevant to a given subject area, plus any mechanisms to be used to integrate them in order to resolve problems in that area. Compared with the human mind, computers do not have a clear technique to acquire knowledge autonomously and represent it internally [160]. Instead, they rely on humans to formulate their memories in the form of knowledge and to specify how to represent it.

Knowledge representation is a topic under-developed in both cognitive science and artificial intelligence (AI). The cognitive science is concerned with the way people process and store information. It is the theory base behind KR, because studying human thinking, particularly mental states, representational structures and computational procedures, i.e. thinking, reasoning and operating on them in the mind, is a key issue in

this discipline [146]. It is assumed in the cognitive science that the human mind can be mentally represented similar to computer data structures. Thus, the mind function can be simulated by computational algorithms.

AI is the area of computer science that investigates the nature of human understanding, knowledge and mental abilities, and applies theories to software [159]. AI aims to develop software programs to perform tasks that simulate human behaviours. In other words, it tries to approximate human reasoning outcomes by running programs to structure, represent, encode and manipulate heuristic and factual knowledge[162].

The *AI techniques* (e.g. object-attribute-value triplets, fuzzy facts, uncertain facts, semantic networks, rules, frames and ontologies, etc.) are utilised to represent knowledge in intelligent systems include *Object-attribute-value (O-A-V) triplets* are employed to represent facts about specific objects and their features [159]. An O-A-V triplet affirms an attribute value of a specific object where objects normally have more than one attribute. *Fuzzy facts* represent uncertainty by using the imprecise and unclear terms generally found in natural languages. In other words, the same term may be interpreted differently by people, as they may have different understandings of it [159]. *Uncertainty of facts* can be described by a simple but commonly utilized extension of O-A-V triplets. A *certainty factor* is defined as a numerical value given to a statement which represents the degree of belief in that statement [163].

Semantic networks provide a knowledge representation technique which aims to reflect cognition [164-166]. They are normally graphs of concepts, objects and situations in

some particular domain of knowledge (the nodes in the graph), joined by relationships of any kind (links or arcs). The network is used to represent several interrelated facts.

Rules are another knowledge representation technique that relates one or more conditions to one or more actions. The conditions are represented in the IF part of the rule, whereas the actions in the THEN part. This means that the actions may be inferred from the conditions when the statements are true. A *frame* is a knowledge representation technique and a data structure used to represent stereotypical knowledge of some concepts or objects. Frames are comparable to objects represented in object-oriented programming and are of two types: *class frames*, used to represent common attributes of similar objects, and *instance frames*, used to represent particular instances of a class frame.

These knowledge representation techniques serve as a medium for human expression, substituting real objects in the world inside an intelligent entity and indicating a partial belief about the way people infer new knowledge intelligently; they also serve as a means for effective computation and make a list of ontological commitments about the world [167]. For example, rules, logic, frames and so on all represent a conception about the types of objects that are essential in the world, i.e. commitments about how to view it. Rules involve inference logic about O-A-V triplets, while frames represent objects and their interactions and are appropriate for taxonomic reasoning.

There is no single ideal KR technique appropriate for all applications [168]. This indicates that developers of intelligent systems should choose the KR technique that best suits the application being developed. In other words, the technique that can end up

driving the application should be selected as the technique chosen can define outcomes through the way that it works. This highlights the importance of being aware of the various techniques to select from. The rationale behind selecting ontology rather than other technologies in this research is discussed in the following section.

4.5.2 Ontologies

Ontologies have lately gained major significance in knowledge representation [34, 112, 119, 169, 170]. They have been found a promising method for modelling contextual information because of their formal expressiveness and the potentials for applying reasoning techniques [171, 172].

4.5.2.1 The Ontology Concept

Ontology is a simple abstract vision of a specific domain which is represented for a specific purpose in the form of a conceptualization of things and the relationships among them. It is also explicit, as all its concepts and constraints are clearly defined; it captures consensual knowledge and is understandable by machines. It consists of a set of concepts, their relations and rules about a domain. Ontological analysis, as an extremely important part of knowledge, illustrates the structure of knowledge into a hierarchical classification or categorization of concepts of knowledge.

Webster's Revised Unabridged Dictionary (1913) defines ontology as "That department of the science of metaphysics which investigates and explains the nature and essential properties and relations of all beings, as such, or the principles and causes of being" [173]. The concept of ontology has been incorporated into many AI technologies; in the realm of AI, it refers to a detailed description of a part of the world in a software

program. Significantly, it was developed to facilitate the sharing and reuse of knowledge in a real-world semantics [167, 174, 175].

In fact, Gruber’s definition of ontology as “a formal explicit specification of a shared conceptualization” [176] has been adopted by many researchers [177-179]. Here, a conceptualization means an abstract simple vision of the world which is represented for a specific purpose. Thus, it is a specification characterising the conceptualization in a very clear form. It is also explicit, as all its concepts and constraints are clearly defined. While “formal” signifies that the ontology should be understandable by machines, “shared” means that it captures consensual knowledge.

Although the Gruber’s definition is of great importance, it falls short in respect of “real-world semantics”, which is covered by Fensel defined real-world formal semantics, as well as consensual terms, intertwined with both machine and human understanding [180]. He highlighted its importance as a facilitator for sharing and reusing ontologies amongst machines and humans.

4.5.2.2 Ontology Features

The knowledge representation and reasoning is significant because it is beneficial to explain the behaviour of complex systems by employing a vocabulary which refers to beliefs, desires, and intentions and so on. It also makes particular problems easy to resolve. Ontology is a vital complement to knowledge about a particular domain as it illustrates the classifications of things, their known terms, their relationships, the applicable axioms and the constraints in the domain [159]. It is used to provide the

following useful features for intelligent systems and knowledge representation [181-183]:

- Ontology provides a *vocabulary*, defining the terms in a subject area. Vocabularies are finite lists of terms and their meanings which are identified in natural language and may be understood differently by people. Unlike human-oriented vocabularies, ontology provides *logical statements* to define the terms and their probable relationships, as well as specifying *the rules* for joining these terms and their relations. Consequently, ontology provides a machine-processable *common understanding* of the domain's terms. It also provides a hierarchical categorization (*taxonomy*) of the classification of concepts within a domain in a machine-readable and machine-processable form (controlled vocabulary), based on shared ontological characteristics.
- An ontology is considered typical of *content theories*, as it specifies classes of objects, their relations and conceptual hierarchies in a certain domain in an elaborate way, using a particular *ontology representation language* [184]. A well-developed ontology enables different types of consistency checking of any contradiction in applications and enhances interoperability among various software applications [170].
- The major purpose of ontology is *knowledge sharing* and *knowledge reuse* by intelligent agents and applications. Sharing ontology enables the building of a certain knowledgebase that describes a specific situation, thus allowing the integration of information from different sources, which may be accessed by

distributed applications to obtain all the available information. However, these sources may provide the required information in various formats and at different levels of detail. This requires that the same ontology is recognized by all the sources, to enable automatic data conversion and the integration and interoperation of information.

Ontologies are widely used in various applications of computer science, including in the field of the Semantic Web (SW), where they play a central role by providing formal models of domain knowledge which can be used by intelligent agents [37, 185-188]. Ontology-based approaches largely support context awareness and the integration of different resource networks [6]. They utilize knowledge-based systems to employ inference and can be built using artificial intelligence modelling techniques such as first-order logic and description logic [169, 188, 189].

Because of its features, ontology is considered a suitable KR technique for auto-configuration tasks [112, 119, 183, 190] which is the aim of this research. Auto-configuration requires systems that recognize when and where unexpected behaviour happens, analyse the problem context and base plans and decisions on the knowledge provided from the context. In particular, auto-configuration requires:

- Human knowledge to be presented and abstracted in a way that people can understand and machines can process (controlled vocabulary).
- Knowledge to be shared by humans and machines.
- Knowledge to be reused by machines.

This clearly indicates that the notion of knowledge is significant for the autonomic auto-configuration system which can be well represented by means of semantic models like ontology. It is also essential to suggest approaches which simplify the design and development of intricate and heterogeneous systems [30].

In particular, it serves as a framework to describe the various elements of the smart home environment. The ontology facilitates the sharing and reusing of automated knowledge between human and computer agents to achieve semantic interoperability thanks to its potential to interlink machine and human understanding using semantics. However, the machine does not attain a real understanding; instead, human understanding is encoded in a way which enables the machine to process it so that it comes to the same conclusions as if it had been processed by a human benefiting from logical reasoning.

4.5.2.3 Ontology Languages

The World Wide Web Consortium (W3C; <http://www.w3.org/>) has developed languages to enable the Semantic Web technology (SW) to be used for developing ontological languages. While research on ontologies within the AI field was first undertaken in the 1990s, the idea of a semantic web [191-193] increased interest in the development of ontologies in the late 1990s.

Ontology has become the backbone of the SW [192, 194-196]. Since then, W3C has ensured that the ontology languages developed are compatible with recent Web standards and layers on top of them. In other words, any such language should be expressed in XML and if possible layered on top of RDF schema (RDFS). Figure 4.3

shows the SW language architecture, which has both syntactic and semantic language layering.

The first (bottom) layer has two components: Unicode and URI. Unicode is a character encoding system designed to create software applications which are able to operate in any language. A uniform resource identifier (URI) is a short string used to identify web resources such as images, services and downloadable files. These two components help the SW to include different languages and all resources.

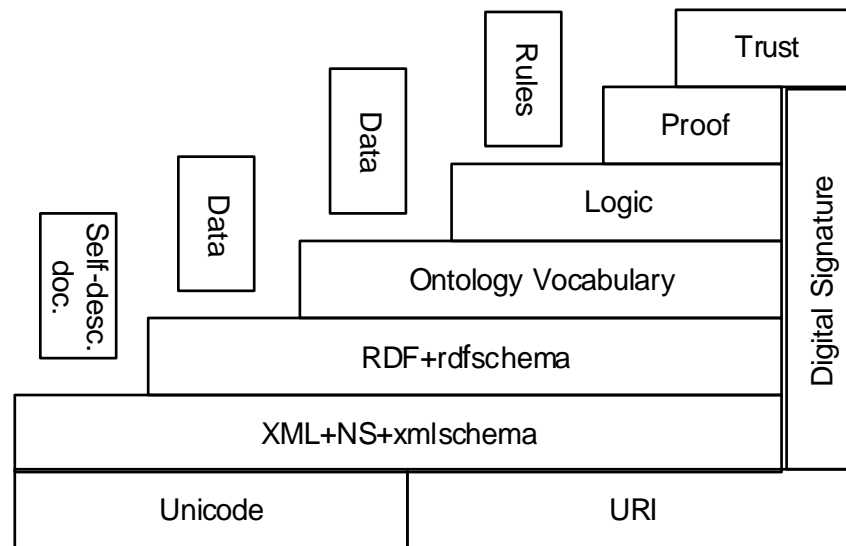


Figure 4.3: Semantic Web Language Architecture [195]

The second layer consists of XML, NS and Xmlschema. XML [197] is a language for describing data using a number of tags with arbitrary names, which can be selected arbitrarily by the creator of the XML document. This implies that an XML document could be understood by a human reader if the names and structure of the tags were selected with care by the author and if the target reader understood the language used. However, this can cause problems, as people have different associations for terms. The

main aim of XML is to provide a technique which can be utilized to mark up and structure documents. This in turn enables machines to identify data in a document by their labels. For example, while some people would claim that the term ‘animal’ comprises humans, others might not accept this classification.

XML document data can be manipulated only in a prescribed way through organizing tags in a particular syntactical structure. XML can be utilized like a data exchange format where all parties participating in the exchange should agree on a general structure for the XML document, or a common XML schema, which is a language used to describe classes of XML documents. In other words, it provides structural prescriptions for XML documents. This means that the XML schema can be used to specify restrictions on what might come between the labels within XML documents, but it cannot be used to identify the meaning of tags, in spite of its feature to build hierarchies of element types. This hierarchy does not comprise conceptual knowledge, but functions only as a syntactic shortcut to enable reuse of complex definitions [192, 195]. NS stands for the namespace of an element, which is the scope in which it is valid. The XML schema keeps the principles of the Standard Generalized Markup Language (SGML) in place.

The third layer includes RDF and rdfschema. RDF is a framework developed in 1999 to represent information on the Web. It annotates the meta-data of web pages, is processed by machine and expresses knowledge in a limited, minimally flexible way. RDF has an abstract syntax which represents an uncomplicated graph-based data model, besides formal semantics with an accurately defined idea of entailment, presenting a basis for well-founded RDF data deductions.

RDFD is a semantic extension of RDF, which provides various essential ontological modelling primitives such as classes and properties [198]. Figure 4.4 shows a hierarchy how can be specified by RDFS. The fundamental construction of any RDF expression can be seen as a directed, labelled graph comprising nodes and labelled directed arcs which connect pairs of nodes.

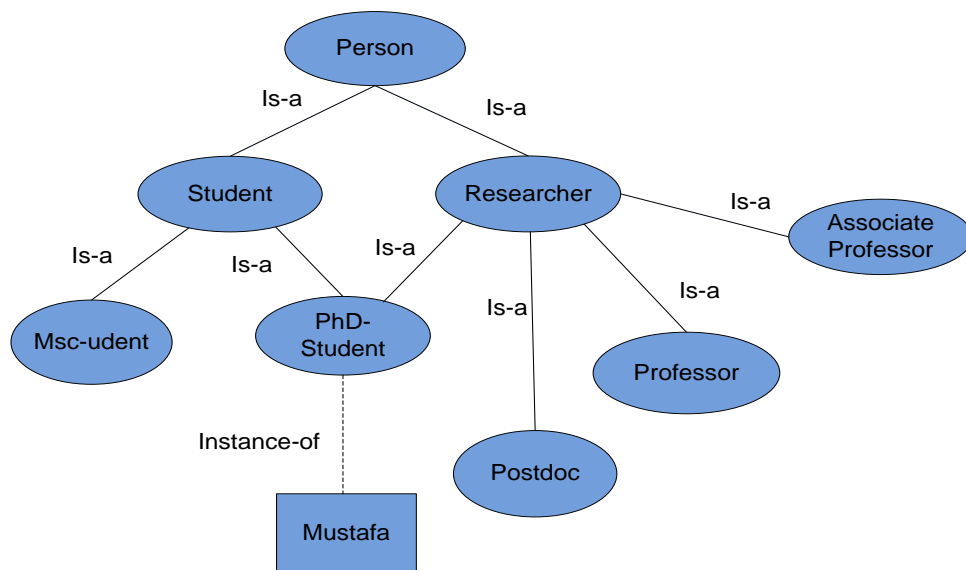


Figure 4.4: An is-a hierarchy (taxonomy) [195]

The RDF consists of a set of triples: Subject-predicate-object.

- The subject is a resource, identified by an URI.
- The predicate is a property name which denotes a relationship and identified by an URI.
- The object is the property value, either another resource or a simple value.
- The literal an identifier of what the node represents.

The fourth layer is the ontology vocabulary, which is used to standardize the means that can be employed in the Web. The third and fourth (RDF and ontology) layers together form the core of the SW language architecture.

The fifth layer is logic, which is a reasoning engine designed to explain the semantic links and infer useful facts about the semantics. The remaining layers in the architecture are trust and proof.

OWL was designed to be employed by applications which need to process information content, rather than merely presenting it to humans. The SW is seen as the future, whereby information accessible on the Web is assigned clear meaning, allowing machines to process and integrate it automatically. SW builds on XML's capacity to identify customized tagging schemes as well as RDF's flexible representation of data. Above RDF, the SW necessitates an ontology language that can formally express the meaning of terminology employed in Web documents. OWL supports greater machine interpretability of Web content than that sustained by RDF, XML or RDFS by offering additional vocabulary, besides a formal semantics to represent information in a way appropriate for applying to ontology.

OWL ontology comprises illustrations of property classes and their instances. With this ontology type, the formal OWL semantics identifies how to represent its logical consequences (facts not accurately apparent) in the ontology, rather than these being determined by the semantics. OWL consists of three expressive sublanguages: OWL Full, OWL DL and OWL Lite.

The ontology and rule-based language together constitute the knowledgebase representation. A rule is a technique for knowledge representation and structure which links one or more premises (conditions or antecedents) or situations to one or more conclusions (consequents) or actions. Researchers have confirmed the need for a Web rule language [112,169,183]. Rules lie above the ontology in the SW language hierarchy (Figure 4.3). A Web rule language is useful to express various types of rules; for example, *standard rules* are used for chaining the properties of ontologies, *bridging rules* for reasoning across domains, *mapping rules* for data integration between Web ontologies, *querying rules* for expressing intricate queries on the Web and *meta-rules* to facilitate ontology engineering. Rule-based languages maintain the standard characteristic of context-aware applications: “When something happens, if some facts are present, then do something”. Therefore, they are used to codify users’ preferences as reaction rules in the form of event-condition-action (ECA) rules.

The structure of an ECA rule is used initially as an application-independent method of expression whose degree of intricacy is based on the intricacy of its atoms (events, conditions and actions). It is also employed as an explanation technique, because the very rule which produces an action provides a valid human clarification of why that particular action was commanded. The context-triggered actions are simple rules, employed to indicate how context-aware systems should adjust as a result of encoding a context which triggers an action. The rationale for using these rules is to govern the system’s proactive behaviour, which can be symbolically expressed to enable users to understand them.

The SWRL[199-201] was developed to represent rule knowledge and is firmly integrated with OWL, as the predicates in SWRL rules may be either OWL-based classes or properties, interoperating between OWL and SWRL not only semantically and syntactically but also inferentially. This means that it is not sufficient to be able to create SWRL rules in OWL which can use OWL ontology's vocabulary; rather, a vital requirement is to conduct reasoning in a semantically consistent way. This further indicates the need to exploit both the rule-based knowledge and the ontology to draw inferences. A combination of OWL and SWRL affords powerful inferential reasoning capabilities [202]. Chapter 6 provides a detailed explanation of SWRL and OWL and the rationale for using them in this research.

4.5.2.4 Ontology Models for Smart Homes

Formal context models need to be developed to provide context representation, as well as semantic interoperability amongst heterogeneous systems [119, 169, 203]. Modelling smart home environments using ontologies is an active research area exemplified by the European Home System (EHS) taxonomy, [169] DomoML [204, 205], pervasive computing modelling [206] and ambient intelligence environments [207].

The EHS taxonomy is defined as a home device classification system developed by the EHS consortium. It consists of the following major classes: *Meter Reading*, which incorporates measurement equipment; *Housekeeping*, which includes household systems and devices, and *Audio and Video* appliances such as telecommunication and multimedia appliances that are capable of establishing interaction [169]. The EHS taxonomy falls short in many respects and provides inefficient house modelling. In particular, it uses an incoherent modelling technique, since overlapping classes are

represented in different branches of the taxonomy. It also fails to deal with the functional representation, abilities and operations of appliances. It merely permits the simple, static depiction of environments, lacking any formal idea of the operating abilities of modelled entities.

DomoML provides a modular ontology for representing home environments. It proposes three main ontologies: DomoML-fun, DomoML-env and DomoML-core [205]. DomoML-fun provides ways of describing each domestic device's functionalities, while DomoML-env simply describes all 'fixed' devices such as doors, furniture elements and walls. DomoML-core supports the connection of devices described by the DomoML-fun and DomoML-env constructs. However, it shows some weaknesses when implemented in real-world domotic systems. Firstly, it mixes various levels of modelling detail, which may lead to over-specification. For example, to define a lamp that can be lit, the modeller needs to describe the lamp including a related switch button, and a single lever. Moreover, it neither tackles and states modelling, nor offers facilities to enquire or auto-complete models. This leads to an awkward modelling effort when a new house needs to be described.

In the ubiquitous computing context, the Standard Ontology for Ubiquitous and Pervasive Applications SOUPA [208] provides a modelling structure which includes vocabularies representing intelligent agents, space, time, user profile, actions, events and policies for privacy and security. SOUPA is organized into a main set of vocabularies which describes the above-mentioned concepts. However, it cannot be directly used to support intelligence and interoperability for home automation systems,

because it lacks several domain-specific concepts, such as suggested modelling of functionalities and devices.

Context-Broker Architecture (CoBrA) is an agent-based context-aware system which enables context-aware services in smart environments. It utilizes OWL to model SOUPA [209] and a privacy policy. In CoBrA, a centralized resource-rich Context Broker manages and maintains the shared context information and controls the devices and agents which contribute to context-aware services in the smart environment. The strongest advantage of CoBrA is that it allows users to identify the privacy policy, thus protecting their privacy [210]. CoBrA employs the OWL SW language to express context ontologies and reasoning about contexts. OWL is used in CoBrA as a sensibly expressive KR language appropriate for identifying many of the common ontologies employed in building intelligent ubiquitous computing applications.

CONtext ONtology (CONON) is an OWL-encoded context ontology developed for modelling context in ubiquitous environments and supporting logic-based reasoning about the context. It provides an upper context ontology which describes general concepts about a context and offers extensibility for including domain-specific ontologies in a hierarchical style [211].

The context model is structured on a group of abstract entities, where each describes a physical or conceptual thing, comprising Activity, Location and Computational Entity (CompEntity). Every entity is linked to its attributes (presented in OWL: DatatypeProperty) and associations with other entities (presented in OWL: ObjectProperty). The built-in OWL property (subClassOf) allows hierarchical

structuring of sub-class entities, thus supporting extensions to include new concepts which are needed in a particular domain [212]. Although it is generic, its concept hierarchy is based on traditional approaches, and several classes are oversimplified and inappropriate for representing domain-specific context knowledge [211].

4.6 Summary

This chapter has reviewed ubiquitous robotics systems and KR technology as the enabling technologies to achieve auto-configuration within smart home environments. Integrating the characteristics of ubiquitous robotics with SW technology was found to be a promising approach to achieving auto-configuration to enable autonomous or semi-autonomous service delivery within such an environment. The main aim is to enable services to utilise any device through any network, seamlessly within the smart environment, anywhere and at any time. It highlights the necessity of ontology-based auto-configuration systems, due to the difficulty of handling rapid changes in the smart home environments without human intervention. Ubiquitous robotics, with computational intelligence embedded both in the robot and in the environment, is fundamental to enabling smart home automation. The ontology-based approach largely supports context awareness and the integration of heterogeneous resource networks [203, 211].

The three revolutionary generations of robotics, namely industrial robotics, service robotics and ubiquitous robotics, have been reviewed. Ubiquitous robotics enables services to utilise any required device through any network within a given smart home

environment, automatically, anywhere and at any time. The difficulty of developing explicit fully ethical robots has been highlighted.

Knowledge representation technologies, particularly ontologies, have also been utilized as a key component of the proposed approach. It has been found to have a profound impact in facilitating the proposed auto-configuration Sobot in order to realise a smart home system. It is particularly used as a framework to describe the various elements of the smart home environment. The ontology facilitates the sharing and reuse of automated knowledge between human and computer agents to achieve semantic interoperability. The coming chapters illustrate the way in which ubiquitous robotics and knowledge representation have been exploited for the purpose of this study.

Chapter 5: Knowledge-based Auto-configuration Sobot Architecture for Smart Home Environments

5.1 Introduction

This chapter describes the detailed characteristics and functionalities of the overall architecture of the proposed approach, named Knowledge-based Auto-configuration Sobot for Smart Home Environments. Auto-configuration is defined as the ability to adjust dynamically to altering circumstances by adapting the given system's own configuration, hence allowing physical or software entities or a combination of both to be added, removed or modified without interrupting the whole service [213]. The auto-configuration Sobot is a key part of Distributed Integrated Care Services and Systems (iCARE) platform, developed in the iCARE project. It is designed as an infrastructure component to facilitate the deployment of iCARE services.

The development of smart home services has begun to be decoupled from that of smart home devices [22, 214]. Rather than being pre-bundled with devices, smart home services have been developed and deployed independently. Manufacturers of smart home devices use different data exchange formats, which are normally not made public. The fact that each device has its features and functions makes it difficult to have heterogeneous devices operating together seamlessly in the smart home network. This hinders the devices in joining or leaving the system without modifying the entire smart home architecture.

The consequence of this trend is a need to link services to appropriate devices efficiently. Due to the increasing number of smart devices and the complexity of smart home services, a major factor deterring the take-off of the smart home concept has been the complexity of installation or deployment. These problems demand standard solutions that benefit from system interoperability. An extensive analysis of the literature identifies auto-configuration as of the main challenges faced by the developers of smart home environments, due to the difficulty of handling variations in the environment without significant human intervention.

Overall, the installation, configuration and integration of large and complex systems into the existing home environment constitute a time-consuming, intricate and error-prone process, even for experts. The proposed approach aims to overcome these challenges.

This chapter is organised as follows: Section 5.2 presents an overview of the iCARE project, whereas the concept of an auto-configuring Sobot within the iCARE

architecture is discussed in Section 5.3. Section 5.4 examines the requirements and specifications of the proposed approach and Section 5.5 illustrates the Belief-Desire-Intention (BDI) model adopted as a suitable approach to build a Sobot system with cognitive capabilities. Section 5.6 details the components of the Sobot solution within iCARE. The chapter ends with a summary and conclusion.

5.2 Overview of the iCARE Project

The iCARE project was designed to accomplish and deliver an integrated distributed home care system. The system architecture has a profound impact on the design and implementation of such a high quality system. This section presents an overview of the iCARE platform, focusing on its fundamental elements and their roles in realizing the iCARE vision. These fundamental functional elements are depicted in the reference architecture of the iCARE system (Figure 5.1). It is a generic layered architecture designed to tackle the issues progressively. The diagrammatic representation of iCARE, showing the various activities and identifying the main focus of the thesis (the configuration Sobot), is highlighted in light green in the harmonisation layer.

The iCARE system focuses on and is driven by user's needs. The care receiver's health status is determined by monitoring a number of parameters which show the interaction amongst the care receivers and their environment. The parameters could be any change in the care receivers' mobility, sleep patterns, etc. which can indicate changes in their functional health status. Based on the detected parameters, the system will automatically provide appropriate health services. This could in turn help to maintain an independent high quality of life and reduce the cost of receiving hospital care for elderly people.

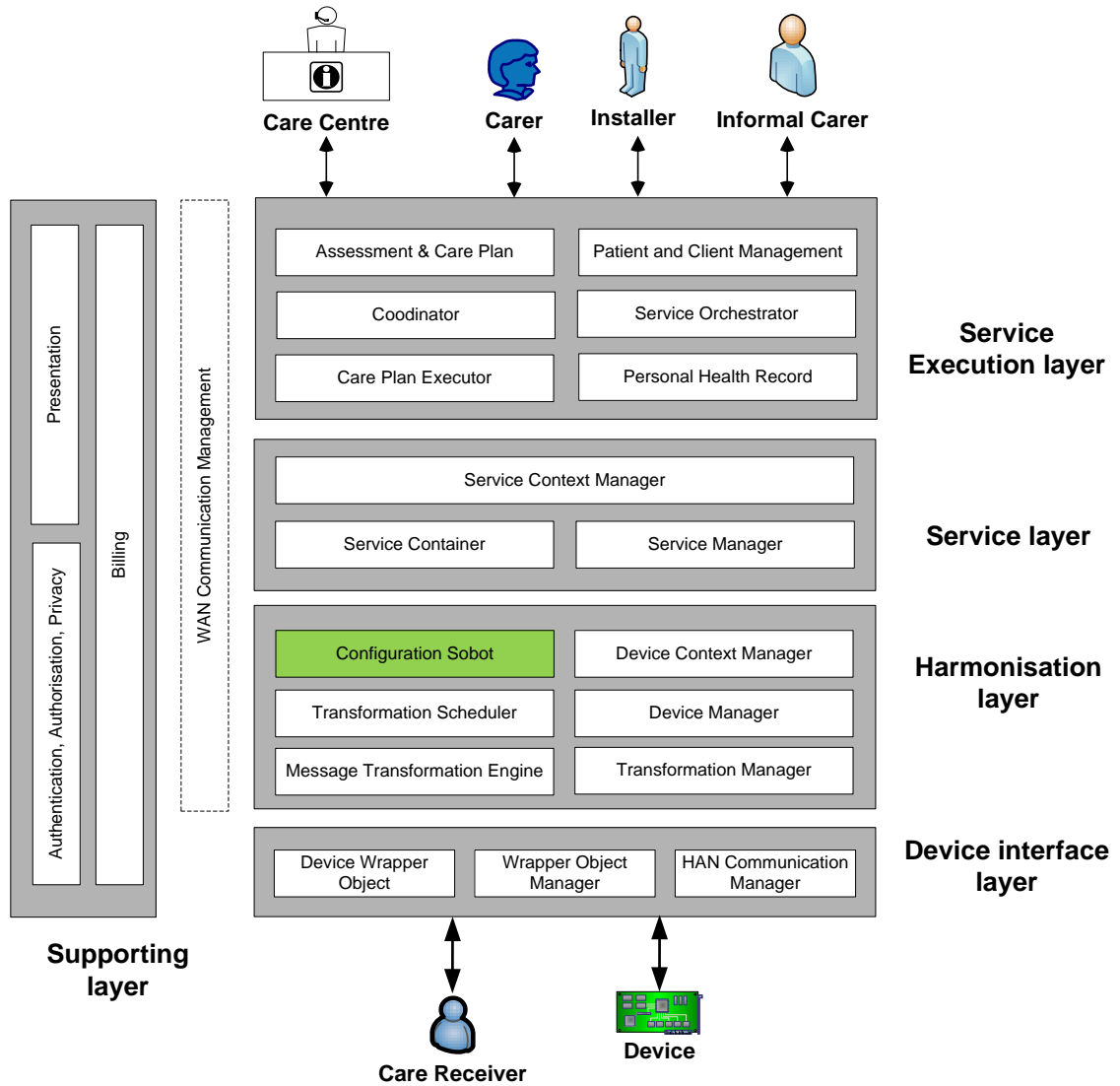


Figure 5.1: Integrated iCARE Reference Architecture

5.2.1 iCARE Actors and Context

Figure 5.1 shows that the integrated care context consists of people and care-related devices, both environmental and medical. Environmental devices monitor and manage the overall quality of the environment, providing care and wellbeing for all care receivers. Medical devices, whether implantable, wearable or embedded in the environment, include sensors to measure blood pressure, heartbeat and body

temperature, and to detect falls, motion and location, among others. They monitor and manage the signs received concerning each care receiver's wellbeing status. The advances in transmission standard technologies for these health monitoring devices (e.g. Zigbee, Bluetooth, RS-232) help to enhance telemonitoring healthcare systems [215]. These message communication standards for medical devices are needed to facilitate the integration of medical devices and systems [214, 215].

The participants are identified as follows:

- **The care receiver** is the focus of the service provision and the recipient of care services. The care receiver's communications are directed to the integrated care platform and can be categorized as the individual's preferences and manually raising the alarm using private devices when needed.
- **Informal carers** are generally the relatives of care receivers. They may react to occasional events, but usually prefer to be informed of the most important events and of progress in handling them.
- **Formal carers** are sources of observational data and can be identified as the major deliverers and contact points of care-related services. They not only provide physical care, but also continually assess the wellbeing and condition of care receivers and identify potential problems. They are also responsible for reacting to alarms, so they must be able to access easily all information related to the care receiver.
- **The care centre** constantly assesses the overall wellbeing of the care receiver and responds by altering the care plan accordingly. It is also responsible, in

emergency situations, for coordinating the actions of formal carers and other emergency response teams and units, e.g. ambulance, police and firefighters. This necessitates that the care centre should be kept informed of the location and actions of formal carers and emergency teams.

- **The installer and maintenance staff** are responsible for maintaining the functionality of devices, installing new devices, upgrading and maintaining existing ones. This requires the staff to be able to access the device resource existing on site.

Based on the above-mentioned human roles, the services can be categorized as follows:

- **Assistive services** are responsible for extending the user's capabilities and helping her/him to accomplish particular tasks. The user triggers the service, which in turn performs the action once. This is deemed a substitution for the user's action. For instance, a user who finds it hard to open a door manually can make use of the control console to do so.
- **Automation services** will automatically execute a function without requiring much intervention from the user, repeatedly executing the action according to sensing data received and fulfilling the user's preferences. For instance, the temperature control service continually monitors the ambient temperature and adjusts it to the user's favoured level. In this case, the user can occasionally modify the preferred temperature.
- **Decision-supporting services** are responsible for a continuous gathering of information in support of human decisions. Unlike automation services,

these do not automatically execute an action, but wait for the user or operator to make a decision and execute the action according to the collected information provided by decision-supporting services. The wellbeing monitoring service will collect data on the user's daily information in support of human decisions. Data fusion, such as combining transitions and activity, provides useful information to the health monitoring service. The decision-supporting services do activities and communicate them to the operator or carer. It may raise an alarm if it detects an irregular status, then the operator or carer will make a decision and execute an action.

5.2.2 iCARE Reference Architecture

This section describes an abstract structure comprising the fundamental components of the home care system design, as represented in Figure 5.1. As mentioned earlier, the reference architecture of the iCARE system is a generic layered one, whose five layers, beginning at the bottom of Figure 5.1, are explained in the following sections:

5.2.2.1 Supporting Layer

The supporting layer is shared by all other layers, facilitates the delivery of care services and deals with the common technical aspects of the system. It consists of these elements:

- **Authentication, authorisation and privacy:** This component addresses security concerns to ensure secure data and information flow through authentication, authorisation and privacy techniques. Authentication refers to techniques to establish and label the identities of individuals; and

authorisation utilizes the identification provided by authentication to implement the access policy and manage individual access to data and information in the system. It also ensures that information and data are encrypted when transmitted between networks. The authentication and authorisation mechanisms help to achieve data privacy, particularly in the case of sensitive data on carers, receivers or clients, based on the privacy policy set by the care centre.

- **Presentation:** This is the general home computer interface to enable users to access the system. It may be either separate client software or accessible via the Web. The user interface may be needed by any of the other layers because of different users' roles. The user interface appropriately adapts to different individuals' needs. This helps to enhance accessibility and good interaction between the system and users with different requirements, skills, abilities and limitations, such as in visual and audio acuity (e.g. touch tablets with Braille labels to be used by blind users) [216-218].
- **The billing component** collects the information needed to calculate the user's bill. It keeps a record of service usage so that the customer can be charged appropriately. The billing information which can be received from the other layers is determined by the business operational model.
- **WAN communication management** is responsible for ensuring uninterrupted communication with the wide area network. It can appear in many layers (e.g. device interface, harmonisation, service), depending on

the physical deployment strategy. It will switch between communication channels according to availability and quality.

5.2.2.2 Device Interface Layer

The device interface layer is meant to resolve problems related to the representation of the device logic in the system. It introduces the care receiver and the medical and environmental devices into the existing system. It integrates the devices into objects and makes provision for services to access relevant devices, as well as allowing the care receiver to access and use the system. The main focus of the system is to make the home network a more dynamic entity which is able to add and remove devices flexibly.

The Home Area Network (HAN) *communication manager* manages various devices and components of the HAN, monitors the availability of devices and handles device registration/ un-registration and network errors within various network domains. Another means of representing devices in the system space, commonly known as the *device wrapper object*, facilitates a quick response to any queries on behalf of devices and gives updates on their status. It also serves to shield particular device communication characteristics, i.e. it translates various specific device communication protocols into the destination high-level protocol. In collaboration with the HAN communication manager, the wrapper object manager manages the lifecycle of device wrapper objects. In detail, it creates a wrapper object when a new device joins its designated network, and destroys the wrapper object once the device leaves the network. The wrapper object manager achieves lifecycle management by utilising the device network protocol. It relies on the device joining network protocol and underlying mechanism to validate the device as complying with the (de facto or international)

standard. By interpreting the network registration information, it can create the right wrapper object for an individual device. Each device wrapper object delegates the accessibility parameters and status of the device to overcome the issue of the device (e.g. a battery powered one) being temporary unreachable.

5.2.2.3 Harmonisation Layer

The harmonisation layer tackles interoperability issues amongst various interconnected devices and services, transforming each device-specific message into a service-explicit message and vice versa. A generic and reusable tool to accomplish such transformations is the already existing *message transformation engine*, which facilitates an efficient message transformation associated with a standard and confirmed solution instead of establishing an individual one. This is achieved by using a transformation script to drive the transformation engine. The *transformation manager* manages the number of engines and transformation scripts available, increasing the number of engines to handle high volume demand and reducing it to release resources. The transformation manager also stores the individual transformation script for each entity according its function, categorises the script and recovers individual transformation scripts when required.

To deal with synchronized transformation problems, the *transformation scheduler* is employed to schedule the transformation process in light of the priority policy. Instead of manually establishing and recording each transformation, the *configuration Sobot* can be utilized to automatically create a transformation channel, depending on the device profile and service profile. It can also remove a transformation channel when it is no longer required, such as when a device or service leaves or enters the system. The *device manager*, which controls the device instances in the system, assists in

discovering accessible devices. The *device context manager* collects and stores context information concerning accessible devices that facilitates the use of services, which means that information such as location can be utilized to characterize a device. It can be employed to realize context-aware services. For instance, the device context manager provides information about the location and motion detection, which in turn allows the service to deduce area occupancy information.

5.2.2.4 Service Layer

The service layer addresses service operation problems with the help of a *service container*, which enables services from diverse provision sources to be freely deployed in the system. The container may be disseminated across different locations to host services, based on the deployment strategy. On the other hand, services can be hosted outside the system and used via the network, in which case the service container may not be used. The *service manager*, in basic terms, deals with the lifecycle of services in the container and monitors the services outside it. It can facilitate an access point to control the various states of a service such as deployment, start, stop and pause. For outdoor services, it provides a central access point for other system components. The *service context manager* gathers and controls information on the service context. It may incorporate service status and the configuration parameters. Such service context information is significant for other services to adjust to changes of service characteristics, as any service can be regarded as a functional component to be used by other services.

5.2.2.5 Service Execution Layer

Finally, the service execution layer comprises the system functions needed to incorporate all available services collectively and accomplish individual care service objectives. The *personal health record* is used to document information on the care receiver's health in a standard format, employed for developing clinical and care plans. It is a complex document, for which a multitude of forms have been suggested: medical images, text-based documents, voice data, machine tracings, etc. The *assessment and care plan* component enables the carer to evaluate the care receiver's conditions and to create and develop a personal care plan. As an iterative process, it enables the carer to perform regular assessments and alter the care plan accordingly.

In order to optimise the carer's productivity, the *care plan executor* bridges the gap between care plan and obtainable care services. It converts the high-level care plan to service implementation and realizes the care plan description. *Patient and client management* deals with information on individual care receivers to ease service delivery. The information concerned may refer to the account (e.g. account name, authentication method), to details of the general practitioner (GP), to the home address or personal contacts. The *service orchestrator* structures a high-level work flow in implementation form to connect certain services together, which in turn represents the intended business model and simplifies the operations.

Finally, as most information management happens within a service team, the *coordinator* coordinates the information flow to ensure consistent service delivery. For example, the care centre will be the first to receive an alarm message and it may react by selecting the closest carer, passing on the alarm message and its related information

to the chosen carer's mobile device securely via authentication and wireless encryption. These two security schemes can ensure protection against unauthorized wireless access to device-transmitted data. It is important to achieve the secure transmission of the required medical data and information to ensure safe and effective use of medical devices. This is particularly vital for medical devices which execute critical life-supporting functions. In dealing with the alarm, the carer may employ the mobile device to access further information (e.g. medical history) and record the actions undertaken. These actions should be automatically reported to the care centre system. It is the coordinator's responsibility to ensure that information flows through the whole system and that the accurate information needed to make a decision is obtainable.

5.3 Service Delivery Schema within the iCARE Architecture

The auto-configuration solution is mainly concerned with the management, configuration and deployment of applications, as well as their integration into the environment. Ubiquitous robotics, which merges ubiquitous computing with robotic technology, offers potential solutions (i.e. auto-configuration Sobot and its powerful features) for smart home auto-configuration, illustrated in Section 4.3. It enables services to utilize any device seamlessly across any network within an intelligent home environment, anywhere and at any time. It automatically delivers services in a cooperative fashion by achieving fast auto-configuration of smart home systems based on a knowledge-based auto-configuration Sobot without user intervention.

The automatic configuration generated by the Sobot is necessary to ease the whole configuration process in the smart home environment. The configuration Sobot executes

three main tasks: checking whether a required operation is supported by the smart home environment; generating a new configuration by performing a series of logistical operations which interconnect the required operations, based on the available knowledge; and providing feedback of the reason for any task failure which occurs if any of these tasks is not completed.

A knowledgebase is used to describe the context of the smart home environments. It allows the Sobot to select automatically the resources most appropriate to the context of the applications, via semantic searching and matching. This automatic selection will reduce the amount of data exchanged and the number of operations needed during an application configuration.

The development from a typical iCARE manual configuration process to an automatic one is illustrated in Figure 5.2. Device manufacturers supply devices and service suppliers provide services. In the typical manual configuration process, device and service manuals are also provided to explain their installation procedures and functionalities. The system installers install the devices and deploy the services according to these manuals and their experience. Their knowledge and experience also help them to link the devices logically to the services. The sensors are simply placed by the installers in the ‘normal’ locations. However, it is not enough merely to place sensors in a room; they must be in the right place so that their functionalities are not compromised. Determining the right place to install them requires a range of factors, of which height of the occupants is one, to be taken into account. This helps to make all the occupants clearly visible to the sensors in all places in the target rooms.

However, this kind of typical practice is problematic, as any wrong interpretation of these manuals will lead to service failure, placing a major burden on the installer's shoulders. The need for experience and knowledge may exceed the capability of a normal household. Additionally, since knowledge and experience of the system configuration are held by an individual installer, it will not be very easy to share or transfer them with others.

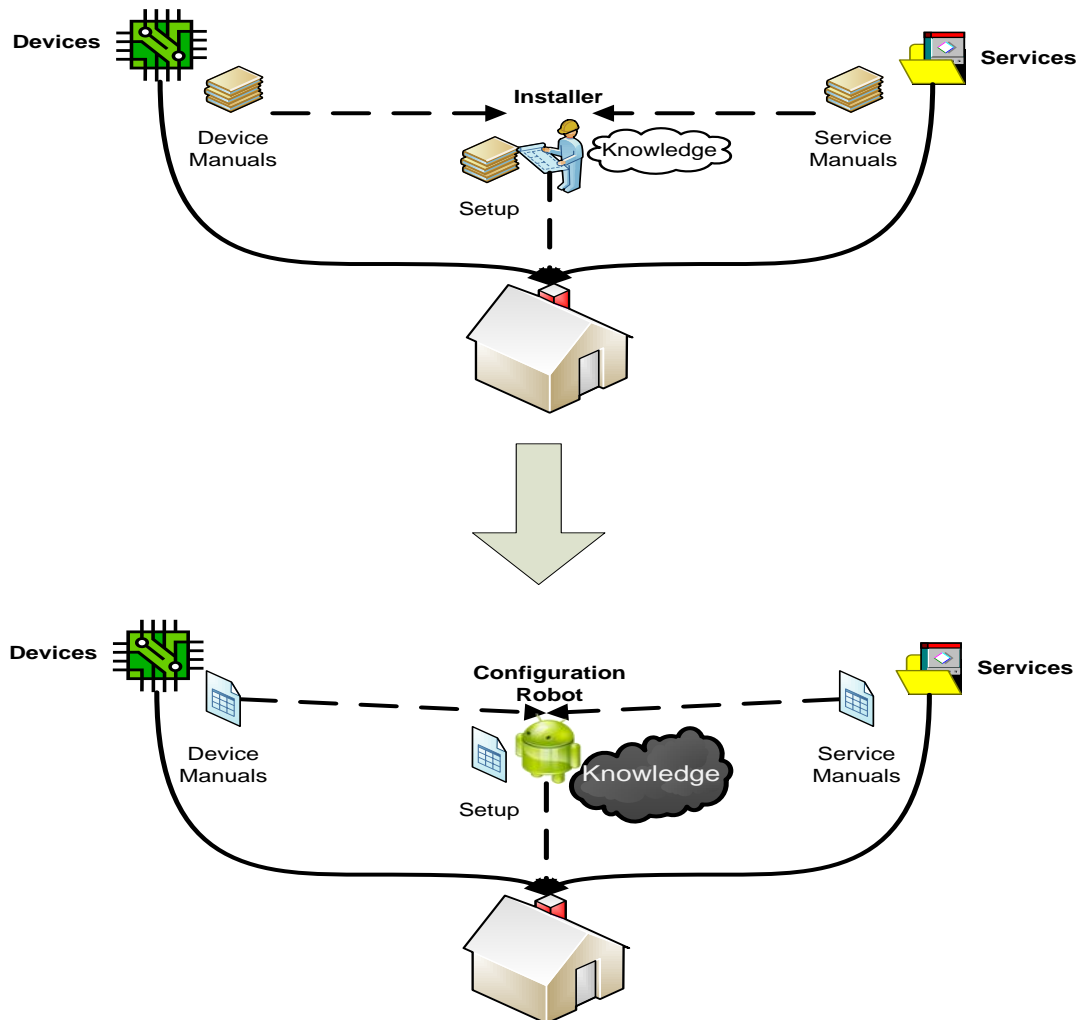


Figure 5.2: iCARE Configuration Process Architecture

The proposed structure supports the sharing of information across a wide range of installations so that lessons learnt from one can be readily incorporated into other practices. Therefore, automating the configuration activities executed by an auto-configuration Sobot is a new approach to overcoming the configuration challenge.

The advance of knowledge-based technology in mimicking the activities of an installer makes the approach feasible. The essential part of the approach is to describe the knowledge formally, which allows the machine to understand and process it within the system. This knowledge includes the formal description of each device by its manufacturer, the formal description of each service by its supplier, the formal description of the setup by the installer and the knowledge needed to create the logical link, also held by the installer.

5.4 Requirements and Specifications of the Proposed Approach

A good understanding of the configuration of the linkage between smart home services and devices, as well as the related requirements and specifications, is considered to be vital for designing a successful solution strategy. The proposed approach should obviously identify the corresponding requirements and specifications to ensure their accuracy and completeness in meeting the smart home's needs.

The auto-configuration system can simplify the complexity and reduce the cost of the management and employment of applications, services and devices in a smart home environment [219]. Techniques inspired by autonomic computing and ubiquitous robotics can be utilized to automate the heterogeneous resources of the home environment. This section discusses autonomic ubiquitous robotics and in particular the

auto-configuration Sobot system. The majority of smart home environments is composed of various resources: personal computers, embedded devices, sensors, smart phones, etc. In light of the ubiquitous computing model suggested by Weiser [54], the environmental information processes are distributed amongst the resources.

The proposed model is very much related to the smart home environment in terms of the relationships between them. In other words, several resources communicate information in a dynamic environment where they quickly enter and leave the system, etc. With auto-configuration, it is likely to deploy many applications automatically. Given its major features of seamless and calm interaction with the environment, context-aware intelligence and context-aware self-learning, an intelligent Sobot is capable of performing a multitude of tasks [132]. By virtue of its intelligence, a Sobot can be aware of the surroundings by itself and interact with the other components of the system. It can also operate as an individual autonomous entity, selecting an appropriate activity according to its own internal state with no help from others.

The auto-configuration Sobot is responsible for forming a new configuration subsequent to its analysis of the latest requirements and for inferring new knowledge about the environment via its inference engine utilizing a user defined rules. The inferred new knowledge allows the smart home system to adapt to these requirements accordingly. It creates a new or alternative feasible configuration which meets the system's requirements and includes a sequence of available smart home activities. The Sobot decides on the new configuration of the smart home, using facts inferred from the ontological model, the user's perceptions and wishes, information abstracted from the

project specification and requirements (see Section 6.5.3). If the Sobot detects any conflict, it will notify the user, who will resolve the conflict manually.

The proposed Sobot is different from traditional applications in terms of adding self-awareness of the dynamic environment. Reliability and flexibility are both pivotal requirements for the proposed approach, because any mistake or failure in the operation of smart home services may prove prohibitively costly. The complexity of the smart home environment is solved in the proposed model by intelligent techniques based on semantic descriptions or a set of rules governing the application. Overall, the approach should be reliable and flexible enough to satisfy the requirements of smart home environments, ease of use and minimum user intervention, and to enhance the healthcare monitoring services.

The Sobot requirements and specifications are as follows:

- **Awareness.** The Sobot needs to be aware of its digital and physical environment. It must sense environmental changes directly or via the iCARE platform. These include devices or services joining or leaving the environment, and the context of changes to devices and services.
- **Reasoning.** The Sobot needs to be able to interpret these changes, extract the relevant contextual data, and then encode these data into appropriate formats. By utilising available knowledge locally or remotely, the Sobot should modify the configuration plan, based on the new information. Eventually, it needs to generate an auto-configuration plan to adapt to current system conditions.

- **Validation.** The Sobot needs to validate any new information or knowledge. It should be able to avoid contaminating the knowledgebase by isolating and eliminating any inconsistent information.
- **Continuous Service.** The Sobot needs to be able to execute the generated auto-configuration plan without interrupting the running of the system.

5.5 The Belief-Desire-Intention (BDI) Model

Based on this review, the Sobot's major features (automatic interaction with the environment and its components, context-aware intelligence and context-aware self-learning) mean that the proposed approach can be considered suitable to facilitate the auto-configuration of smart home services (discussed in Section 4.3). A Sobot can learn from experience, which in turn will be shared across a group of installations. Thus, it is useful to model a computing entity as a Sobot when creating intelligent ubiquitous computing systems [220]. Sobot is a powerful software agent which mimics an autonomous and adaptable agent in nature and is able to migrate and take control of the other two types of Ubibot[221].

This study has adopted the belief-desire-intention (BDI) architecture as a suitable approach to build Sobot systems with cognitive capabilities [222, 223]. BDI architecture is extensively employed in AI research [224-226]. The BDI model is not only a well-known approach for the execution of the Sobot's practical reasoning techniques [227-229], but also highly appropriate for dynamic environments, because it offers a higher performance than task-oriented systems in uncertain and dynamic environments, which requires automatic recovery from erroneous situations [230].

BDI software model was developed for programming deliberative Sobot using symbolic representations of their beliefs, desires and intentions to reach given goals [231, 232]. A BDI Sobot has goals relating to the anticipated target smart home system state (desires) and acquires knowledge of itself and the surrounding smart home environment (beliefs), based on which it makes decisions to respond to changes based on plans [231]. These goals can be achieved by executing particular tasks (called intentions) utilizing actuators in the virtual and real worlds [233]. These tasks are generally predefined by the software developer and continually reassessed to establish their efficient advancement of the system towards the target state [234, 235]. If not sufficiently advanced, it may re-examine its decisions, eliminate suboptimal tasks and start new encouraging ones instead [224].

This further implies that the BDI architecture facilitates the behaviour of the goal-driven system. The focus of the BDI approach is on reasoning in a resource-bounded agent [236]. Resource bounds arise from the potential limits to agents' memory or computational power and their ability to gather information on the environment. This means that agent reasoning takes time. Moreover, such agents act in dynamic environments, which may change while the agent is reasoning. The agent cannot stop the environment changing around it while it continues reasoning. It may also be slow in deciding on a plan, so that by the time it has finished, the environment may have altered sufficiently to invalidate the plan [237]. Each time the agent recognized a new fact, it would have to begin planning from scratch, which would mean that it spent all its time planning and re-planning, thus never actually doing anything.

This BDI architecture has been adopted as a suitable approach to build Sobot systems with cognitive capabilities. The BDI concept has been incorporated with semantics through its logical semantic-based programming languages. The Sobot's beliefs, which are stored in the Sobot's belief base, are entirely based on semantic data. New information acquired from the environment and delivered to the Sobot is also stored in the knowledgebase, which is automatically and hence non-deterministic enriched, using description logic (DL) reasoning. Furthermore, semantic reasoning is executed to generate the new knowledge necessary for the actions required to achieve goals, based on the semantic beliefs. The DL reasoner [238] supports consistency checking amongst all classes and relationships in the smart home environment and classification (subsumption) or instance checking, which is concerned with inferring a new hierarchy from the stated definitions [37].

Subsequently, agents can retrieve the outcomes of reasoning through these beliefs. This provides vital support in working towards the targeted interoperability within the smart home environment system. This BDI Sobot-based technique has been proposed to facilitate the dynamic reconfigurable testing of service systems [239]. Its purpose is to benefit from the Sobot's collaborative ability, autonomy and reactivity.

Thus, the auto-configuration framework whose design is presented in this thesis was inspired by the BDI model for the employment of the Sobot's reasoning mechanism. The rationale for this is that it is recognized not only as one of the most effective approaches to reasoning, but also as appropriate for dynamic smart home systems. Figure 5.3 shows how the structure of the Sobot based on the BDI model.

The following is a more detailed explanation of the application of the BDI model to the Sobot. First, *beliefs* represent the Sobot's recent factual knowledge of the environment in relation to an internal world model. This model needs to be updated frequently by measurements elicited from sensors embedded in the environment, caching the existing world model state and storing continuous information such as the attributes of applications, users' preferences and the Sobot's properties.

Desires are the Sobot's goals related to a preferred target system state. They describe high-level concepts in the Ubibot's brain, serving to adjust its activities to attain goals as rapidly as possible. In pursuit of these goals, the Sobot performs tasks or intentions utilizing actuators in the environment. *Intentions* stand for the Sobot's commitment to its desires (goals) and the plans designated to attain them. Intentions must be consistent and cannot contradict each other.

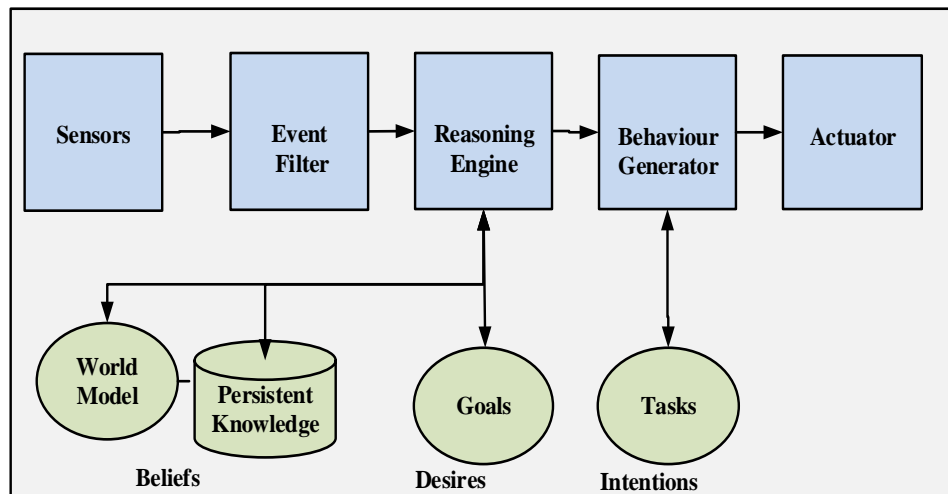


Figure 5.3: The BDI-based Structure of Sobot [230].

The tasks presently being carried out are continually re-examined to validate the efficient progress of the system towards the target state [230].

Above all, it must be able to execute the auto-configuration plan it has generated without interrupting the running of the smart home system. A detailed description of the conceptual solution of the configuration Sobot, specifications and functions is provided in the following section.

5.6 Configuration Sobot Conceptual Solution

As shown in Figure 5.1, the Sobot which is the focus of this thesis is a key component in the iCARE project. The Sobot is responsible for linking devices and services automatically. The sharing of discovered knowledge is functionally embedded in the Sobot. The figure shows the Sobot and its components as an integral part of the iCARE platform.

The architecture of the Sobot has the following four interdependent components, shown in the large shaded box in Figure 5.4: a *resource handler*, a *local knowledgebase*, a *configuration plan generation engine* and a *configuration plan executor*.

The *device manager* manages all devices which join the iCARE ecosystem and monitors their status, while the *service manager* does the same for all services within the iCARE ecosystem. The *XML message bus* is responsible for interlinking the devices and services to deliver the required services. It can be auto-configured via API. The *centre knowledgebase* stores the stable, refined and validated knowledge shared across

the iCARE ecosystem. The following sections illustrate the functions of Sobot and its components which will be further elaborated in Chapters 7 and 8.

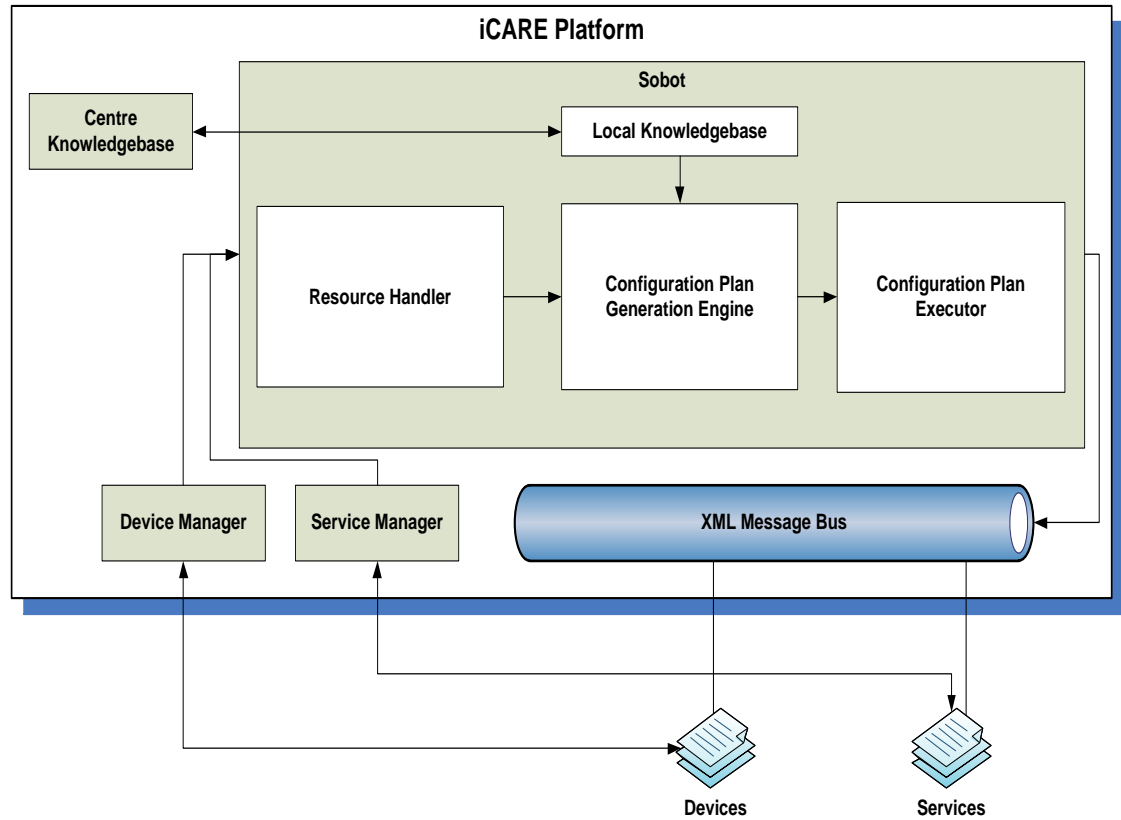


Figure 5.4: Sobot Architecture

5.6.1 The Resource Handler Component

The resource handler is the input of the Sobot, monitoring the status of devices, services and context. It generates the configuration of relevant events and encodes the relevant information. The resource handler provides knowledge regarding the requirements and specifications of the iCARE resources (services, devices and context) needed for decision making. While the specification and requirements are presented in XML files, environmental knowledge is presented in a standard file.

5.6.2 The Local Knowledgebase

The local knowledgebase localises the central knowledgebase to an individual or special environment by extracting common or shared knowledge from the central knowledgebase. The knowledge model component extracts explicit information, which is then analysed and modelled to make it easy to access and manipulate.

Based on the contribution of knowledge, the knowledge model is divided into four models: device model, service model, configuration model and context model, as shown in Figure 5.5.

The device and service models are provided to formally describe the devices and service entities. The context model is used by installers to describe the physical environment so that the devices and services to be deployed can be specified. The configuration model formally encodes the practice and experience of installers to create logical connections between devices and services. The sub-models will be further explained in Chapter 6.

5.6.2.1 Context Model

The purpose of the context model is to describe the physical environment characterising the operating environment of the devices and services. The context model includes two models to describe two physical systems: a location model and a physical device model. The location model is intended to model various parts of the home in order to specify physical entities related to location, segment physical areas, identify individual physical areas and describe the relations between them.

The physical device model is designed to model devices that cannot be directly interacted with via digital messages, but of which the digital system needs to be aware.

The status of these devices can be monitored only via add-on sensors and they can be controlled only via add-on actuators. The physical device model will be further discussed in Section 6.4.2.1.

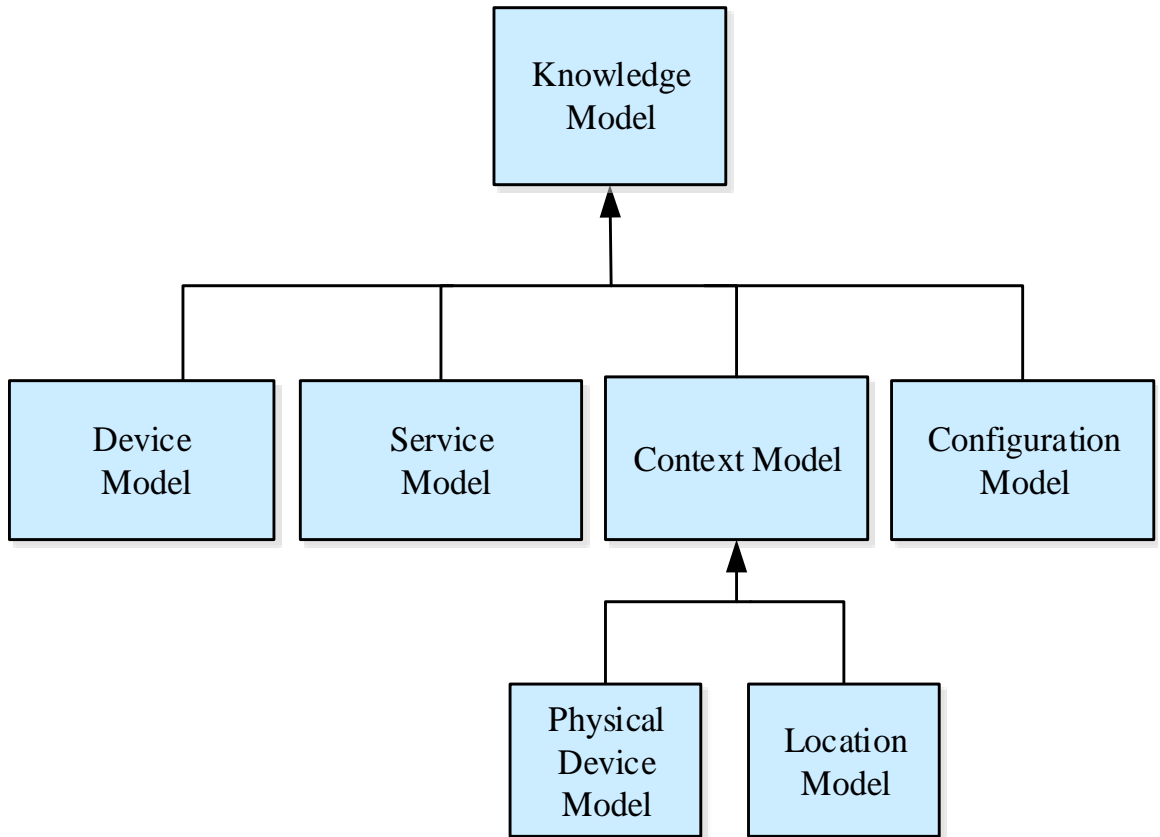


Figure 5.5: Knowledge Package Model

5.6.2.2 Device Model

The aim of the device model is to impose a proper structure through which one can model a device across various domains in a consistent fashion. Unlike the physical device model, devices within this model can interact actively with a digital ecosystem by generating and/or receiving digital messages. The digital ecosystem is a distributed, adaptive, networked architecture which helps to optimize control decisions and facilitate

information obtainability and accessibility [20, 21, 23]. Thus, to achieve highly precise automation within several domains, shared formal semantics should be added. The device model focuses on specifying a general structure to present the meaning and intention of messages. In this research, the device entity is specified as the top artefact of the device model, which represents an independent physical entity within the digital environment. Thus, it is regarded as host to the attributes and functions related to the specific device, which will be discussed later in Section 6.5.2.2.

5.6.2.3 Service Model

The service model is a model of the device (or resource) artefacts from an application perspective, so the service can be connected to and utilise the correct resources. The service model introduces a new concept by deriving it from the device concept. The main method of derivation can be summarised as refinement and reclassification. For refinement, IndoorTemperature is defined as a temperature measurement and location is IndoorArea. For re-classification, FireAlarm is defined as SmokeAlarm or COAlarm or GasAlarm. The service entity is specified as the top artefact of service model, which has URL address, and a parameter. A further illustration of the service model is provided in Section 6.5.2.3.

5.6.2.4 Configuration Model

Rather than introducing new concepts, the configuration model mimics the practice and experience of installers to create logical connections between devices and services in the form of rules. It is basically utilized to connect the smart home services with the available devices through semantic searching and matching. For example, it automatically creates a link between temperature sensor and heating control service

inferring that the temperature sensor measures the outdoor temperature, so the outdoor temperature sensor should be connected to the heating control service. The outdoor temperature sensor is used to achieve the early start and early stop home temperature control. The early start and stop energy conservation control strategy uses three variables—active occupancy, preferred internal temperature and external temperature—to reduce space heating operation time (discussed in detail in Section 8.3). Further elaboration of the configuration model and this example is provided in the following Chapters.

5.6.3 Configuration Plan Generation Engine Component

The Sobot has an inference engine that derives recommendations from the knowledgebase models. By utilising the local knowledgebase, the configuration plan generation engine generates a valid configuration plan. A piece of software called the *reasoner* is used to infer logical consequences from a set of declared facts or axioms. The reasoning (inference) component of the decision engine consists of inferred rules and a forward-chaining rule decision engine for performing intelligent reasoning, which in turn generates a final configuration. The decision engine generates a feasible configuration that meets the given requirements using the smart-home project specification, while the knowledge-based model uses the Pellet resources configuration in the project. Reasoning mechanisms are required for checking and validating the consistency of the knowledge model, besides checking and testing the related rules for understanding the context.

The reasoning process is executed in two steps: the first one is ontology-based reasoning which is utilized to check the consistency amongst classes and relationships

in the context. The second one is user-defined rule-based reasoning (discussed in detail in Section 6.5.3) which employs automatic first-order logic to help the Sobot to infer new knowledge using a built-in reasoner engine (Pellet) in Protégé software. In other words, high-level implicit contextual data cannot be immediately obtained from sensors, but is instead inferred from sensor-driven, low-level explicit contextual data, such as environmental information and physical location, for knowledge reasoning.

5.6.4 Configuration Plan Executor

The configuration plan executor will automatically execute the generated configuration plan to ensure that the system is configured without interrupting normal operation, based on the knowledge models and the inference rules specified in the reasoning engine. The result is an auto-configuration of the devices and /or the services leaving or joining the smart home system. Its functional units together provide the mechanisms that control the implementation process of the configuration plan, resulting in dynamic updates. Further elaboration is provided in Section 7.4.4.

5.7 Summary

This chapter has presented a detailed description of the proposed ubiquitous robot system, namely the Knowledge-based Auto-configuration System for automating smart home services. It has presented the specifications of the proposed approach required for it to be applicable as an infrastructure for smart home services. It has also discussed the general architecture of the system and the proposed configuring Sobot, including the resource handler, the local knowledgebase, the configuration plan generation engine and the configuration executor, generating a feasible configuration that meets the

requirements of the smart home project specification. This high-level conceptual description will be discussed in more detail in chapters 6 and 7. Specifically, these chapters will clarify the means of building the knowledgebase model and generating the inference rules, as well as the language used. The development platform for ontology modelling and knowledge acquisition will also be presented.

Chapter 6: Knowledgebase Model for Auto-configuration Sobot within a Smart Home Environment

6.1 Introduction

This chapter presents the knowledgebase model created to enable the Sobot to achieve auto-configuration, in order to realise the vision of a smart home system environment. To achieve auto-configuration of a smart home system, it is necessary to solve the problem of how to select particular components automatically for specific operations. In order to do this, very significant issues in the field of auto-configuration are the semantic web language, which should be expressive enough to formally describe the relevant smart home components, their functionalities and relationships, and to facilitate seamless automatic configuration; the technique adopted to generate interaction among

these components dynamically; and the criteria chosen for selecting automatic software components. This interaction is required to accomplish the given system's goals, based on the descriptions of these components.

This chapter begins with an explanation of ontology-based knowledge representation in Section 6.2, then sections 6.3 and 6.4 present the ontology modelling method adopted and the smart home ontology design for the proposed auto-configuration Sobot. Section 6.5 explains the design and building of the knowledgebase model using Protégé and the chapter ends with a summary.

6.2 Ontology-based Knowledge Representation

The knowledgebase is used as a framework to describe the various elements of the smart home environment. It plays a significant role in helping the Sobot to select automatically the most appropriate resources to meet the demands of the applications via semantic searching and matching. To achieve this, modelling the environment helps to understand the smart home resources and their relations, facilitating the Sobot's role in selecting those classes which are essential to accomplish auto-configuration. The knowledgebase for the auto-configuration Sobot is modelled by building an ontology of the smart home's concepts, sub-concepts and instances (individuals), including their relations, and the creation of a number of rules and facts.

As mentioned in Chapter 4, ontology is "a formal explicit specification of a shared conceptualization of a domain" [240]. It is generally expressed in a logic-based language to define precise and meaningfully distinguished classes, their properties and

relations. It consists of machine-readable definitions of the main concepts in a particular domain of knowledge, their properties and relations. Thus, the creation of an ontology is a vital step in devising an efficient knowledge representation system, and vocabulary is required to perform an efficient ontological analysis of the domain. The reasons for using ontology are to share a common understanding of the information structure amongst software agents and people, to facilitate reuse of domain knowledge, to clarify the domain assumptions and analyse its knowledge.

The ontology knowledge model of the resources in the target smart home environment is described using OWL DL, developed by W3C. The interoperability of OWL, which is used for defining web ontologies, is attained by using the standardized eXtensible Markup Language (XML). OWL supports knowledge reuse and sharing, which are vital for adding new facts to the designed model and updating it. OWL DL, a dialect of OWL, is employed here to express the proposed knowledge model, in order to ensure that reasoning will be computable as well as decidable.

Protégé' [37], which is utilized for modelling the knowledgebase, has two components: the terminological box **TBox** (formal specification) and the assertional box **ABox** (instances of resources that conform to **TBox**) as shown in Figure 6.1. Each resource has an OWL document with instances (individuals) of all the classes of the resources. To support the addition of new resources to the smart home environment, the present model needs to introduce more particular sub-concepts for each resource, their associated relationships and their related instances of the objects.

The ontology document comprises of **TBox** only, while the objects' OWL documents comprise of both parts: **TBox** and **ABox**.

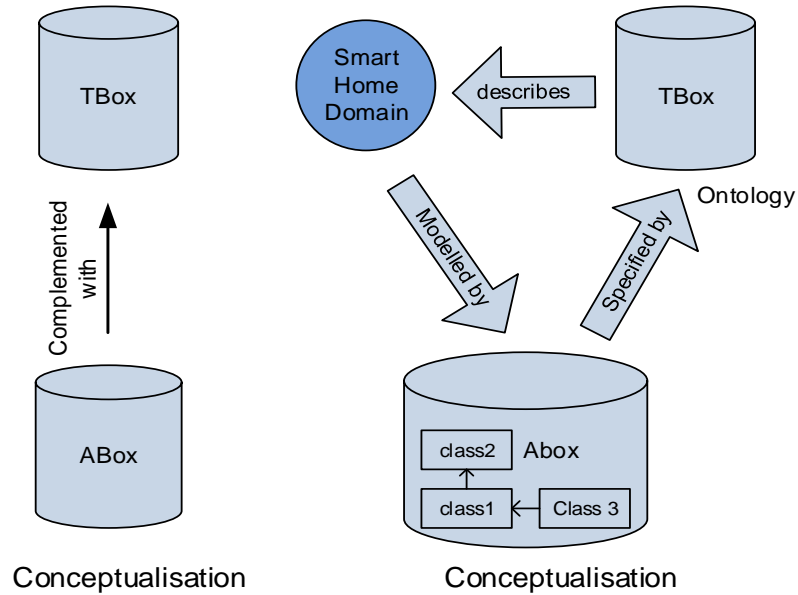


Figure 6.1: Knowledgebase of the Smart Home Domain [30].

To build the ontology model, Protégé 4.1 successfully served as ontology design environment [35]. It is an ontology editor as well as a knowledgebase framework. Ontology within Protégé can be displayed in various formats: OWL, RDF(S) and XML Schema. It provides a plug-and-play (PNP) environment which helps in quick prototyping and software application development.

In Protégé, OWL properties signify the relationships between two instances (individuals). The three main types are *object*, *data* and *annotation* properties. An object property describes the relationships between two different classes. These relationships are needed to represent appropriately the way various entities in the iCARE environment collaborate to model a situation that holds in the target environment.

Object properties connect individuals from the domain to individuals from the range. As to the second type, every class has its own features to specify facts of the class and/or to manage the individuals generated from the class. Data properties connect an individual to Resource Description Format (RDF) literal or an XML Schema data value. The third type of property of OWL is the annotation property, which can be employed to add useful information to classes and individuals, such as labels, comments, author creation date or references [241].

6.3 Ontology Modelling Method

This section describes the ontology modelling method adopted in this thesis. The present ontology is modelled following the stages enumerated by [241, 242], as a method specifically designed for domain experts rather than ontology engineering.

The general stages of designing and developing the ontology are shown in Figure 6.2 and the details are in the following stages:

The first stage of developing an ontology begins by defining its source and domain, then generally extends to defining its scope and purpose. The purpose of the ontology is to guide the design process by acting as a domain conceptualization [241]. This is achieved by answering the following questions [243], as illustrated in Section 6.4:

- Which domain does the ontology cover?
- What is the purpose of the ontology?

- What are the questions that can be answered via the information provided in the ontology?

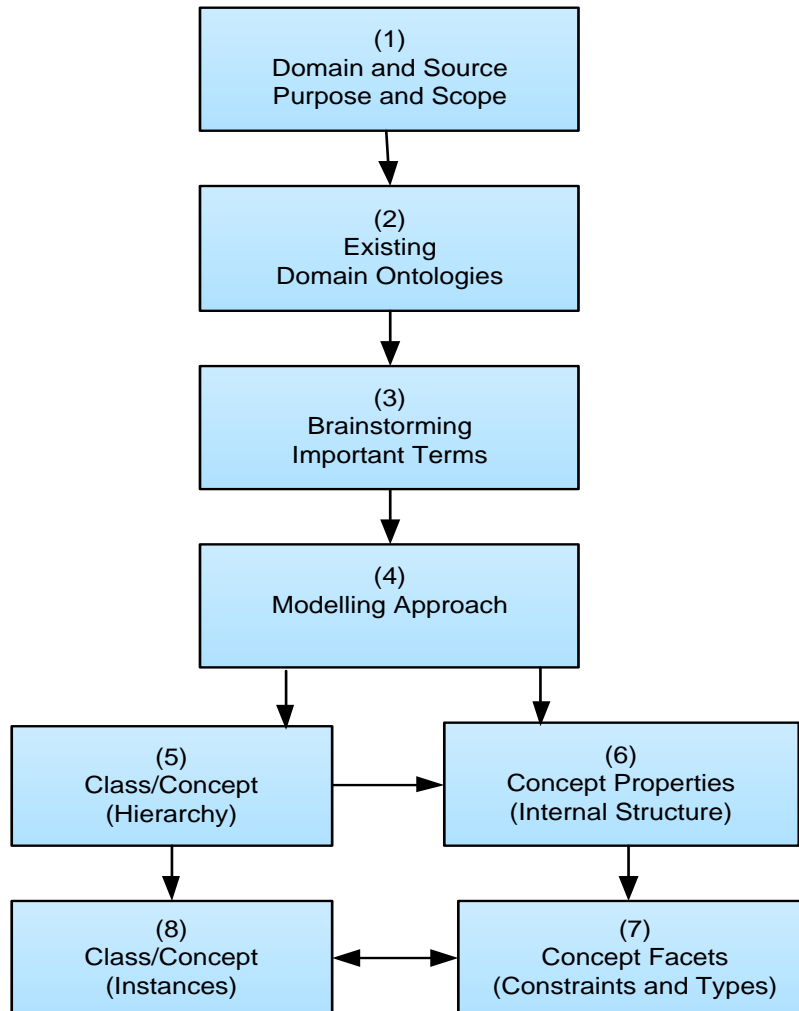


Figure 6.2: Ontology Development Process [241]

The second stage entails checking the ontologies already developed in a similar area. It is easier to manipulate an existing ontology to suit one's needs and requirements than to generate a new one from scratch.

The third stage is brainstorming, to identify and enumerate the most important ontological concepts and phrases in the given domain. Creating a taxonomy of smart home components is a way of classifying a set of concepts, utilizing a hierarchical structure.

The fourth stage involves identifying a suitable approach to modelling the ontology: top-down, middle-out or bottom-up. The top-down method starts by identifying the domain's most general concepts, then more specialized ones, whereas the bottom-up approach commences by defining the most detailed classes, then groups them into more general concepts. The middle-out approach starts by defining the most essential terms in each area before moving on to more specific and abstract ones. The selection of a particular approach should take into consideration the following factors:

- *The bottom-up approach* requires more effort; it makes it difficult to identify commonality amongst related concepts, because such an ontology results in a very high level of detailed concepts, which requires more effort. It also raises the risk of inconsistencies, leading in turn to more effort being required.
- *The top-down approach* results in better control of the level of detail, although it can lead to arbitrary high level classifications being imposed. This can make the model less stable, so that it requires more effort and rework. It also leads to commonalities between the interlinked concepts being missed, because it focuses on dividing concepts instead of putting them together.

- The *middle-out approach* is the preferred method adopted in this research, because it strikes a balance in the level of detail. This means that detail arises only when needed, by identifying the fundamental concepts. This implies that some unnecessary effort is avoided.

The fifth and sixth stages are closely interlinked. The arrow between them is unidirectional, because once the class hierarchy is completed; it becomes possible to identify its concepts' properties. The classes (concepts) and their hierarchy are identified at the fifth stage, while their properties are identified at the sixth stage. The class hierarchy can again be determined by using the top-down, middle-out or bottom-up approaches. All terms listed at stage three which have an independent existence should be extracted into the various classes (concepts) of the ontology. Determining its hierarchical organization involves asking if each of the instances in a class could also be an instance of a more general class. If so, then the former class becomes a subclass of the latter and drifts further away from the ontology's root concept. After defining all the classes, the internal structures (properties) of the concepts need to be described. Yet again, these properties should be easily available from the list generated at the third stage.

At the seventh stage, facets are attached to the properties. This means describing the value type (data property), the allowed values, their number (cardinality) and other features which are considered essential. Consequently, constraints are placed on the types of data allowed. The constraint relationship is employed to represent a restriction on operations which may be executed which in turn helps to limit the number of links

between instances (individuals) of the entities or relationships which are permitted [244].

The final stage involves creating individuals in the classes, which means providing examples of each of the classes.

Ontology is used for representing knowledge, not only in the form of concepts, sub-concepts and relations hierarchies but also involving a number of “IF ... THEN ...” rules and facts [245]. Decisions can be based on the detection of situations and events occurring in specific contexts. This information acquired in relation to a smart home is stored in an active database. In this case, ECA rules are invoked in response to incoming information. Ontology offers rules to support performing many tasks such as detecting information inconsistency and creating new knowledge as new activities for various applications and new relations among concepts, activities and contexts, besides creating applications in a hierarchical way. The knowledgebase consists of the identified smart home concepts and their relations represented in OWL, and from rule knowledge and facts encoded using SWRL, which is basically a combination of OWL and Rule Markup Language (RuleML).

6.4 Smart Home Ontology Design

The following are the stages through which the smart home environment was modelled, in light of the method presented in Section 6.3.

6.4.1 Domain, Purpose, Source and Scope

The main purpose of this work was to develop an ontology to facilitate the auto-configuration of a smart home environment; including the modelling of service and device functionalities and other related concepts. As to the scope of the work, the newly generated ontology is customized for the smart home domain, in particular an auto-configuring system within the iCARE project. The smart home was chosen as the domain of the present ontology, which was newly developed using the well-known Smarthome website (<http://www.smarthome.com>) as its main source, belonging to a large retailer of thousands of popular and affordable home automation and smart home products.

6.4.2 Existing Domain Ontologies

Recently available classifications from service providers and manufacturers of smart home devices and their ontologies are not compatible with the existing standards [245-246]. The problem of flexible ontology modelling, where modellers from different domains represent the same entities and concepts in the real world using their own differing terminologies to describe these concepts and their relationships, results in limited knowledge sharing or no interoperability of vocabulary.

This arises from the syntactical and semantic attributes mismatch among the smart home ontology concepts and their relationships [246]. Syntactic attributes represent the structure of a service operation, e.g. the input and output parameters which describe the operation's messages. The semantic attributes define features, the approach and constraints under which the operation is executed [247]. Thus, adding axioms to the

designed ontology could help narrow the selection criteria and discover any semantically mismatched items [248].

Due to the impossibility of identifying in advance the types of relationship which would later be required, the list of relationships was developed in parallel with the development of the ontology. The smart home environment is the source of the concepts in this ontology. The present ontology was modelled from scratch to address the challenges of interoperability in the target domain. The proposed ontology conforms to the well accepted standards of ontology interoperability [249]. In general, the focus of these ontologies is to provide semantic interoperability amongst heterogeneous components in the contexts and most of the automation actions supported by such ontologies are not terminological uniformity-aware [250]. The fundamental problem of these ontologies is that their implementation is driven by their particular needs, with no regard to knowledge reuse and sharing with other smart home information systems.

The knowledgebase model was a key component in developing the auto-configuration Sobot for intelligent home systems. It is used as a framework to describe the various elements of the smart home environment. As shown in Figure 6.3, the knowledge model is divided into four modules: the device, service, configuration and context models. This section explains the method used to build these models. After designing the knowledge package models shown in Figure 6.3, the researcher added the communication port ontology, as shown in Figure 6.4, to represent individuals populated within each class of the knowledge package model ontologies.

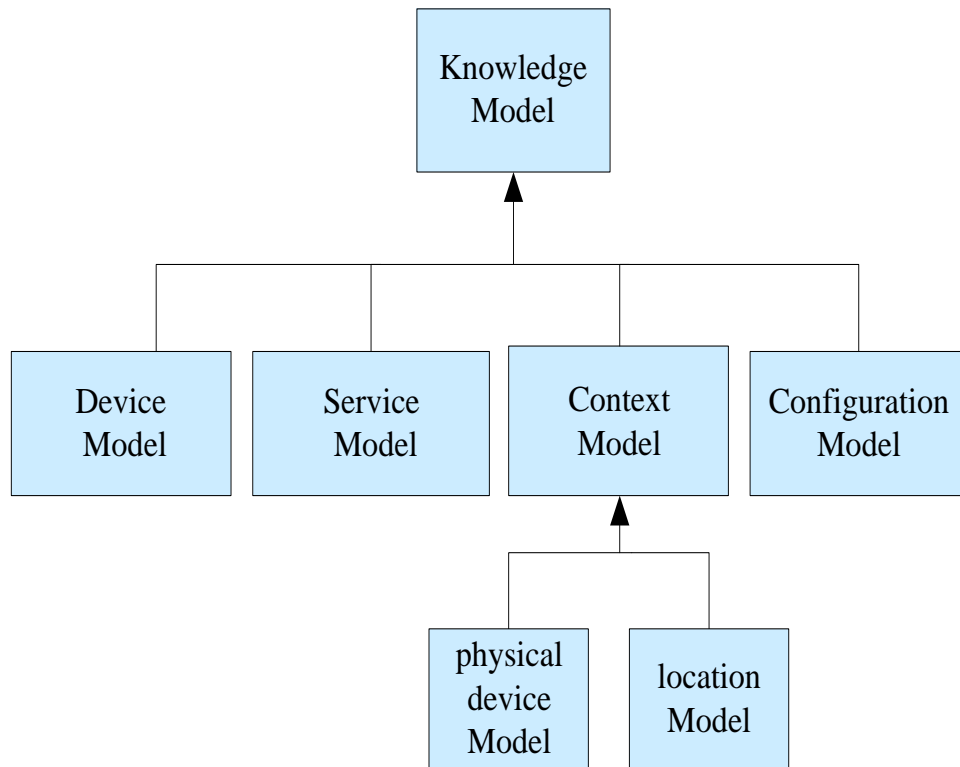


Figure 6. 3: Knowledge Package Model

Figure 6.4 shows the relationships within the knowledge package models ontologies and with the communication port ontology. The service ontologies have a subclass relationship with the device ontology, a constrain relationship with the context ontology and an annotate relationship with the communication port ontology. The device ontology has an annotate relationship with the communication port ontology, whereas the configuration ontology inserts axioms into the communication port ontology.

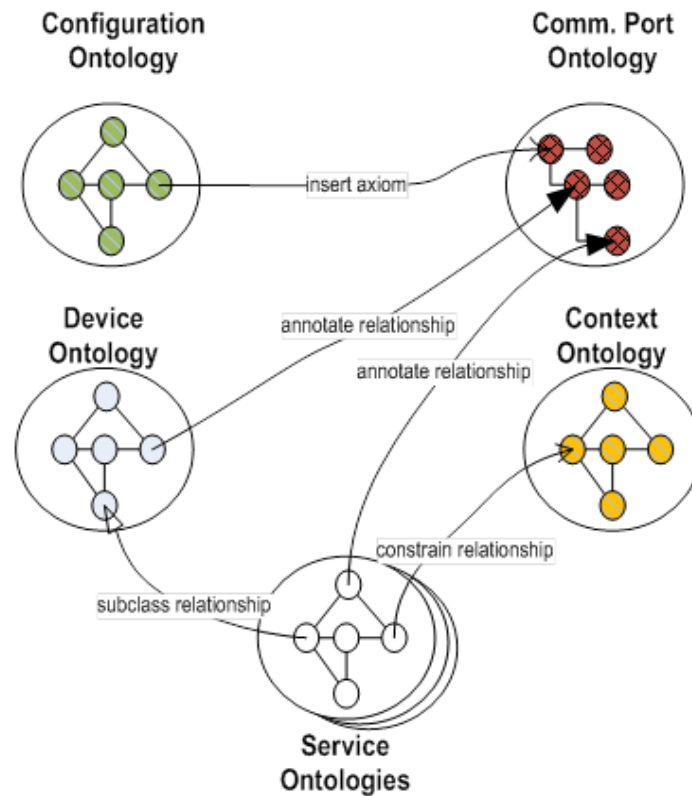


Figure 6.4: Knowledge Layer Model

6.4.2.1 Context Ontology Model

The main purpose of the present context model is to describe the physical environment characterising the device and service operation environment. Since environmental information cannot be automatically digitalised, it will facilitate the installer to digitalise this information. The context ontology model includes two models to describe two physical systems: a location model and a physical device model.

- **The Location Model**

The location model is intended to model the home in order to specify physical entities related to location, segment the physical areas, identify individual physical areas and

describe the relations between them. Location can be represented as a descriptive term (outdoor area or indoor area) indicating a symbolic place in the house. One symbolic area has a floor plan specified by physical address and it contains several indoor and outdoor areas. The indoor area is further divided into one or more floor (e.g. first floor, second floor). Each floor includes several room areas (e.g. lounge, bedroom). The outdoor area is linked to the indoor area via an entrance. One room area can be connected to another room area via the entrance. Similarly, outdoor area can be further divided into small areas (e.g. garden, garage). The outdoor area is linked to the indoor area via entrance. Further description of the location diagram is provided in Chapter 8.

- **The Physical Device Model**

The physical device model is intended to model devices which cannot interact directly via digital messages, but of which the digital system needs to be aware. The status of these devices can be obtained only via add-on sensors and they can be controlled only via add-on actuators. The physical device model enumerates a large number of devices such as lamp, TRV valve and door lock, but the relations within the model are quite simple. These concepts introduced here are those adjudged most important in the smart home environment by assessing their use by resource manufacturers, service providers and vendors on the Smart Home website. It is believed that more complex relations among devices will depend strongly on the related services. More illustration is provided in Section 6.5.2.1.

6.4.2.2 Device Ontology Model

The aim of the device model is to impose a proper structure through which one can model a device across various domains in a consistent fashion. Unlike the physical device model (e.g. garden temperature sensor), the devices within this model can actively interact with a digital ecosystem by generating and/or receiving digital messages. Thus, the device entity model focuses on specifying a general structure to present the meaning and intention of such messages. It holds the attributes and functions related to the particular device.

The attribute (Parameter) is divided into two categories: dynamic and static. While the dynamic parameter (e.g. location) is identified by the installer and can be changed, the static parameter (e.g. manufacturer name) is identified by manufacturers and cannot be changed after production. The message offering and receiving is specified by DataPorts which present the functions. The DataPort is further divided into EventDataPort and ActionDataPort. EventDataPort determine the event message exchange which implies the publish and subscribe interaction scheme. The event message is intended to inform and receive the status, which includes the measurement of physical phenomena (e.g. temperature) related to Measurand, smoke detection (e.g. alarm notification) related to Alarm, and actuator (or action) status (e.g. switch is on) related to ActionStatus. ActionDataPort specifies the action message exchange. The action message presents the received command to manage the environment or the delivered command for another device to perform.

6.4.2.3 Service Ontology Model

The service model, from a configuration perspective, models device (or resource) artefacts from an application perspective, so the service can be connected to and utilise the correct resources. The service model is the knowledge-based model for service suppliers to formally present service-related information. The home service ontology, as a domain upper ontology, introduces key concepts in the smart home environment, by deriving them from the device concept and formalizes their relations. The service model is explained in detail in Section 6.5.2.3.

6.4.2.4 Configuration Ontology Model

Ontology is a tool that supports representing knowledge, not only in terms of concepts, sub-concepts and a hierarchy of relations, but also involving a number of rules and facts such as “IF-THEN” [245]. Rather than introducing a new concept, the configuration model contains the practice of the accessible installer information to establish logical connections in the form of rules. Using these configuration rules in the ontology created here represents its integration with the knowledgebase system. In this case, both the ontology and its rules are embedded in a common logical language, allowing knowledge sharing in a consistent and coherent way. Further elaboration of the configuration model is provided in Chapters 7 and 8.

6.4.3 Brainstorming Important Terms

The rationale for building this smart home system ontology was to provide an ontology which encompasses knowledge of concepts common to popular smart home resources, their various activities, their context and relationships. This has resulted in the

development of reusable/shareable ontologies. To capture fully the smart home resource concepts, the study began by investigating general classifications of smart home devices provided by various resource manufacturers, service providers and vendors on the Smarthome website and discussing them with domain experts. Thus, while no uniformity of resource specifications exists among these vendors, the present ontology has the potential to be adopted widely by vendors and manufacturers to create a uniform classification of smart home devices, which would permit automated reasoning over smart home devices and related characteristics.

The brainstorming process helps to build the structure of a number (first glossary) of pertinent concepts and relations between them from among the available concepts. Brainstorming and consulting experts in the field are good quick methods to refine the resultant list of concepts and their relationships. In other words, the proposed ontology encodes knowledge of the most popular smart home devices, allowing comparison amongst various brands of common smart devices based on their characteristics.

For the purpose of auto-configuration, the concepts in the smart home system domain need to be identified. The first step in building the system is to create the semantic model.

To this end, the designed ontology should meet the following requirements:

- It should align with smart home device manufacturers' practice, making it easy to document device-specific information.

- It should be easy to extend, allowing service suppliers to introduce new concepts from the service perspective.
- It should be able to document the know-how of installer practice, to establish logical links between service application and device resources.

6.4.4 The Modelling Approach

The middle-out approach was adopted for this ontology because of its advantages over the other two approaches. It is based on the combination of both bottom-up and top-down ontology modelling. It strikes a balance in the level of detail, so that detail arises only when needed, by identifying the fundamental concepts. This implies that some effort is avoided. Defining the most fundamental concepts first, and allowing the higher level concepts to be derived from them, results in a natural development of the higher level categories. This facilitates the identification of commonality, which in turn reduces inaccuracies, thus rework and effort. Further examples of the essential concepts identified within the smart home and their specific and abstract ones are shown in the following section.

6.4.5 Identification of Concepts and their Properties

In the use case of the smart home, the service logic model represents the functional logic to realise the service functions, composed of the function provided by the software service or by the device; the service interacts with the device or other services via notification or command message. For example in Figure 6.5, the PIR sensor output (occupancy status notification) and the “receive occupancy status” software function

accept “occupancy status notification” as input, so the PIR sensor can provide a service which is requested by the “occupancy status notification” function.

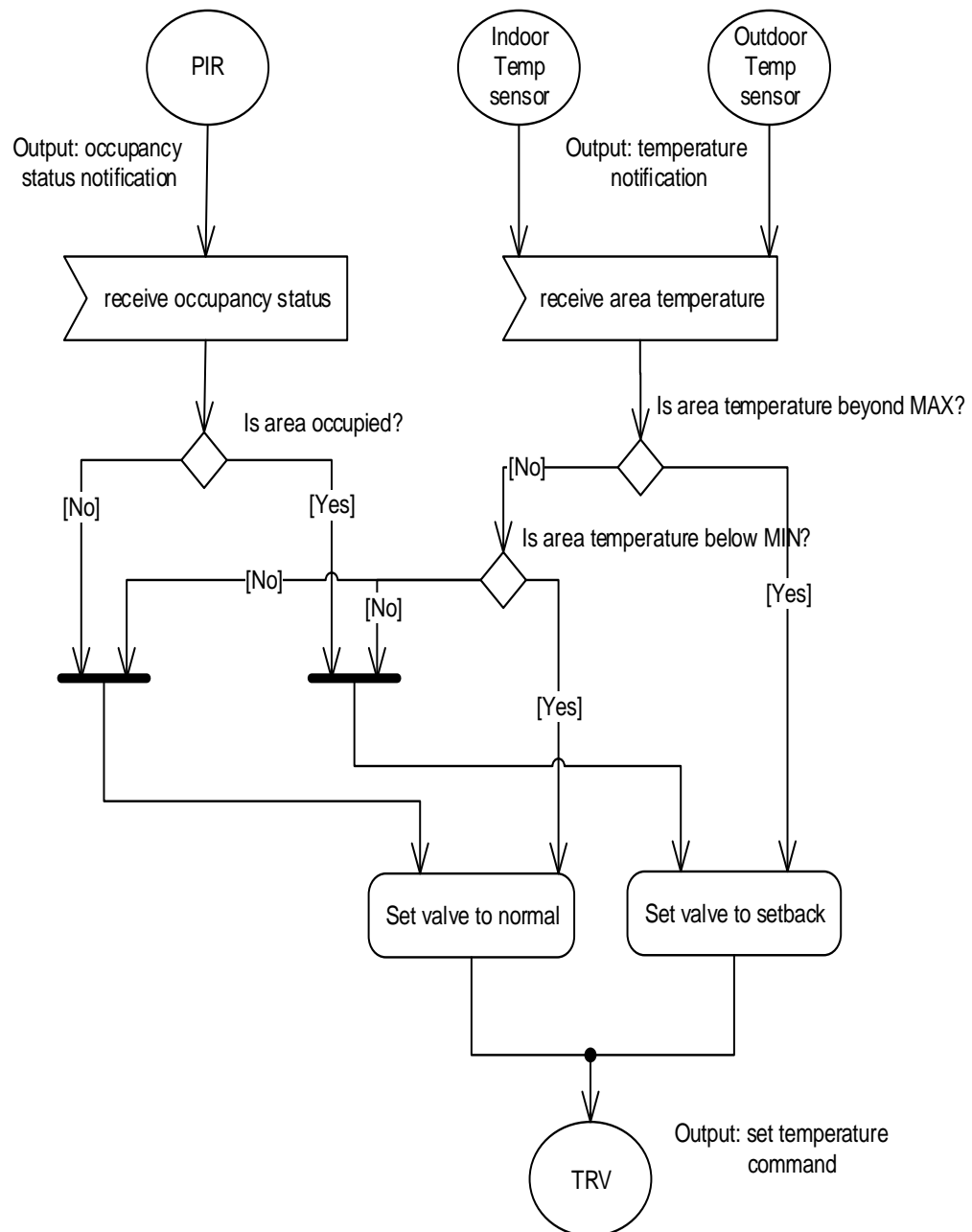


Figure 6.5: Temperature Regulation Service Object

The process of configuration aims to bind the service provision with the service request using the proper interface. For the purpose of configuration, the concepts in the domain of the smart home system such as functionality, service and device need to be identified and the abstract model of those concepts needs to be developed. Based on the service object diagram specified in the use case, some extensions are made in order to more easily discover all the instances of the abovementioned concepts. The devices are added in the proper position in each service object diagram and the messages between the device and service are explicitly indicated. The identified concept instances are listed in examples of these concepts (Temperature Control Service) are shown in Table 6.1

Table 6.1 : Identified Concepts of Temperature Regulation Service Object

Service	Temperature Control Service
Device	1. PIR sensor 2. Temperature sensor 3. TRV valve (Thermometer Radiator Controller)
Function	1. Detect occupancy 2. Monitor room temperature 3. Alter valve status

Based on a preliminary analysis of the development process, the high level basic coarse-grained concepts (classes) are refined into fine-grained concepts in the domain and there is subsequent specialization of the related concepts (subclasses) and their various features and attribute slots (sometimes called roles or properties) in the smart home system.

6.5 Building a Knowledgebase Model Using Protégé

The knowledgebase for the auto-configuration Sobot in the smart home domain was modelled in two steps. The first was to build ontology of applicable concepts, sub-concepts, instances of classes and their relation hierarchy, then the second was to design a number of rules and facts.

6.5.1 Development Languages

The Sobot prototype was developed using the OWL and SWRL languages. The structural knowledge in the smart home domain ontology was represented in OWL, whereas the rule knowledge was formalised in SWRL. This subsection explains and justifies these choices.

6.5.1.1 Web Ontology Language (OWL)

OWL was designed to be employed in applications which need to process information content rather than merely presenting it to humans. The OWL ontology comprises illustrations of the properties of classes as well as their instances. The formal semantics of OWL ontology identifies the way to derive logical consequences, which are facts not overtly apparent in the ontology, but rather determined by the semantics [251]. OWL consists of three expressive sublanguages: OWL Full, OWL DL and OWL Lite [252]. The ontology knowledge model of the project was described using OWL DL, whose interoperability was attained by using XML. OWL was chosen because it supports knowledge reuse and sharing, allowing the model to be updated [253]. The use of OWL DL ensured that reasoning would be computable and decidable [254].

6.5.1.2 Semantic Web Rule Language (SWRL)

There are two kinds of knowledge in the system: static (domain) knowledge, represented by ontologies, and dynamic (inference) knowledge, dealing with the reasoning process and represented by rules [255]. The present knowledge-based system was developed on the basis of OWL ontology and some SWRL rules. The major reasons for selecting SWRL to represent rule knowledge were to take advantage of its formal model-theoretic semantics and to benefit from its close association with OWL, facilitating the incorporation of the OWL-based configuration ontology into the rules [256, 257].

SWRL is also a descriptive language which consists of independent concrete rule implementation languages within inference engines [258]. Therefore, one is free to choose the computing platforms on which to apply the inference processes for configuration. Subsequently, based on structural knowledge which has been modelled in OWL, rule knowledge in the particular domain is expressed in SWRL. This enables users to create Horn-like rules expressed in OWL concepts to infer new knowledge from the existing OWL individuals [259]. As a result of the execution of SWRL rules, newly deduced knowledge was automatically added to the knowledgebase.

Interoperation between OWL and SWRL occurs not only semantically and syntactically, but also inferentially. This means that it is not sufficient to be able to create SWRL rules in OWL which can use the vocabulary and OWL ontology; rather, a vital requirement is to conduct reasoning in a semantically consistent way. This further indicates the value of exploiting both rule-based knowledge and the ontology to draw

inferences. A combination of OWL and SWRL thus affords powerful inferential reasoning capabilities.

The actual configuration reasoning process was then executed using these two components, with the support of the Pellet rule engine, as explained in Chapter 7.

6.5.2 Building the Smart Home Ontology

The knowledgebase model mentioned in Section 6.3 is explained in detail in the followings subsections. The ontology, which was based on the “is-a” relationship and some other relationships between classes, is presented here using graphical notation to make it easier for non-experts to understand its main features. Figures (6.6- 6.20) were generated by the Protégé software.

6.5.2.1 Context Model

Context is defined by Yau et al [250] and Loke [260] as “any information acquired from a system or an environment, which can be employed to characterize the situation of an entity, where the entity can be a person, object or place”. Such entities are related to the communication between an application and its user, as well as to the application and user themselves[261, 262]. The context has also been viewed as a set of descriptions of the static and dynamic aspects of the world. While the former are simply identified as properties, dynamic aspects of the world are represented as events. This information can be acquired by means of sensors. The context model describes the physical environment, characterising the device and service operation environment, and allows the installer to digitalise the information. It includes two models to describe two physical systems: a location model and a physical device model.

- **The Location Model**

The purposes of the location model of a home are to specify the physical entities related to location, to segment the physical areas, to identify individual physical areas and to describe the relations between them. As shown in Figure 6.6, each house has a symbolic area; one symbolic area contains several indoor and outdoor areas; and the outdoor areas are linked to the indoor areas via an entrance.

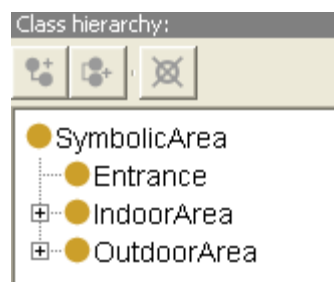


Figure 6. 6: Location Model

Each indoor area is further divided into one or more floors. Figure 6.7 shows the subclasses of IndoorArea: Floor (e.g. ground floor and first floor), Hall, IndoorUnclassifiedArea, Landing, RoomArea and Stair.

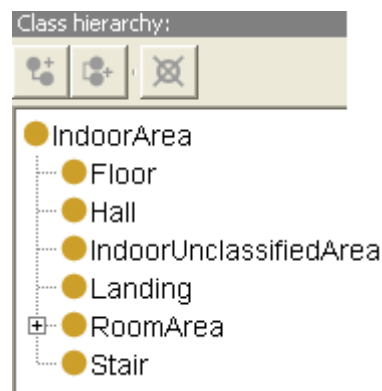


Figure 6. 7: Subclasses of IndoorArea

Similarly, OutdoorArea can be further divided into smaller areas. Figure 6.8 shows the subclasses of OutdoorArea: Garage, Garden, OutdoorToilet, OutdoorUnclassifiedArea, and OutdoorUtilityroom.

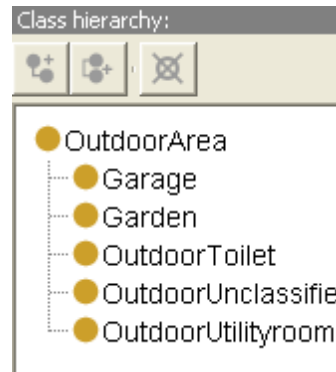


Figure 6.8: Subclasses of OutdoorArea

Each floor includes several room areas (e.g. bedroom, lounge). These rooms are connected via object properties called hasAreaRelation, as shown in Figure 6.20. One room area can connect to another room area via the entrance. The subclasses of the RoomArea are Bathroom, Bedroom, Diningroom, IndoorToilet, IndoorUtilityroom, Kitchen, Livingroom, Lounge and Office, as shown in Figure 6.9.

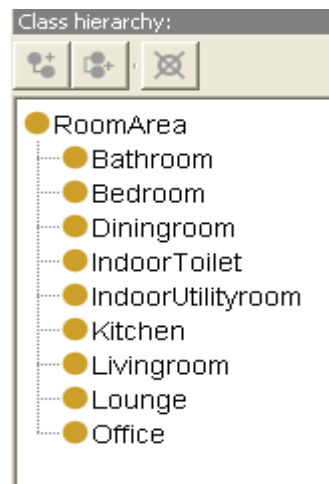


Figure 6.9: Subclasses of the RoomArea

- **The Physical Device Model**

The physical device model status can be obtained only via add-on sensors and they can be controlled only via add-on actuators. The concepts introduced here are those adjudged most important in the smart home environment by assessing their use by resource manufacturers, service providers and vendors on the Smart Home website. Figure 6.10 shows that the physical device model enumerates a large number of devices and the relations within it are quite simple (i.e. they are at the same level). It is believed that more complex relations among the devices would depend strongly on the service perspective. Thus, further specification should be carried out in the service model for the purpose of auto-configuration. The PhysicalDevice class consists of ConsumerUnit, DVDPlayer, GasValve, HiFi, Lamp, Lock, PlayStation, TRValve, TV and WaterValve.

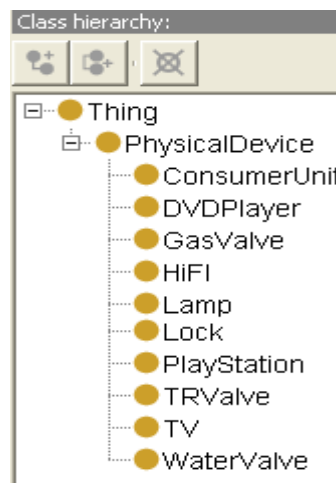



Figure 6.10: Physical Device Model

6.5.2.2 Device Model

The aim of the device model is to impose a proper structure through which one can model a device across various domains in a consistent fashion. Devices within this

model, unlike those within the physical device model, can interact actively with the digital ecosystem by generating and/or receiving digital messages. Thus, the device model focuses on specifying a general structure to present the meaning and intention of messages. Figure 6.11 shows DeviceEntity as a subclass of Entity within the whole ontology. It represents an independent physical entity within the digital environment and is thus considered to be the host, holding all attributes and functions related to the specific device. The symbol  means that the subclasses of that class are equivalent.

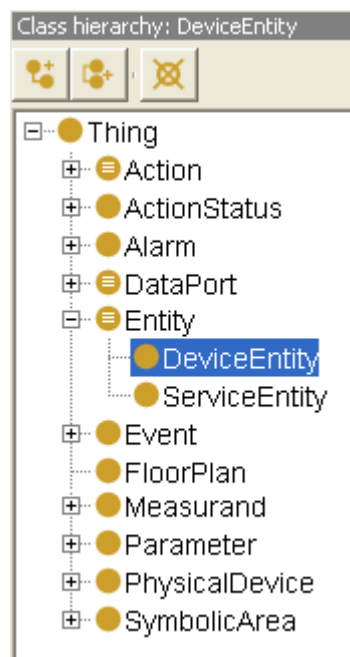


Figure 6.11: Device Model

The attribute named Parameter can be divided into two categories: static and dynamic. A static parameter (e.g. manufacturer name, measurand) is specified by the manufacturer and is not changed after production, while a dynamic parameter such as location is specified by the installer and is subject to change by users. The subclasses of

Parameter as shown in Figure 6. 12 are BooleanParameter, DescriptionParameter, FloatParameter, IntegerParameter and LocationParameter.

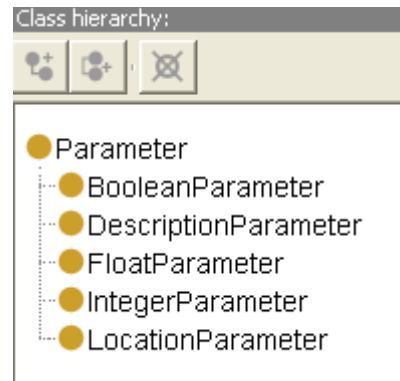


Figure 6. 12: Subclasses of Parameter

As shown in Figure 6.13, the subclasses of Measurand are BatteryMeasurand, ContactMeasurand, EnergyMeasurand, HumidityMeasurand, OccupancyMeasurand, MovementMeasurand and TemperatureMeasureand. The subclasses of EnergyMeasurand are ElectricityConsumptionMeasurand and GasConsumptionMeasurand.

OccupancyMeasurand and MovementMeasurand are equivalent classes. The subclasses of TemperatureMeasurand are IndoorTemperatureMeasurand and OutdoorTemperatureMeasurand.

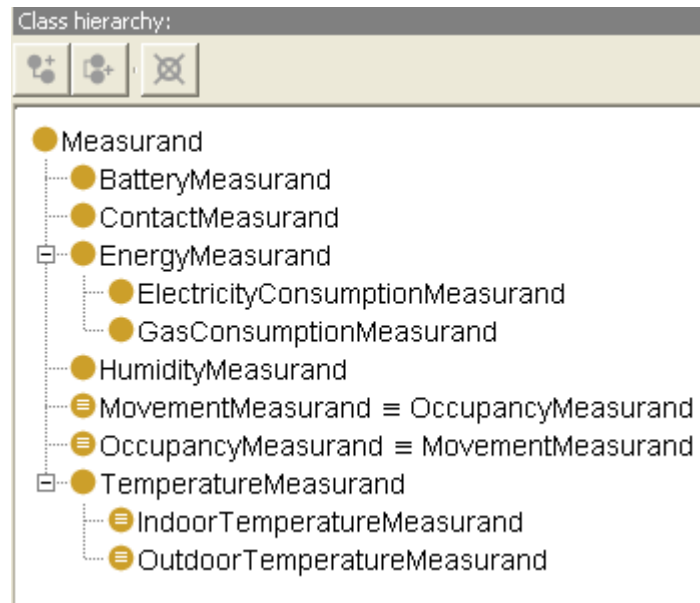


Figure 6.13: Subclasses of Measurand, EnergyMeasurand and TemperatureMeasurand

The subclasses of Event are shown in Figure 6.14. They are ActionStatusEvent, AlarmEvent and MeasurandEvent.

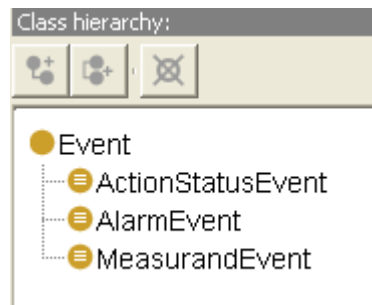


Figure 6.14: Subclasses of Event

The functions are represented by DataPort, which specifies the sending and receiving of messages. DataPort is further classified into ActionDataPort and EventDataPort, as shown in Figure 6.15. The EventDataPort specifies the exchange of event messages, while ActionDataPort specifies that of action messages. An event message implies the publishing and subscription of an interaction scheme. Its main purpose is to convey and

receive information on status, including the measurement of physical phenomena (e.g. temperature), related to Measurand, smoke detection (alarm notification), related to Alarm, and actuator (or action) status (e.g. switch is on) related to ActionStatus.

DataPort belongs to the communication ontology and is directly linked to the device ontology. An action message represents a command received to manipulate the environment or issued for another device to execute. It has two parts: undertaken action and final status. The combination the information in these two parts will specify an activity.

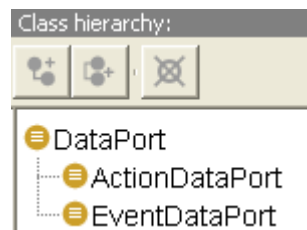


Figure 6.15: Subclasses of DataPort

Figure 6.16 shows the subclasses of Action and ActionStatus. The subclasses of Action are SetBooleanStatus and SetTRValve. The subclasses of SetBooleanStatus are SetConsumerUnit, SetGasValve, SetLock, SetSwitch and SetWaterValve. The subclasses of SetSwitch are SetEntertainmentSwitch and SetLightLampSwitch. The subclasses of ActionStatus are BooleanStatus and FloatStatus. The subclasses of BooleanStatus are ConsumerUnitStatus, GasValveStatus, LockStatus, SwitchStatus, TRValveStatus and WaterValveStatus.

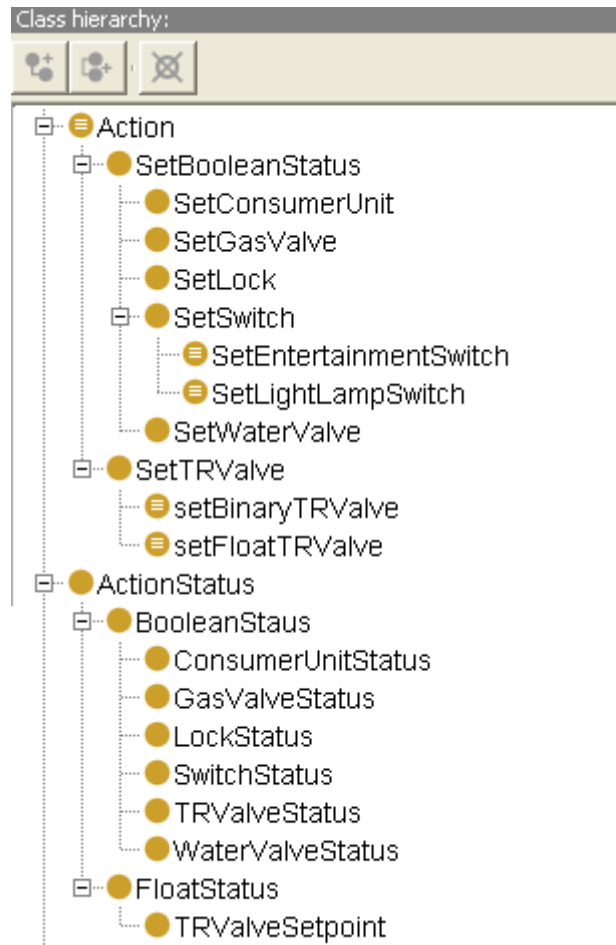
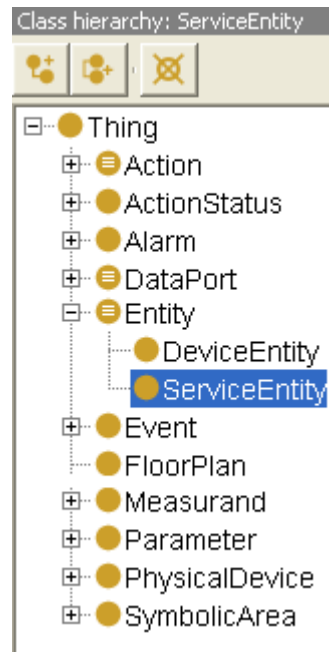


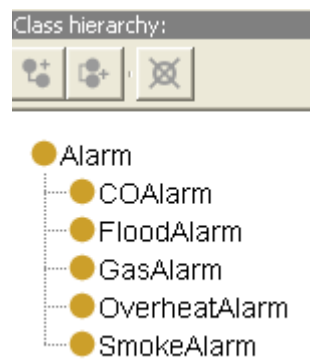
Figure 6.16: Subclasses of Action and ActionStatus

6.5.2.3 Service Model

As shown in Figure 6.17, ServiceEntity is a subclass of Entity within the whole ontology. It allows each service to be connected to and utilise the correct resources. The service model introduces key concepts by deriving them from the device concept. The main methods of derivation can be summarised as refinement and reclassification. For refinement, IndoorTemperature is defined as a temperature measurement whose location is IndoorArea.

**Figure 6.17: Service Model**

The subclasses of Alarm are COAlarm, FloodAlarm, GasAlarm, OverheatAlarm and SmokeAlarm as shown in Figure 6.18.

**Figure 6.18: Subclasses of Alarm**

Overall, the main classes of the iCARE ontology model are thirteen subclasses of Thing (the most general class of the ontology) as shown in Figure 6.19: Action, ActionStatus,

Alarm, DataPort, Entity, Event, FloorPlan, Measurand, Parameter, PhysicalDevice and SymbolicArea.

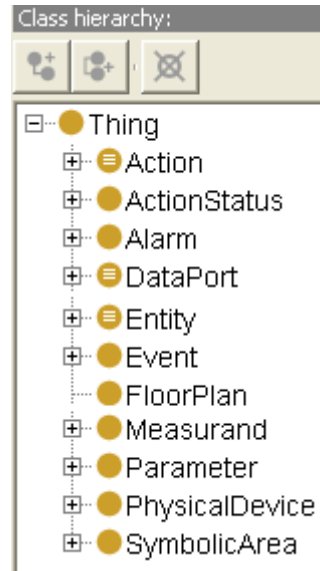


Figure 6.19: Main Classes of the iCARE Ontology Model

Object properties and data properties hierarchies of this system are shown in Figure 6.20. An object property relates individuals of a class to other individuals of other class, whereas a data type property relates individuals to data type values such as integers, floats, Booleans and strings.

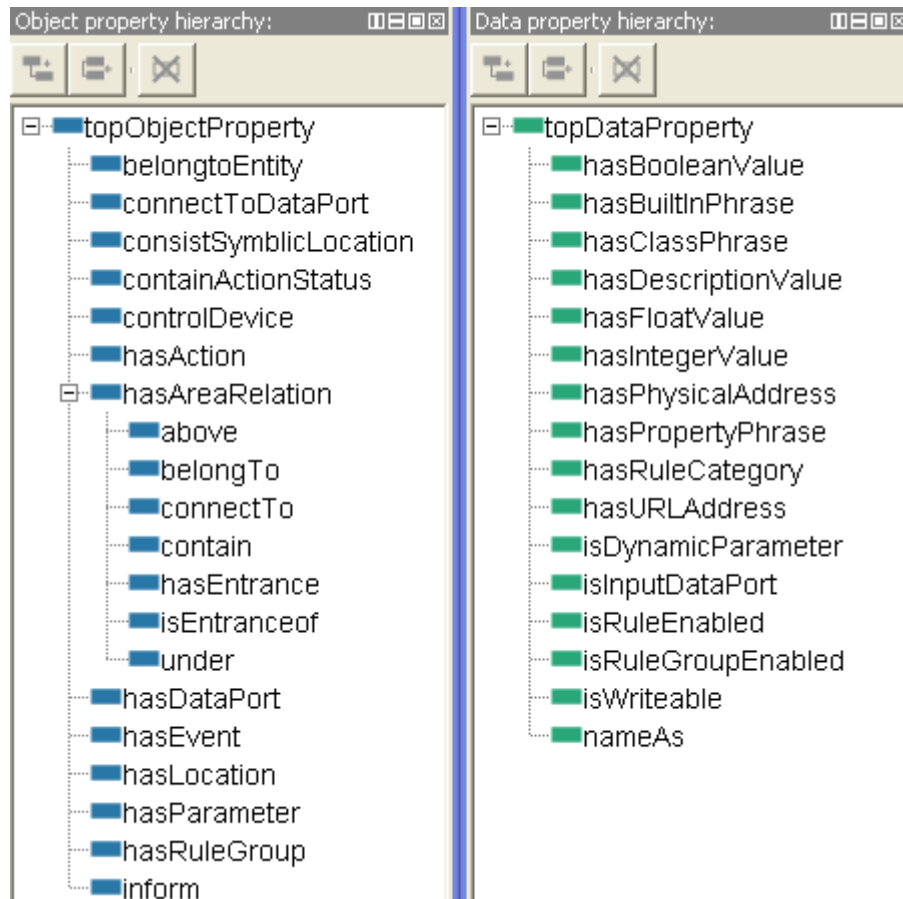


Figure 6.20: Object Property Hierarchy and Data Property Hierarchy

6.5.3 Configuration Rule Design

It is useful here to return to the use of “IF-THEN” rules. It is essential to note that the policies represented in the examples are ECA rules, which could also be termed as showing cause-effect relationships between environment processes. These rules and facts are encoded using SWRL whose RuleML component provides a neutral platform for the semantic interoperation of many rules including the ECA type. SWRL exploits logic operators to describe rules, and SWRL rules are stored as concepts in the ontology. Using these inference rules in the proposed ontology represents its integration with the knowledgebase system. In this case, both the ontology and its rules are

embedded in a common logical language, allowing knowledge sharing in a consistent, coherent way.

SWRL is exploited in this study to express a variety of rule knowledge which exists in configuration models. Therefore, SWRL rules are vital in this case to support OWL DL usage, because [101]:

- OWL is only able to place restrictions on a single property at a time.
- As a consequence of the first limitation, the resource identity criteria are also restricted to one slot at a time.
- OWL is unable to state that one property could be composed of two other properties.

SWRL rules can overcome the above limitations and complement OWL. The reasons for selecting SWRL were to overcome the lack of expressivity in OWL DL (including OWL Lite) and to sustain the necessary decidability in a knowledgebase which includes rules [263]. The undecidability occurs as a result of adding rules to OWL-DL, which leads to the loss of any form of model property [263].

SWRL has a formal model-theoretic semantics and a very close association with OWL. Consequently, it can properly incorporate the OWL-based configuration ontology into the rules. Furthermore, SWRL is a descriptive language which is to select the computing platforms which really execute the inference processes for configuration. Accordingly, based on the structural knowledge modelled in OWL, rule knowledge in the target domain is expressed in SWRL.

To ensure decidable (a complete and effective in determining if a variable is derivable or not) reasoning, DL-safe SWRL [200], a subset of SWRL, was used. The so-called DL-safe rules are added to SWRL rules to restrict their operation to identified individuals of the ontology, clearly presented in the Abox [264]. This restriction is adequate to make SWRL rules decidable. In the DL-safe rules method, classes and roles are permitted to happen in the antecedent and consequent of Horn-like rules in the datalog [263]. Horn-like rules, named after the logician Alfred Horn, consist of an antecedent (a set of atoms) and a consequent (a single atom), where all the variables in the consequent should happen in at least one of the antecedent's atoms [265] .

SWRL enables to write Horn-like rules which can be expressed in a form of OWL concepts in order to support more powerful reasoning abilities than OWL alone [266]. SWRL helps to extend the OWL axioms to comprise Horn-like rules [200]. DL-safety guarantees that each variable is bound to individuals which are clear in the ABox. A rule is considered safe when each variable in the consequent also become visible in the antecedent. This safety condition for SWRL rules is meant to add further expressive power and at the same time sustain decidability. The implementation of structural and rule knowledge in smart home domain ontology is described in the following chapter.

The knowledgebase consists of the identified smart home concepts and their relations represented in OWL and from rule knowledge formalised in SWRL.

The Sobot has an inference (reasoning) engine that derives recommendations from the knowledgebase models. By utilising the local knowledgebase, the configuration plan generation engine will generate a valid configuration plan. The reasoner is software

used to infer logical consequences based on a set of declared facts or axioms. The inference engine component consists of inferred rules and a forward-chaining rule reasoning engine for performing intelligent reasoning, which in turn generates a final configuration. The reasoning engine generates a feasible configuration that meets the given requirements using the smart home project specifications, the knowledgebase model and Pellet resources configuration in the project.

Reasoning mechanisms are required for checking and validating the consistency of the knowledge model, as shown in Appendix A.1, besides checking and testing the related rules for understanding the context by determining if contradictory concepts exist and in which cases. The final configuration process is performed by the configuration plan executor. The result of this process is the auto-configuration of the devices and/or the services leaving or joining the smart home system. The configuration plan executor automatically executes the generated configuration plan to ensure that the system is configured without interrupting the normal operation, based on the knowledge models and the inferring rules specified in the reasoning engine.

User-defined roles utilizing SWRL were employed to benefit from further inference. Description Logic allows the identification of a terminological hierarchy utilizing a constrained set of first-order rules. This allows OWL to utilize the substantial existing body of DL reasoning to fulfil essential logical requirements, namely concept satisfiability, class consistency and individual checking. A subset of reasoning rules which support OWL [102] is shown in Table 6.2

Table 6.2 : OWL Ontology Reasoning Rules [102]

Name	Rule	
	IF	THEN
Subclassof	$(?X \text{ rdfs:subClassOf } ?Y)$ $\wedge (?Y \text{ rdfs:subClassOf } ?C)$	$\rightarrow (?A \text{ rdfs:subClassOf } ?C)$
Subpropertyof	$(?X \text{ rdfs:subPropertyOf } ?Y)$ $\wedge (?Y \text{ rdfs:subPropertyOf } ?C)$	$\rightarrow (?X \text{ rdfs:subPropertyOf } ?C)$
Inverseof	$(?P \text{ owl:inverseOf } ?Q) \wedge (?X ?P ?Y)$	$\rightarrow (?Y ?Q ?X)$
Symmetric	$(?P \text{ rdf:type owl:SymmetricProperty})$ $\wedge (?X ?P ?Y)$	$\rightarrow (?Y ?P ?X)$
Disjointwith	$(?C \text{ owl:disjointWith } ?D)$ $\wedge (?X \text{ rdf:type } ?C) \wedge (?Y \text{ rdf:type } ?D)$	$\rightarrow (?X \text{ owl:differentFrom } ?Y)$

For this method, the Pellet reasoner [267] was utilised along with a set of rules written in SWRL including two OWL sub-languages: OWL DL and OWL Lite[211]. SWRL basically deals with Horn-like rules which are in the form of an implication between an antecedent (body) and a consequent (head). If the condition identified in the antecedent is met, the consequent's specified condition must also be met. They can be written as "IF-THEN" statements. Both the antecedent and consequent comprise zero or more atoms. Atoms can be of the form $C(x)$, $P(x, y)$, same as (x, y) or different from (x, y) , where C is an OWL class, P is an OWL property, x and y are variables, OWL individuals or OWL data values. SWRL uses logic operators like the logical connectors

AND (\wedge) and IMPLY (\rightarrow) to create the rules, which are then stored as concepts in the ontology. The first section of the SWRL identifies the reasoning rules, while the second identifies the appropriate action to be taken. Table 6.3 shows the rule atoms and their descriptions.

Table 6.3 : SWRL Rule Atoms [101]

Atom	Description
C(x)	C stands for a particular OWL class or data range while x is either a variable, an OWL instance or an OWL data value. This is referred to as a class or concept atom.
P(x, y)	P is an OWL property x is either an OWL instance or a variable. y can be an OWL instance, a variable, or an OWL data value. The P atom is referred as a property or role atom.
sameAs(x, y)	This indicates that x is the same as y. x and y are variables or OWL instances
builtIn(r, x, ..., z)	The builtIn atom is utilized when the ontology developer needsto employ one of the built-in relations in SWRL. Where r is thebuilt-in relation and x, ..., z are OWL data values.
builtIn(r, x, ..., z)	The builtIn atom is utilized when the ontology developer needsto employ one of the built-in relations in SWRL. Where r is thebuilt-in relation and x, ..., z are OWL data values.
differentFrom(x, y)	This indicates that the given x is different from y . where x and y are variables or OWL instances

The knowledgebase can also be effectively sustained by reasoning over inconsistencies with the help of the Pellet DL reasoner, which is firmly equipped with Protégé. SWRL,

which is natively included in the Protégé editor, facilitates the operator's encoding of the rules. Detailed processes for implementing the proposed approach are elaborated in Chapter 7.

Rather than introducing new concepts, the configuration model mimics the practice of the installer to establish the logical connection in the form of rules. Since the device entity and service entity host the parameter, it is necessary to propagate the parameter into the low level artefacts (e.g. data port, informing measurement). The following parameter propagation rules are designed and will be further illustrated in Chapter 8.

Entity(?en), hasDataPort(?en, ?d),hasParameter(?en, ?p)->hasParameter(?d, ?p) (1)

DataPort(?d), hasEvent(?d, ?e), hasParameter(?d, ?p) -> hasParameter(?e, ?p) (2)

Event(?e), hasParameter(?e, ?p), inform(?e, ?m) -> hasParameter(?m, ?p) (3)

6.6 Summary

This chapter has presented a knowledgebase model for the auto-configuration Sobot within a smart home environment proposed in Chapter 5. The implementation of this model, which is a semantic description of the application and a set of governing rules, is reported in Chapter 7. This chapter has also set out the approach followed in building the model. The first step in developing the Sobot was to model its knowledgebase, which could then be used as a framework to describe the various elements of the smart home environment. The knowledgebase comprises an ontology, with a set of concepts, sub-concepts, instances of classes (individuals) and their relations hierarchy, together

with a number of rules and facts. The structural knowledge in the smart home domain ontology is represented in OWL, whereas the rule knowledge is formalised in SWRL.

The body of this chapter has listed the steps taken in developing the ontology and addressed the defining of class hierarchies, data and object properties and individuals. Bearing in mind that there is no one correct ontology for any particular domain, the approach recommended in “Ontology Development 101: a guide to creating your first ontology”[241] was adopted here. The knowledgebase comprised four modules: a device model, a service model, setup knowledge and a context model. Designing an ontology is a creative process, which means that no two ontologies created by different persons will be the same and that their quality can be assessed only by using them in the applications for which they were designed.

Chapter 7: Sobot Implementation

7.1 Introduction

The Sobot conceptual design, set out in Chapter 5, was realised by adopting the Open Services Gateway initiative (OSGi) framework [268] to achieve auto-configuration of the smart home system based on inferred facts about the environment and ontological knowledge. It allows the system to adapt to these requirements and their complexity. The Sobot development environment, including the ontology development environment and the reasoning engine, is illustrated in Section 7.2. The rationale behind adopting the OSGi framework in this work is that it is a modular runtime environment which reduces complexity by offering a modular architecture not only for large-scale distributed systems but also for small embedded applications [269].

It supports different application configurations at runtime [270]. The characteristics of the OSGi are presented in Section 7.3. A detailed account of the implementation of the prototype Sobot on the OSGi framework is illustrated in Section 7.4. The workflow for Sobot auto-configuration is provided in Section 7.5 and the chapter ends with a summary.

7.2 Sobot Development Environment

The implementation of the Sobot was empowered by the knowledgebase description and reasoning tools and by OSGi technology [271]. Consequently, the main Sobot development environment consists of the OWL and SWRL languages, Protégé empowered by Pellet reasoning engine, the Eclipse OSGi bundle development environment and the OSGi evaluation environment. These are described in detail in the following subsections.

7.2.1 Ontology Development Environment

The smart home ontology was created using the Protégé-OWL editor, version 4.2. Protégé is a flexible open source platform which provides easy integration of a set of tools to build and edit the ontology of domain models and knowledge-based applications [36, 203, 272]. The use of OWL is significant because it allows the distribution of ontological knowledge [37, 273]. The ontology editor used here provides methods and classes to load and save OWL files, to query and manipulate OWL data models and to execute reasoning. It implements a rich set of knowledge-modelling that enables users to load and save OWL ontologies, define logical class characteristics as OWL expressions and to edit and visualize classes and properties. Protégé was found to provide a flexible base for quick ontology development. Its primary advantage over other platforms is that it can deal with XML, the

main format used in interoperating among the different tools in the prototype. It also offers a user-friendly interface and a well-supported document and error-checking mechanism[37, 274] .

7.2.2 Reasoning Engine

One of the great advantages of the ontology is the reasoning capability of the knowledgebase, allowing new knowledge to be inferred from any clearly defined knowledge. An additional advantage of this reasoning capacity is the ability to sustain the consistency of the ontology throughout its development. Pellet was employed here to check the consistency of the OWL documents and to infer supplementary facts from the knowledge and relationships within the ontology model of the given environment. It is an open source reasoning engine written in Java which provides cutting-edge reasoning services for OWL, as it [275-278]:

- Is a sound and complete OWL DL reasoner with extensive support for individuals (instances) reasoning.
- Has been found a reliable tool for operating with OWL DL ontologies with OWL extensions.
- Provides a variety of interfaces for loading ontologies.

The present knowledge-based system was developed on the basis of OWL ontology and some SWRL rules which are illustrated in Section 6.5.1. Interoperation between OWL and SWRL occurs not only semantically and syntactically, but also inferentially. A combination of OWL and SWRL thus affords powerful inferential reasoning capabilities.

The following is an illustration of the benefits of using Pellet [279-281]:

- **Consistency checking** guarantees that the ontology contains no contradictory facts.
- **Classification** computes the subclass relations amongst every given class to generate the complete class hierarchy, which in turn can be employed to answer specific queries, such as identifying only or all of the direct subclasses of a class.
- **Realization** finds the most particular classes to which an individual relates, i.e. it computes the direct types for each individual. Realization can be executed hierarchy. Based on the classification hierarchy, it becomes possible to identify all the types for certain individuals.

7.3 The OSGi Framework

The OSGi framework has recently become the most widely adopted technology for building automation systems for the networked home [282-286]. It was mainly built to maintain software component configuration [285]. It also has the following advantages [286]:

- It can deal with different technologies which in turn increases system interoperability. It is a modular system and service platform for the Java programming language, which deploys a dynamic component model [287]. It provides greater modularity and portability than standard Java programming because applications can be joint ones, from numerous existing bundles [137, 288].

- It is a runtime environment which reduces system complexity by offering a modular architecture for large distributed systems as well as small embedded applications. It supports different application configurations at runtime [270].
- The OSGi dynamic model integrates several devices in a networked environment. This helps to reduce operational costs and facilitate device interaction, which enhances smart home service delivery. It runs on a Java virtual machine to provide a shared deployment environment which installs, uninstalls and updates bundles without requiring a restart of the system [288].
- It accelerates service and application deployment and development via open specifications. It hosts several services from various providers on a single gateway platform and supports many home-networking technologies [289]. This promotes ease of use for home network service consumers and service providers [290].
- The OSGi service platform runs different applications under strict and secure control of a management system through the following specifications, which provide powerful tools to securely control the operation of the device [289, 291].
 - i. It is an open service platform providing a safe support for easy delivery of various downloadable and extensible Java-based service applications, called bundles [292].
 - ii. It provides a built-in mechanism to support the life-cycle management (install, start, stop, update and uninstall) for

bundles. In particular, the update process is executed without restarting the Java Virtual Machine (JVM). All applications are started in the same JVM, which in turn helps to save memory and resources. At the end of the lifecycle of a bundle, the code and its resources are cleared from the system [293].

- iii. Java 2 Code Security is chosen to offer the permissions mechanism, which protects resources from particular actions. Each bundle has limited access to other resources, based on a specified permissions policy. Files can be protected using File Permission classes which take the name of the resource as well as a number of actions as parameters. Each bundle consists of a set of permissions which can be changed on the fly and new permissions are instantly efficient once obtained [294]. In other words, each bundle has its resources and actions. A resource is protected by asking the Java Security Manager to check if there is permission to execute an action on that resource. The Security Manager then ensures that all callers on the stack have the necessary permissions. This process helps to protect the resource from attackers [285].

Figure 7.1 shows the OSGi layered model. Bundles are the software application components developed and plugged into the framework. They are Java Archive (JAR) files containing Java class files to register and execute one or more services [292]. They

can be remotely installed, started, stopped, updated and uninstalled without any disruption of the operation of the whole system. All applications in an OSGi Service Platform are started in the same JVM. This saves resources, memory, and CPU cycles [285]. Bundles can cooperate with other bundles by providing application components known as services. A bundle deployed in an OSGi service platform can provide application functions to other bundles [295]. Services are specified by a Java interface and link bundles dynamically, where bundles can implement this Java interface and register the service with the service registry, which enables bundles to detect any services added or removed and adjust accordingly. This registration process advertises these services so that other bundles can utilize them. The framework handles dependencies between bundles and services to enable coordination between them [292] .

Bundle lifecycle management, which refers to starting, stopping, installing, etc., is done via application programming interfaces (APIs), which enable remote downloading of the bundles' management policies. The framework also provides application developers with a reliable programming model by defining only interfaces amongst the framework and the registered services [296]. The modules define how a bundle can export or import code, while the security layer manages the various aspects of security.

Finally, the execution environment determines which methods and classes are available on a particular platform. This separation of service definition from implementation guarantees that services provided by different companies interoperate on the managed framework [32]. Because of the benefits of the OSGi mentioned above, the Sobot was implemented as a bundle on the OSGi framework, which provided lifecycle management support for the Sobot through the use of OSGi APIs.

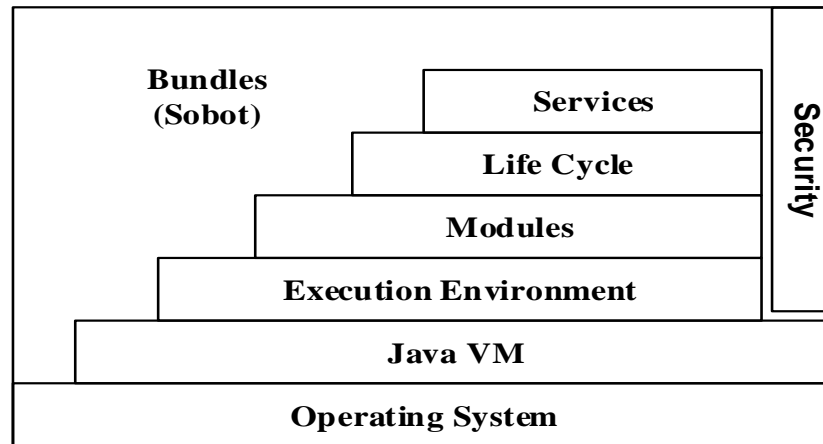


Figure 7.1: OSGi Architecture [268]

7.4 Implementation of the Sobot Prototype in the OSGi Framework

The prototype of the auto-configuration Sobot's components, all of which are interdependent, were implemented as a bundle on the OSGi framework as shown in Figure 7.1, providing the home environment with the features required to transform it into an autonomic system, capable of making inferences from the list of the services offered and available and from contextual information. The following subsections describe in turn each of the four major components of the Sobot and their functions.

The overall architecture of this prototype is depicted in Figure 7.2. The OSGi framework was able to simplify and speed up the whole process of prototyping the Sobot by reducing complexity through offering a modular architecture not only for large distributed systems but also for small embedded applications. The architecture of the proposed configuration Sobot has four major components: a resource handler, a local knowledgebase, a configuration plan generation engine and a configuration plan executor, each having its own subcomponents.

7.4.1 The Resource Handler

The smart home system configuration is processed by the Sobot, which begins to perform the reasoning operation once any change (the appearance of a new object or the disappearance of an existing one) is detected in the environment. The functional units in the resource handler are responsible for providing the mechanisms that gather, filter and report all contextual details collected from managed resources, i.e. devices, services and their context status. The resource handler consists of three functional units: resource monitoring, the resource information translator and the configuration event generator.

In detail, the resource monitoring unit is responsible for capturing and monitoring all contextual information collected from various resources running on the smart home network and detecting any changes in the contextual environment parameters. It specifically provides knowledge regarding requirements and specifications of the iCARE resources (services, devices and context) needed for decision making.

The resource information translator encodes the relevant information and reports it to the configuration event generator. The specifications and requirements are presented in XML files, and environment-based knowledge in standard files. Different data sources may be utilized if they are mapped into the designed ontology that constitutes the knowledge module. The configuration event generator then generates the configuration events relevant to the data provided by the resource information translator.

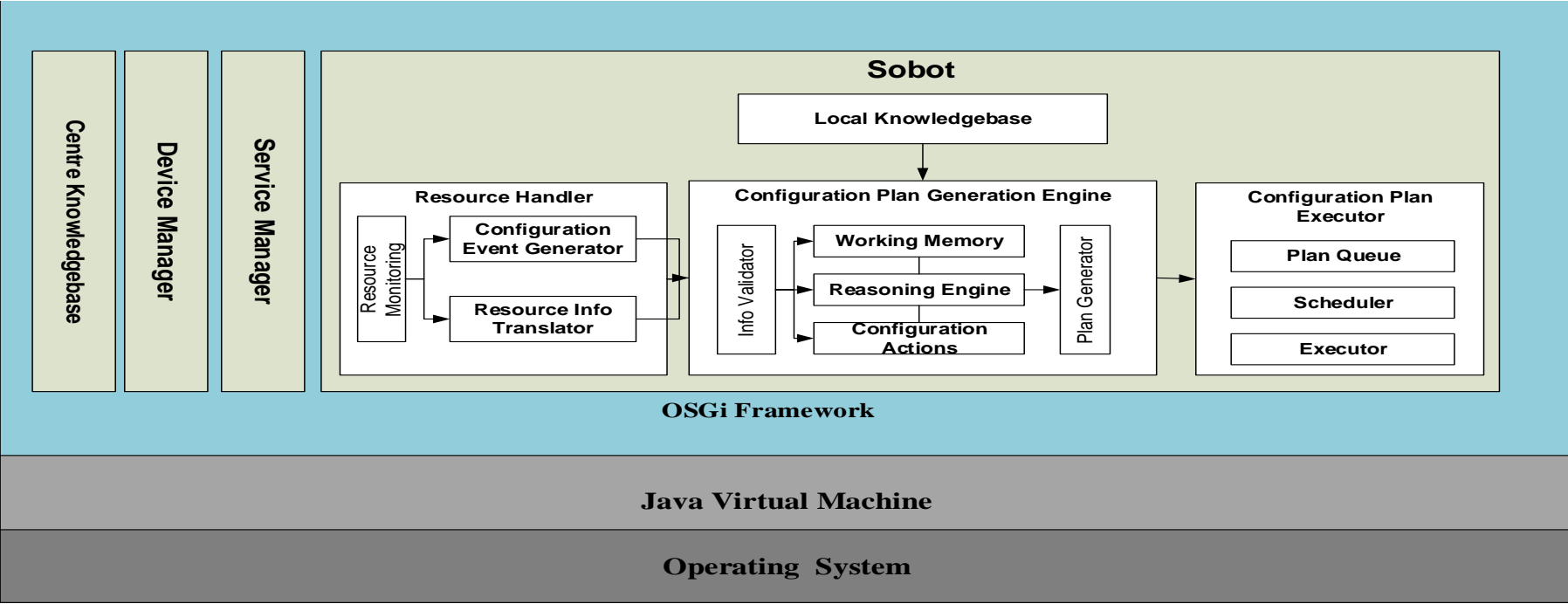


Figure 7.2: Sobot Architecture within the OSGi Framework

7.4.2 The Local Knowledgebase Model

The local knowledgebase is at the core of the decision-making system in the proposed approach. It localises the aforementioned central knowledgebase for a particular environment and extracts the requisite local knowledge from it. Additionally, it can cope with off-line situations and maintains the essential description of the domain that the Sobot requires. The knowledgebase model extracts explicit information, which is then analysed and modelled. In particular, it models all of the concepts associated with the smart home environment and their relationships. The semantics of these concepts and their relations must be defined accurately and precisely. Without a comprehensive and semantically rich knowledgebase, it would be difficult for the system to execute the required configuration tasks. The Sobot obtains this knowledge from an ontology repository and utilizes it for all configuration activities of the smart home.

Based on the contributor of knowledge, the knowledge model is divided into four modules: the device model, the service model, setup knowledge and the context model. The device model allows device manufacturers to provide information formally describing the devices, while the service model is for service suppliers to provide formal service-related information such as telehealthcare monitoring. A device within this model can interact actively with a digital ecosystem by generating and/or receiving digital messages. Thus, the device model focuses on specifying a general structure to present the meaning and intention of messages, so that the service can be connected to and utilise the correct resources. The context model is used by installers to describe the physical environment so that the devices and services to be deployed can be specified.

The context model has two components to describe two physical systems: a location model and a physical device model. The location model, described in detail in Section 8.2, is designed to model the home in order to specify the location of physical entities, to segment physical areas, to identify individual physical areas and to describe the relations among them. The physical device model serves to model devices that cannot interact directly via digital messages, but which the digital system needs to be aware of. The status of such devices can be obtained only via add-on sensors and they can be controlled only by add-on actuators. The setup knowledge formally encodes the practice and experience of installers to create logical connections between devices and services. The activities usually carried out by the installers, such as linking services with the required devices, will be performed automatically.

7.4.3 Configuration Plan Generation Engine

Once the required facts about the environment have been captured, the configuration plan generation engine starts to operate, based on the output from the resource handler, when any change in the environment is detected, thus activating the inference engine. The configuration plan generation engine, which provides the techniques that create the actions required to accomplish the target goals and objectives, consists of five functional units: the information validator, the working memory, the reasoning engine, configuration actions and the plan generator, which together are responsible for the reasoning function. The information validator first validates the information coming from the resource handler by comparing it with a reference device. The configuration planning mechanism uses rules to guide its work. The ontology model (OWL) file and the SWRL human-defined and machine-processed rules are then loaded into the

working memory and the input to the reasoning engine is analysed according to the specified rules, which are generated from high-level policies to control the behaviour of resources within the home system. The reasoning engine derives the recommendations from the knowledgebase models. By utilising the local knowledgebase, the configuration plan generation engine generates a valid configuration plan.

The reasoning engine, as explained in Section 7.2.2, infers logical consequences from a set of declared facts or axioms, i.e. according to sets of rules predefined in the scope of the ontology concepts. It loads the set of individuals related to the home context (devices and installed services) and to available services, then accomplishes reasoning by matching this contextual information with the defined rules to obtain new individuals (instances of classes or concepts) which describe the set of services that suit the context. The reasoning engine component consists of inferred rules and a forward-chaining rule reasoning engine for performing intelligent reasoning, which in turn generates a final configuration. The reasoning engine thus generates a case-based viable configuration that meets the specified requirements using the knowledgebase model.

Reasoning is required to check ontology consistency and the implied relationships, besides testing and checking the rules created for understanding the given context. Two types of reasoning are identified: ontological reasoning, which is used to check the ontology itself for inconsistencies between its classes and their relations, and user-defined rule-based reasoning, which uses first-order logic to compute inferred reasons. Once there is an inference output, the configuration action component decides the appropriate actions, taking the inference engine output (a set of individuals describing services) and preparing a list of potential services. Thus, the output of the inference

engine contains a list of service individuals that are to be conveyed to the configuration plan executor.

7.4.4 Configuration Plan Executor

The functional units in the final component, the configuration plan executor, together provide the mechanisms that control the execution of a configuration plan, allowing for dynamic updates. By means of the plan queue, scheduler and executor, the configuration plan executor automatically executes the generated configuration plan to ensure that the system is configured without interrupting normal operation, based on the knowledge models and the inferring rules specified in the reasoning engine. The newly inferred facts are then conveyed to the running services and devices via an XML message bus to enable them to react appropriately to the dynamic environment.

The result is an auto-configuration of the devices and /or the services leaving or joining the iCARE system. The results are then stored in the context model held in the working memory. It should be stated that the ontology repository file (stored in OWL) is not updated in light of the newly deduced facts, since these are not part of the permanent context but are subject to dynamic change.

7.5 Workflow for Sobot Auto-configuration

The Sobot configuration process within the smart home environment is illustrated in the form of a flow chart in Figure 7.3, which shows that any change of service or device status will update the local knowledgebase, based on the properties of existing services and devices in that environment. The reasoning engine will be activated by triggering the SWRL rules, based on the updated local knowledgebase, to generate a viable

configuration which will satisfy the specified requirements based on the ontology knowledge model. It utilizes Pellet to create a logical link between the service and the appropriate device individuals via a data port, then starts to check whether the required link exists in the local ontology model as the sequence of viable next links is identified in the ontology. If all of these links are available, it then finds the logical links that can efficiently connect them. It accesses the first satisfied required link and marks it as the current link, while searching to find the next satisfied required link. In case it does not find such required logical links that connects them, it indicates that it cannot connect them. Subsequently, it keeps searching until it finds the next required link, otherwise the system remains on the current configuration. Once a new configuration has been successfully generated, the Sobot configuration plan executor deploys it in the system.

This process is not straightforward, as heterogeneous devices are managed by different controllers. The plan executor therefore utilizes the designed knowledge model of the smart home environment to decide the best way to perform the new configuration and specifies the controllers that control the targeted resources. Auto-configuration within the smart home environment is achieved by automatically linking the required individuals, based on the reasoning engine's newly inferred facts (i.e. the logical consequences of the existing individuals) according to predefined rules. Chapter 8 provides further illustration of how to achieve smart home auto-configuration.

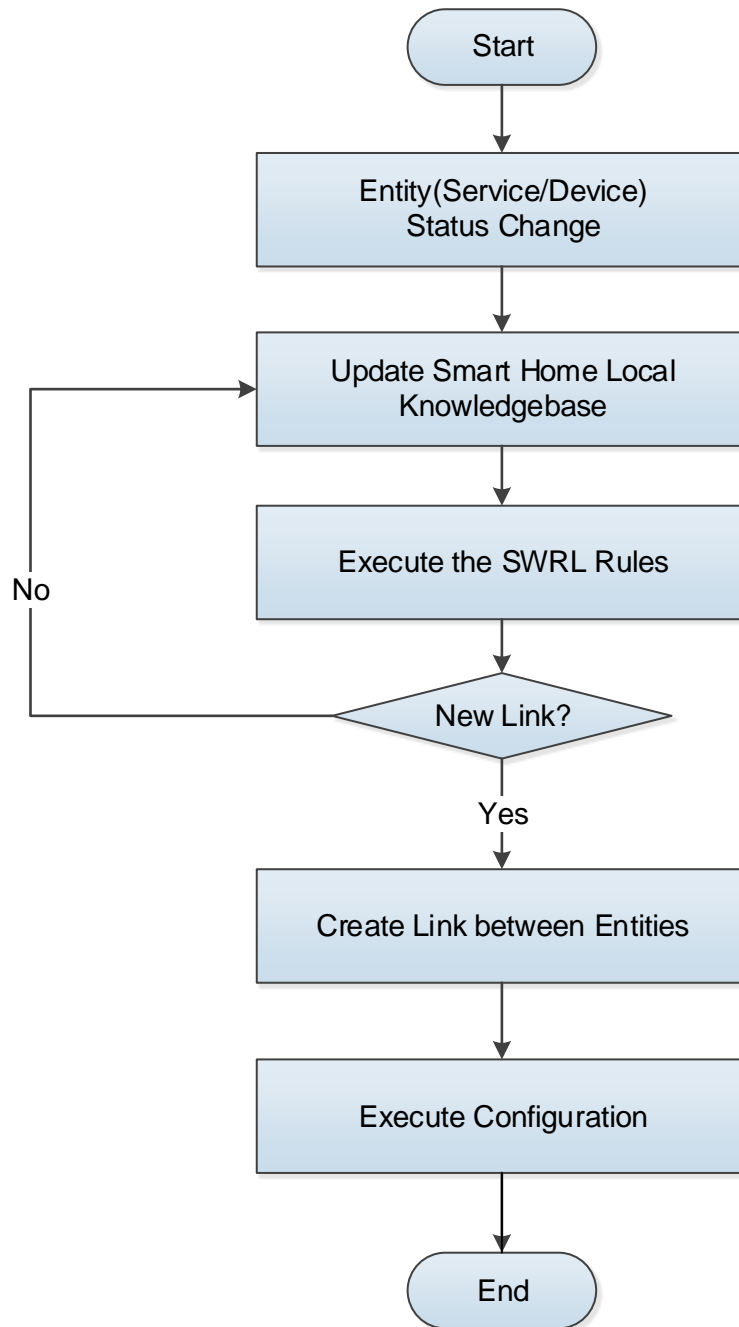


Figure 7.3: Workflow for Sobot Auto-configuration

7.6 Summary

This chapter has explained the development and implementation of the Sobot for auto-configuration in a smart home environment, using Protégé as the ontology design environment, OWL and SWRL as the development languages and Pellet as the reasoning engine. In order to properly validate the auto-configuring system, an execution environment that allows real scenarios to be emulated was developed. The OSGi framework was used to run the system, because it has bridges to several technologies used in smart home automation systems and offers high-level implementation constructs. The chapter ended by explaining the workflow of the Sobot configuration process. The following chapter reports an experimental case study to assess and validate the performance of the Sobot auto-configuration system.

Chapter 8: Validation of the Sobot Prototype

8.1 Introduction

This chapter presents an evaluation of the Sobot prototype within a smart home environment, based on conducting experimental case studies to evaluate its viability and capabilities in identifying and responding to configuration of smart home systems. The test cases involved using the Sobot to generate configurations to allow various services to explore available devices and to deliver their capabilities. The prototype is assessed in terms of a number of attributes: flexibility, awareness and adaptability of the Sobot as well as the usefulness and effectiveness of the ontology and the associated inference mechanism.

This chapter is organised as follows: Section 8.2 describes the testbed developed to mimic the real home environment for assessment of the proposed Sobot. Section 8.3 gives details of the scenarios for the case studies carried out for validating the Sobot's

capabilities in connecting the heating control service to temperature sensors. Section 8.4 explains the resource instantiation, Section 8.5 considers the usefulness and effectiveness of the inference mechanism and Section 8.6 presents the evaluation of the results in light of the performance metrics and evaluation criteria. The chapter ends with a summary.

8.2 The Testbed

To validate the use of the Sobot configuration approach within the context of the smart home environment, a testbed (pioneered by MRG) was constructed to replicate the setup of a real house and a set of experiments were conducted to assess the Sobot's ability to automatically configure logical connections among services and devices within the smart home environment. The Sobot is expected to enable any service applications, once delivered, to utilise the available and connected devices, taking into account the smart home settings, to meet the user's requirements. The testbed floor layout is shown in Figure 8.1 as the representative environment for the delivery of various services (e.g. healthcare, home security and energy management) to the smart home.

As Figure 8.1 shows, the testbed contains all representative functional areas for viability assessment. It is desirable because the characteristics of different functional areas or rooms within the smart home environment are likely to be closely linked to the available devices and services. Typical British homes have eight functional areas or rooms: bedroom, hallway, kitchen, lounge, toilet, bathroom, garden and office [297]. The floor layout of the testbed consists of several indoor and outdoor functional areas. A functional area is classified as indoor if it is assumed that the room has a roof.

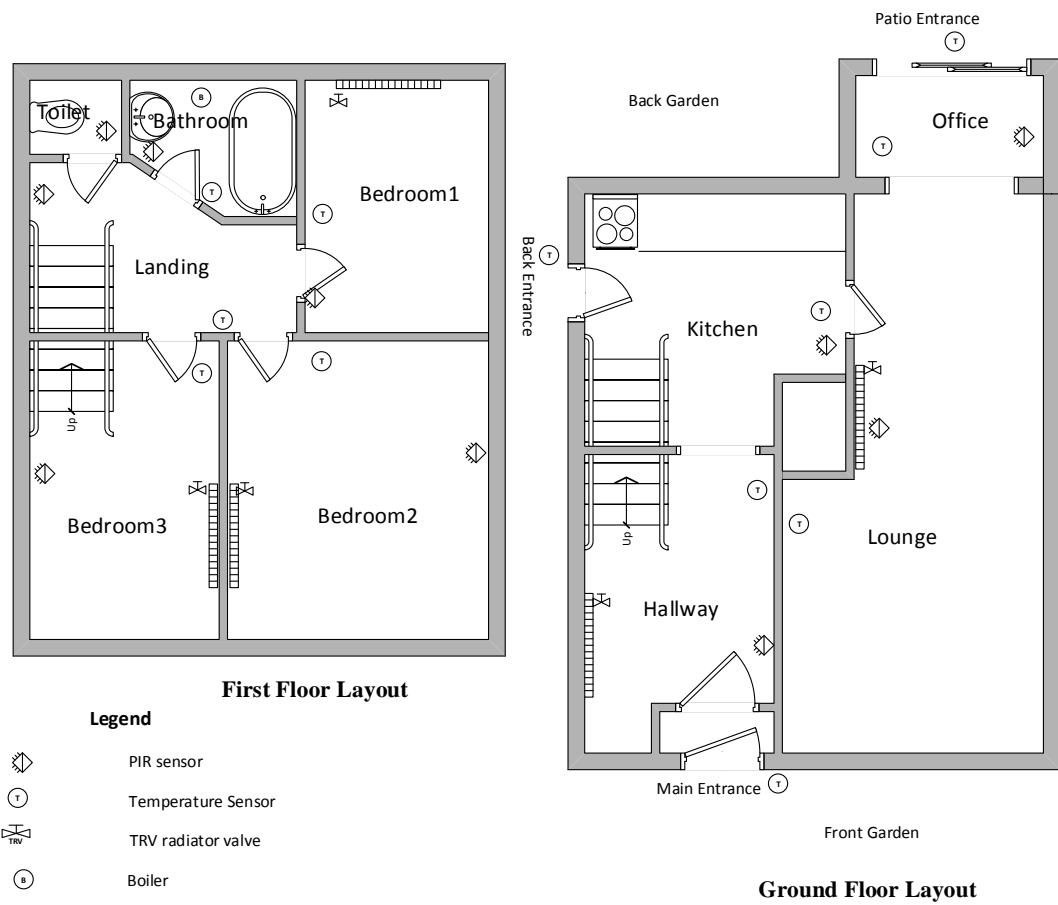


Figure 8.1: Testbed Floor Layout

Based on their physical locations, the indoor functional areas are grouped into the ground floor and first floor, by convention and according to the requirements of potential services. Each room is connected to adjacent rooms via entrances. Similarly, the outdoor area is linked to the indoor area via entrance(s). There are three entrances (the main entrance, back entrance and patio entrance). The testbed is equipped with multiple devices (sensors and actuators). In particular, a passive infrared (PIR) sensor, a temperature sensor and a thermostatic radiator valve (TRV) are installed in the functional areas as shown in Table 8.1.

Table 8.1: Sensors and Actuators in the Testbed

Functional area	Temperature sensor	PIR	TRV	Boiler
Kitchen	1	1	-	-
Hallway	1	1	1	-
Lounge	1	1	1	-
Bedroom1	1	1	1	-
Bedroom2	1	1	1	-
Bedroom3	1	1	1	-
Landing	1	1	-	-
Office	1	1	-	-
Bathroom	1	1	-	1
Toilet	-	1	-	-
Front Garden	1	-	-	-
Back Garden (Back Entrance)	1	-	-	-
Back Garden (Patio Entrance)	1	-	-	-

8.3 The Scenarios for Case Studies

Heating control scenarios were selected for the case studies, mainly because the heating control service is not only a typical and representative service of the UK domestic environment, but also demanded by most householders. High energy prices and environmental concerns mean that householders seek different ways to reduce their energy consumption. The primary energy wastage [298] related to the space heating could be ascribed to daily activities such as heating an unoccupied house, or oversetting TRVs when the house is not occupied or not actively occupied (e.g. sleep).

Automating energy conservation activities is considered an effective way to reduce energy consumption in the home environment and thus lower energy costs, without affecting the occupants' comfort. In the smart home environments, energy conservation strategies are devised to mimic the occupants' conservation activities to address wastage. As a large number and various contexts of sensors and actuators are involved, significant configuration efforts are required to fully achieve these energy conservation strategies. The proposed auto-configuration (Sobot) approach becomes necessary and desirable to address this issue.

The heating control service consists of several energy conservation scenarios summarised as follows:

8.3.1 Early Start and Stop Scenarios

The strategy for the early start and stop scenarios uses three variables—active occupancy, preferred internal temperature and external temperature—to reduce an area's heating operation time. These strategies incorporate the anticipated adjustment of

the temperature. The system also learns about factors such as the rate at which a space heats up, which is itself a function of the heat transfer parameters of the space and the external temperature, and adjusts accordingly. It is commonly recognised that a house or area should be at the residents' preferred temperature as soon as it begins to be actively occupied (e.g. when they arrive home or wake up), whereas it can be set to a lower temperature when not actively occupied (e.g. when the home is empty or the occupants asleep). It is noteworthy that there is a predefined lowest temperature which users can set in order to prevent the home temperature falling below their comfort level.

For the early start, it brings the thermostat setpoint to comfort temperature forward as shown in Figure 8.2, so that the room temperature can reach the preferred temperature at a preset time regardless of the environmental conditions (e.g. external temperature). This strategy provides greater comfort to the users so that they do not need to raise the setpoint of the boiler thermostat when the room temperature is lower than anticipated. Energy can be saved by avoiding the adjustment of the thermostat setpoint, since absent-minded users are liable to leave it at the higher setting longer than needed or intended.

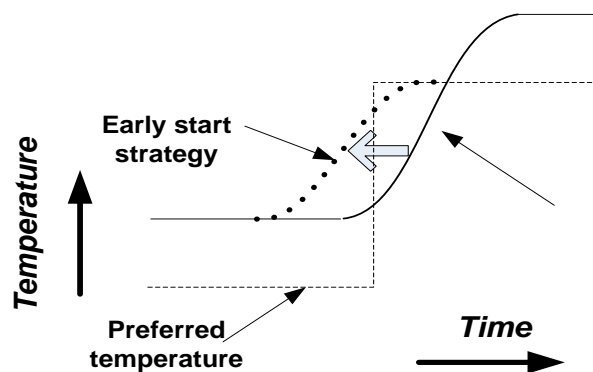


Figure 8.2: Early Start Strategy

As to the early stop, it brings the thermostat setpoint to setback temperature forward at a preset time as shown in Figure 8.3 by exploiting the phenomenon of thermal mass inertia of the house. It generates the energy saving by reducing the heating demand.

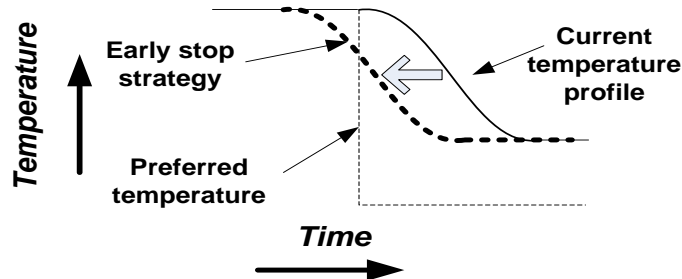


Figure 8.3: Early Stop Strategy

8.3.2 Non-occupancy Setback Scenario

The non-occupancy setback scenario is designed to control the setpoint temperature of a TRV according to the occupancy status of the designated area. In case no activity is identified in a predetermined timeframe, it considers the designated area not actively occupied and lowers the setpoint temperature of its TRV within the area taking into consideration a predefined lowest temperature which users can accept. It also brings temperature back to the user's comfort level once an activity is identified within this area. This saves energy by reducing the heat supply to unoccupied areas. In order to avoid switching the TRV setpoint too frequently if the area is quickly reoccupied, a delay time has been specified.

8.4 Resource Instantiation

In order to carry out the configuration process, the resources need to be instantiated, based on the proposed semantic knowledge model. In this context, two resource types

are involved. The first type represents physical entities that form the environmental context to be sensed, measured or actuated: people, locations and physical objects. This resource type needs to be specified by a human during installation or the initialisation process. For example, the installer needs to specify a location named “Back Garden”, which is an instance of the class “Garden”.

The second type of resource comprises the digital components of the smart home (e.g. digital devices, application services) offering digital functions. The digital devices can provide information about the environment. The application service can make logical operation decisions, based on sensing information received. Unlike the first type of resources, the second is automatically populated by a transforming mechanism, because this second type of resources within the testbed has been developed and/or adopted as being able to provide the interface description encoded in the Web Services Description Language - Semantics (WSDL-S) standard. The transformation from WSDL-S to the defined ontology model is illustrated in Figure 8.4.

Since each device or service has one interface definition, each will map to an entity and its attributes will map to parameters. The organisation structure (e.g. service, port, portType) will be collapsed into the *DataPortGroup*. *DataPort* will be used to represent each operation. Since operation input or output is annotated by a predefined concept, the input and/or output will be classified into the corresponding Event or Action class. The devices and services can introduce new concepts by defining new concepts based on existing concepts encoded in SWRL. For example, *OutdoorTemperature* can be defined as measuring outdoor temperature.

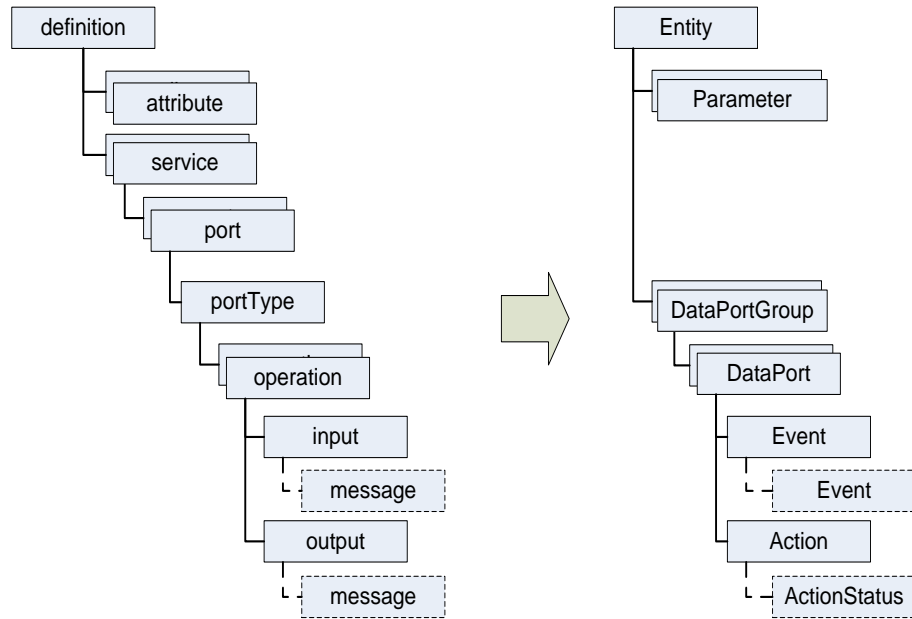


Figure 8.4: Transforming Diagram

Each entity can be presented as a functional block which contains an input port, output port and parameters. The functional block presentations of the services and devices involved are clearly illustrated, as shown in Figure 8.5. It is noticeable that the device interface does not syntactically match the service interface. For example, the temperature sensor provides temperature output and the heating control service requires *OutdoorTemperature* and *IndoorTemperature*. This challenge will be resolved through semantic reasoning.

The individuals of the devices and services are summarised in Table 8.2. The entity is either device or service name. It can also be the device ID or service ID. Each entity has one or more *DataPort(s)*. The individual of the *DataPort* of the entity is in the form *Entity Name_DataPortNo*. Each *DataPort* has one Event or Action. The individual of

the Event or Action of the DataPort is in the form of *(Entity Name_DataPortno_ (Event/Action)No*. Each event must inform certain phenomenon. The individual of the phenomenon is in the form of *PhenomenonNo*

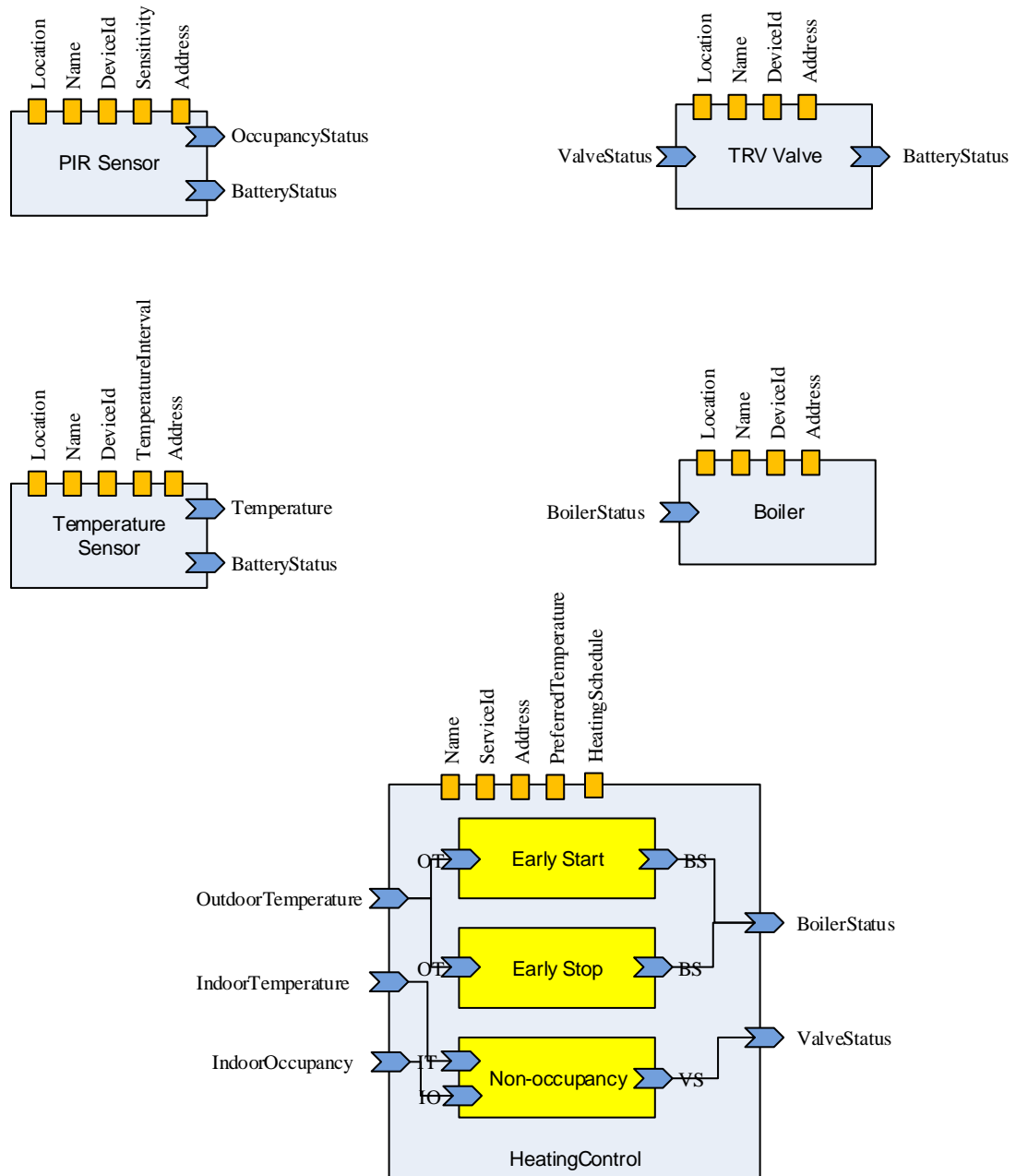


Figure 8.5: The Functional Block Presentations of the Services and Devices Involved

Table 8.2: Individuals by Type

Serial No.	Entity	DataPort	Event/Action	Measurand/ Control
1	HeatingControlService	HeatingControlService_DataPort1	HeatingControlService_DataPort1_Event1	Temperature1
2	GardenTemperatureSensor	GardenTemperatureSensor_DataPort2	GardenTemperatureSensor_DataPort2_Event2	Temperature2
3	KitchenTemperatureSensor	KitchenTemperatureSensor_DataPort3	KitchenTemperatureSensor_DataPort3_Event3	Temperature3
4	KitchenPIRSensor	KitchenPIRSensor_DataPort4	KitchenPIRSensor_DataPort4_Event4	Occupancy4
5	HallwayTemperatureSensor	HallwayTemperatureSensor_DataPort5	HallwayTemperatureSensor_DataPort5_Event5	Temperature5
6	HallwayPIRSensor	HallwayPIRSensor_DataPort6	HallwayPIRSensor_DataPort6_Event6	Occupancy6
7	HallwayTRVactuator	HallwayTRVactuator_DataPort7	HallwayTRVactuator_DataPort7_Event7	Control7
8	LoungeTemperatureSensor	LoungeTemperatureSensor_DataPort8	LoungeTemperatureSensor_DataPort8_Event8	Temperature8

Serial No.	Entity	DataPort	Event/Action	Measurand/ Control
9	LoungePIRSensor	LoungePIRSensor_DataPort9	LoungePIRSensor_DataPort9_Event9	Occupancy9
10	LoungeTRVactuator	LoungeTRVactuator_ DataPort10	LoungeTRVactuator_ DataPort10_Event10	Control10
11	Bedroom1TemperatureSensor	Bedroom1TemperatureSensor_DataPort11	Bedroom1TemperatureSensor_DataPort11_Event11	Temperature11
12	Bedroom2TemperatureSensor	Bedroom2TemperatureSensor_DataPort12	Bedroom2TemperatureSensor_DataPort12_Event12	Temperature12
13	Bedroom3TemperatureSensor	Bedroom3TemperatureSensor_DataPort13	Bedroom3TemperatureSensor_DataPort13_Event13	Temperature13
14	Bedroom1PIRSensor	Bedroom1PIRSensor_DataPort14	Bedroom1PIRSensor_DataPort14_Event14	Occupancy14
15	Bedroom2PIRSensor	Bedroom2PIRSensor_DataPort15	Bedroom2PIRSensor_DataPort15_Event15	Occupancy15
16	Bedroom3PIRSensor	Bedroom3PIRSensor_DataPort16	Bedroom3PIRSensor_DataPort16_Event16	Occupancy16
17	Bedroom1TRVactuator	Bedroom1TRVactuator_ DataPort17	Bedroom1TRVactuator_ DataPort17_Event17	Control17

Serial No.	Entity	DataPort	Event/Action	Measurand/ Control
18	Bedroom2TRVactuator	Bedroom2TRVactuator_ DataPort18	Bedroom2TRVactuator_ DataPort18_Event18	Control18
19	Bedroom3TRVactuator	Bedroom3TRVactuator_ DataPort19	Bedroom3TRVactuator_ DataPort19_Event19	Control19
20	LandingTemperatureSensor	LandingTemperatureSensor_DataPort20	LandingTemperatureSensor_DataPort20_Event20	Temperature20
21	LandingPIRSensor	LandingPIRSensor_DataPort21	LandingPIRSensor_DataPort21_Event21	Occupancy21
22	OfficeTemperatureSensor	OfficeTemperatureSensor_DataPort22	OfficeTemperatureSensor_DataPort22_Event22	Temperature22
23	OfficePIRSensor	OfficePIRSensor _ DataPort23	OfficePIR_DataPort23Sensor_Event23	Occupancy23
24	BathroomTemperatureSensor	BathroomTemperatureSensor_DataPort24	BathroomTemperatureSensor_DataPort24_Event24	Temperature24
25	BathroomPIRSensor	BathroomPIRSensor_DataPort25	BathroomPIRSensor_DataPort25_Event25	Occupancy25
26	ToiletPIRSensor	ToiletPIRSensor_DataPort26	ToiletPIRSensor_DataPort26_Event26	Occupancy26

Table 8.3 shows the symbolic location of the functional area of the testbed

Table 8.3: Symbolic Location

Area Name	Location Class	Super Class
Bedroom1	Bedroom	Indoor
Bedroom2	Bedroom	Indoor
Bedroom3	Bedroom	Indoor
Kitchen	Kitchen	Indoor
Hallway	Hallway	Indoor
Lounge	Lounge	Indoor
Landing	Landing	Indoor
Office	Office	Indoor
Bathroom	Bathroom	Indoor
Toilet	Toilet	Indoor
Front Garden	Garden	Outdoor
Back Garden (Back Entrance)	Garden	Outdoor
Back Garden (Patio Entrance)	Garden	Outdoor

8.5 The Usefulness and Effectiveness of the Inference Mechanism

To check the usefulness and effectiveness of the inference mechanism, the designed SWRL rules should be able to infer new facts from the instantiated individuals without explicit description. The newly inferred facts help the Sobot to auto-configure a proper linkage between the service and the required device within the smart home system without direct user intervention and without disturbing the operation of the system.

Based on the syntax symbol alone, the linkages among the system components are illustrated in Figure 8.6. The diagram indicates that only the boiler and TRV actuator can be correctly connected to HeatingControlService based on the syntax symbol match. The main reason is that OutdoorTemperature, IndoorTemperature and IndoorOccupancy are new concepts introduced by the HeatingControlService and derived from primary concepts (i.e. Temperature and Occupancy). Through semantic inference, a match can be found at the semantic level. The results of this reasoning are illustrated in Figure 8.7. The following section uses OutdoorTemperature as an example to explain how the reasoning process has been carried out.

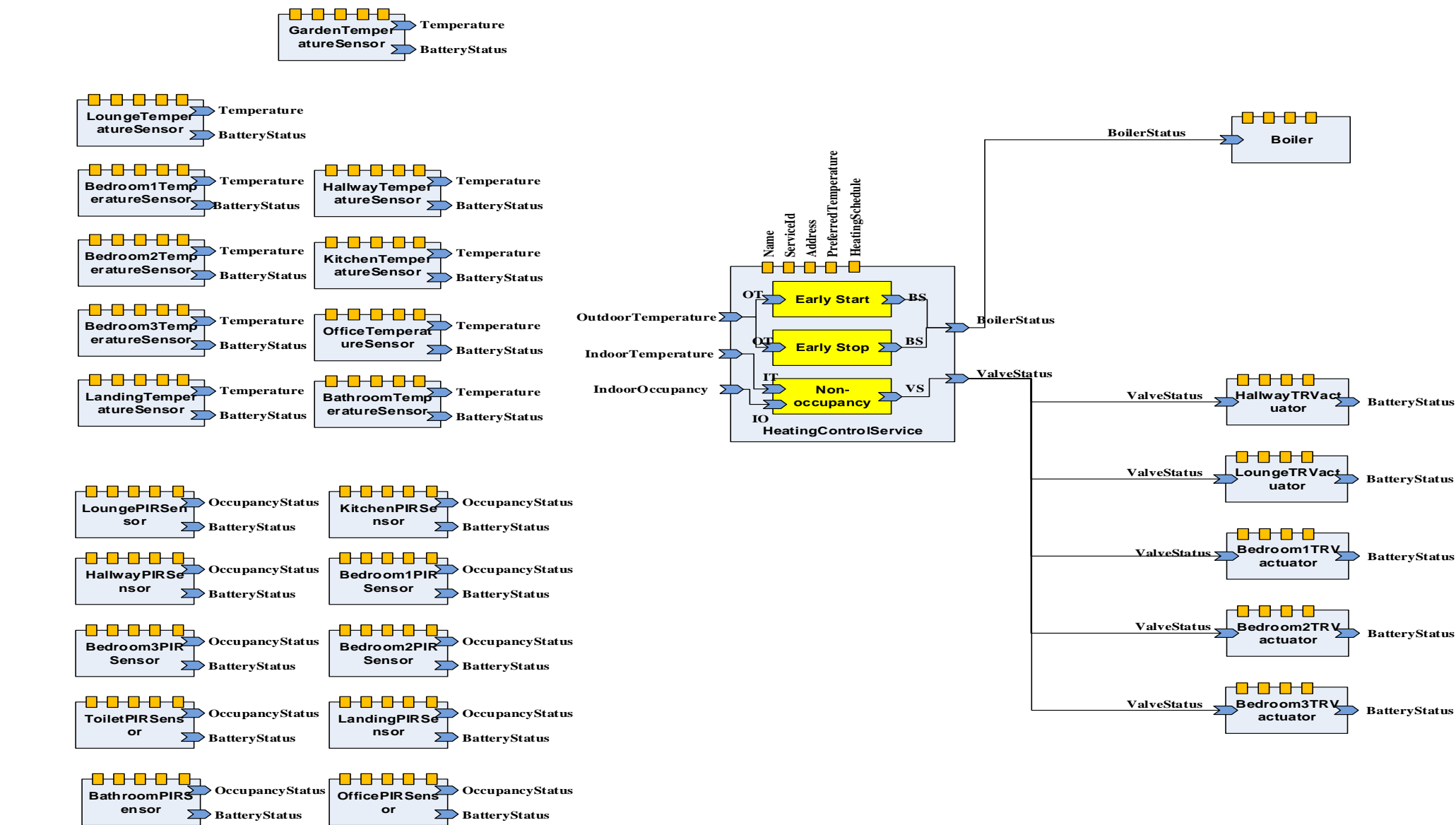


Figure 8.6: Syntax Symbol-based Linkage

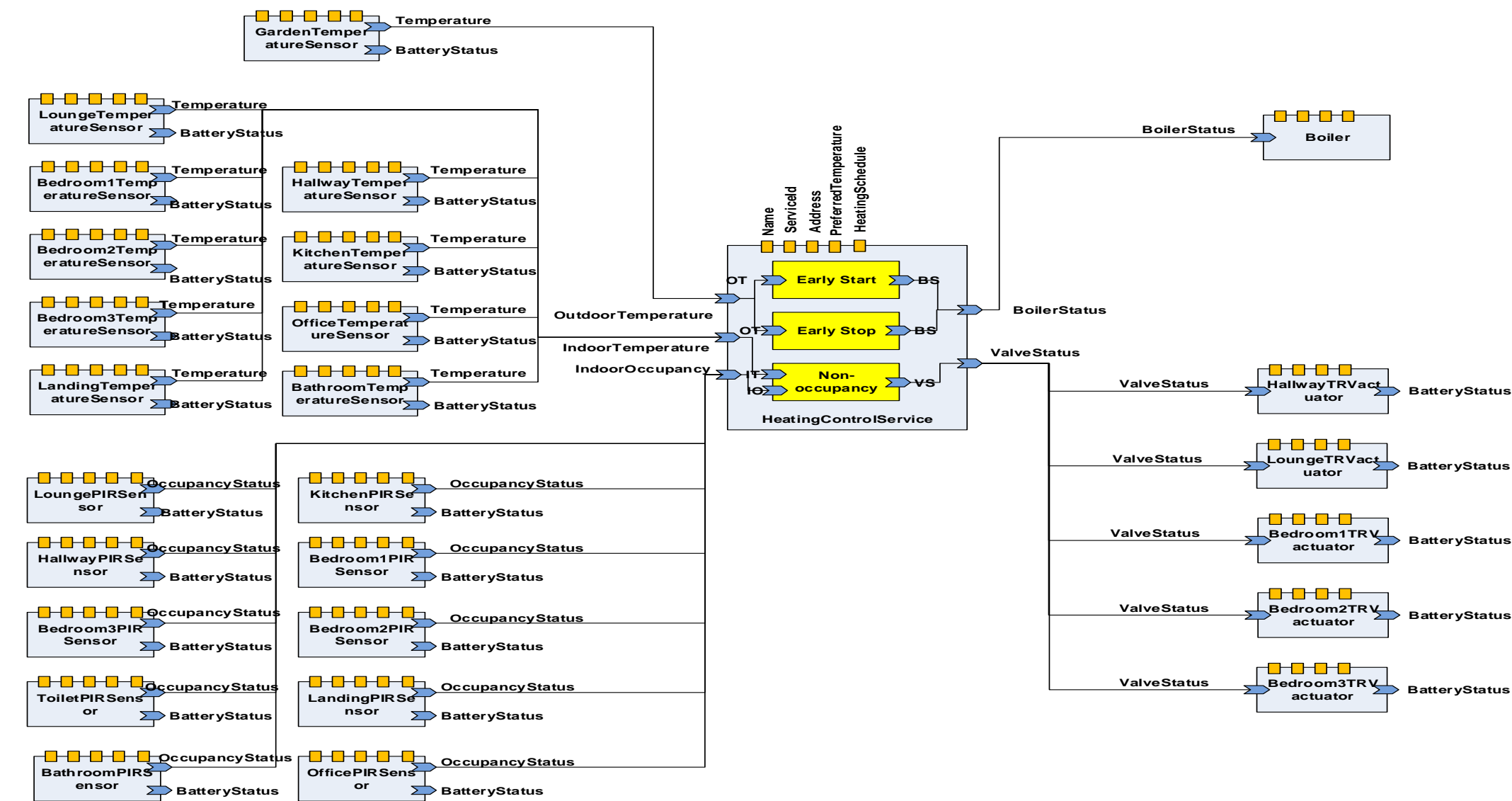


Figure 8.7: Ontology-based Linkage

The SWRL rules have been designed in Protégé. An SWRL rule consists of an antecedent (body) and a consequent (head), forming atoms for particular smart home scenarios. The antecedent of the SWRL rule defines the condition of the reasoning rules, whereas the consequent specifies the action to be taken. Whenever the conditions specified in the antecedent are met, then the actions specified in the consequent must be taken.

Rule 1 is the configuration rule, designed to create a link between two different entities (the service and the required device) within the smart home. In this case study, the goal of this rule is to infer the fact that *GardenTemperatureSensor_DataPort2_Event2* should be connected to *HeatingControlService_DataPort1_Event1* utilizing the object property *connectToEvent*.

Rule 1:

Entity(?en1),Entity(?en2),OutdoorTemperatureMeasurand(?m1),OutdoorTemperatureMeasurand(?m2), hasDataPort(?en1, ?d1), hasDataPort(?en2, ?d2), hasEvent(?d1, ?e1), hasEvent(?d2, ?e2), inform(?e1, ?m1), inform(?e2, ?m2), isInputDataPort(?d1, "1"^^long), isInputDataPort(?d2, "0"^^long), DifferentFrom (?en1, ?en2) → connectToEvent(?e1, ?e2)

Rule 1 reads as: two data port events will be connected together if they fulfil the following criteria:

- They belong to different entities;
- One data port is an input and the other is an output;

- Their measurand is outdoor temperature, which is determined by the instances of the class *inform* (e.g. *Temperature1* and *Temperature2*).

In this example, *HeatingControlService_DataPort1_Event1* and *GardenTemperatureSensor_DataPort2_Event2* belong to different entities (i.e. *HeatingControlService* and *GardenTemperatureSensor* respectively). *GardenTemperatureSensor* is the input, while *HeatingControlService* is the output, which satisfies the first two criteria. The measurand of *HeatingControlService_DataPort1_Event1* (i.e. *Temperature1*; Table 8.2), which is an instance of *OutdoorTemperatureMeasurand*, satisfies the third criterion. The concept of *OutdoorTemperatureMeasurand* is not obvious to *GardenTemperatureSensor_DataPort2_Event2*, despite the fact that the measurand of *GardenTemperatureSensor_DataPort2_Event2* (i.e. *Temperature2*; Table 8.2) is an instance of *TemperatureMeasurand*. This is because *OutdoorTemperatureMeasurand* is a new concept introduced by *HeatingControlService* and unknown to the *GardenTemperatureSensor* (or *Temperature2*).

The concept of *OutdoorTemperatureMeasurand* is defined as: a *TemperatureMeasurand* which *hasParameter* particularly *LocationParameter* (i.e. *hasLocation* only *OutdoorArea*). To determine whether *Temperature2* belongs to *OutdoorTemperatureMeasurand* or not, its *LocationParameter* should be specified. If its *LocationParameter hasLocation* belongs to *OutdoorArea*, it then belongs to *OutdoorTemperatureMeasurand*.

However, only the *GardenTemperatureSensor* entity has the attribute parameter *GardenTemperatureLocation* which is *LocationParameter*. This *LocationParameter* needs to be propagated to lower levels in order to help the Sobot to create a connection between the service and the required device within the smart home system. It is necessary to propagate the parameter into the low level artefacts (e.g. data port, informing measurement), since the device entity and service entity host the parameter. Thus, the first step towards achieving auto-configuration is to propagate the parameter by executing Rules 2-4.

Rule 2:

Entity(?en), hasDataPort(?en, ?d), hasParameter(?en, ?p) → hasParameter(?d, ?p)

This rule is designed to propagate the *DataPort* parameter. It reads as: if an entity (en) has a data port (d) and a parameter (p), then the data port of this entity will inherit the parameter (p). An *Entity* is an OWL description (class name), whereas *hasDataPort* and *hasParameter* are OWL properties. The expressions (?en), (?d) and (?p) are used to represent either variables, OWL individuals (instances) or OWL data values, as appropriate.

To validate Rule 2, it has been populated with some individuals from the real case study (Tables 8.2 and 8.3). For example, *GardenTemperatureSensor* represents the OWL individual (en). *GardenTemperatureSensor* has the OWL DataPort individual (d), which in this case is *GardenTemperatureSensor_DataPort2*. *GardenTemperatureLocation* represents the OWL parameter (p). This reasoning rule implies that the action which

should be taken is that *GardenTemperatureSensor_Dataport2* has parameter *GardenTemperatureLocation*.

Rule 3:

$\text{DataPort}(?d), \text{hasEvent}(?d, ?e), \text{hasParameter}(?d, ?p) \rightarrow \text{hasParameter}(?e, ?p)$

This rule reads as: if the data port (d) has an event (e) and a parameter (p), then the event of this data port will inherit the parameter (p). To validate this rule, it has been populated with some individuals from the real case study (Tables 8.2 and 8.3). *GardenTemperatureSensor_Dataport2* is an individual of the class DataPort (d). This individual has *GardenTemperatureSensor_Dataport2_Event2* as an event (e) and *GardenTemperatureLocation* as a parameter (p). This reasoning rule implies that the action which should be taken is that *GardenTemperatureSensor_Dataport2_Event2* has parameter *GardenTemperatureLocation*.

Rule 4:

$\text{Event}(?e), \text{hasParameter}(?e, ?p), \text{inform}(?e, ?m) \rightarrow \text{hasParameter}(?m, ?p)$

This rule reads as: if the event (e) has a parameter (p) and an inform (m), then the measurand of this event will have the parameter (p). To validate Rule 4, it has been populated with some individuals from the real case study (Tables 8.2 and 8.3). *GardenTemperatureSensor_Dataport2_Event2* is an individual of Event (e) which has parameter *GardenTemperatureLocation* and *inform* of measurand *Temperature2*. This reasoning rule implies that the action which should be taken is that the measurand *Temperature2* has parameter *GardenTemperatureLocation*.

Executing the parameter propagation Rules 2, 3 and 4 together by Pellet successively resulted in inferring two individuals belonging to the class `OutdoorTemperatureMeasurand`, as shown in Figure 8.8, from the existing individuals (Tables 8.2 and 8.3). The Sobot inferred that two facts belonging to `OutdoorTemperatureMeasurand` are *Temperature 1*, and *Temperature 2*.

As shown in Figure 8.8, *Temperature2* is inferred as an instance of `OutdoorTemperatureMeasurand`, since *BackGarden* is an instance of *Garden*, which is a subclass of *OutdoorArea*. The inference of *Temperature2* as an instance of `OutdoorTemperatureMeasurand` means that *Temperature2*, which is the measurand of *GardenTemperatureSensor*, is now obvious to the `OutdoorTemperatureMeasurand` concept, which was first introduced by *HeatingControlService* and unknown to *GardenTemperatureSensor* or *Temperature2*.

The purple diamonds under the OWL class denote its OWL individuals. The number shown between brackets next to the class name indicates the number of such individuals.

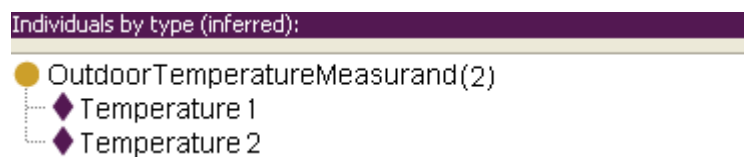


Figure 8. 8: The Result of Executing the Parameter Propagation Rules

This satisfies the third criterion, hence the configuration Rule 1 is fulfilled. Through semantic reasoning, the Sobot was able to infer that the temperature sensor located in the back garden measured the outdoor temperature, so that *GardenTemperatureSensor*

was connected to *HeatingControlService*. Thus the configuration goal was achieved. The Sobot's configuration process will be further elaborated in Section 8.6.

The Sobot's inference of these individuals has resolved the challenge of the syntactic match between the service interface and device interface within the smart home environment, shown in the functional block presentations. These OWL inferred individuals are then stored in the smart home ontology to form the knowledgebase, which helps the Sobot to achieve auto-configuration of the smart home system by linking the smart home service with the required device.

8.6 Results, Performance Metrics and Evaluation Criteria

This section reports the evaluation of the proposed approach, based on several requirements of the smart home environment. These evaluations are generally used to highlight critical issues in the software application and its weaknesses, as well as to discuss how it can be improved.

The evaluation criteria focus on the capabilities of the proposed approach to meet the requirements of the smart home environment and to cope with its dynamicity, thus fulfilling its users' needs. It is expected that the key gains from auto-configuration will be in the form of enhanced performance and reductions in operating costs. Other benefits of auto-configuration could include enhancing the whole system's work capacity and improving its performance and service quality, due to better adaptation to the characteristics of the dynamic system. Thus, the following criteria were used to evaluate the Sobot's effectiveness in meeting the requirements of the smart home environment.

- **Adaptiveness:** The Sobot needs to be able to interpret the changes it detects and extract the relevant contextual data. It needs to validate any new information or knowledge and encode this data into an appropriate format. By utilising available knowledge locally or remotely, it must be able to reason a configuration plan based on the new information. According to its context, the Sobot should be able to change its behaviour, i.e. changing its configuration and functions.
- **Flexibility:** The Sobot should be flexible enough to deal elegantly with the complexity of the conditions (i.e. introducing new concepts) in the smart home environment. It must efficiently infer new facts about more complex situations to meet the system's goal, based on the smart home ontological model, so that the system can adapt to the changed environment. It should be able to avoid contaminating the knowledgebase, by isolating and eliminating any inconsistent situational information.
- **No need for direct user intervention:** The Sobot needs to auto-configure efficiently and generate a proper linkage between the devices and services required by the smart home system, with no need for direct user intervention, in order not to disturb the operation of the system.

The evaluation of the capabilities of the proposed Sobot in relation to the above-mentioned criteria, to meet the requirements of the smart home environment, is illustrated as follows:

The Sobot's inference of two individuals belonging to `OutdoorTemperatureMeasure` and (as shown in Figure 8.8) from the existing ones helped the Sobot to determine

automatically the configuration individuals which should be selected to achieve the configuration of the smart home system with no need for direct user intervention. Through reasoning, the Sobot was able to infer that the temperature sensor located in the back garden measured the outdoor temperature, so that *GardenTemperatureSensor* was connected to *HeatingControlService*.

Figure 8.9 shows the result of the whole Sobot configuration process connecting *GardenTemperatureSensor* to *HeatingControlService* by utilizing the *ConnectToEvent* object property. It shows that the object property *hasParameter* connects the individuals *GardenTemperatureSensor* and *GardenTemperatureLocation*, while the object property *hasLocation* connects the individuals *GardenTemperatureLocation* and *Back Garden*.

Regarding the connection of the device-related individuals, the object property *hasDataPort* connects the individuals *GardenTemperatureSensor* and *GardenTemperatureSensor_Dataport2*. Next, the object property *hasEvent* connects the individuals *GardenTemperatureSensor_Dataport2* and *GardenTemperatureSensor_Dataport2_Event2*. Finally, the object property *ConnectToEvent* connects the individuals *GardenTemperatureSensor_Dataport2_Event2* and *HeatingControlService_Dataport1_Event1*.

Regarding the connection of the service-related individuals, the object property *hasDataPort* connects *HeatingControlService* and *HeatingControlService_Dataport1*. Next, the object property *hasEvent* connects *HeatingControlService_Dataport1* and *HeatingControlService_Dataport1_Event1*. Finally, the object property

ConnectToEvent connects the individuals *HeatingControlService_Dataport1_Event1* and *GardenTemperatureSensor_Dataport2_Event2*.

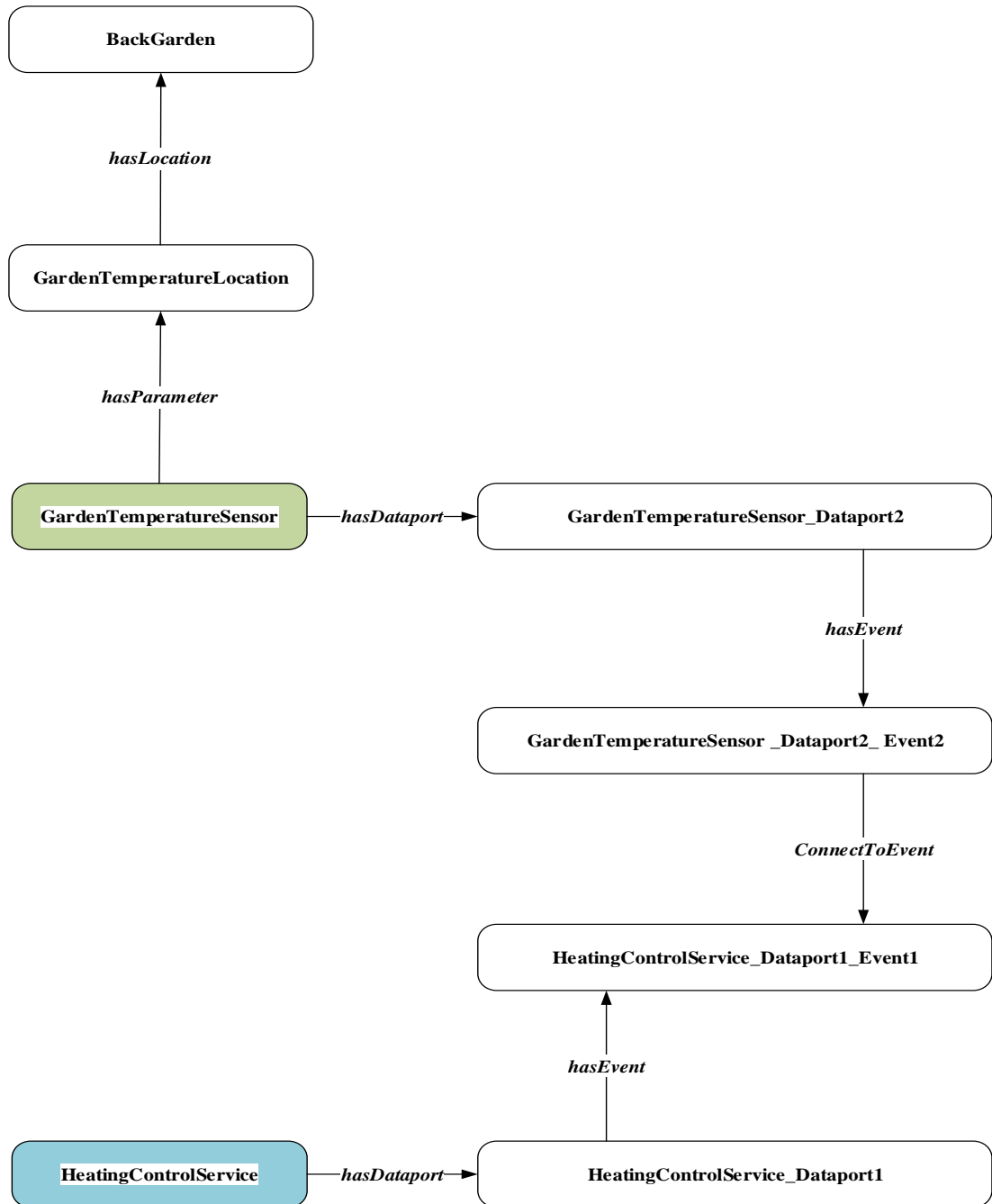


Figure 8. 9: The Result of the Sobot Configuration Process – GardenTemperatureSensor is connected to HeatingControlService

The Sobot has successfully achieved the configuration goal. *HeatingControlService* is connected to *GardenTemperatureSensor* by the Sobot's creation of a connection between *GardenTemperatureSensor_DataPort2_Event2* and *HeatingControlService_DataPort1_Event1* via the object property *connectToEvent*. This connection satisfies Rule 1. In other words, the three configuration criteria of Rule 1 have been fulfilled. First, the two data port events have been connected as they belong to different entities, i.e. *GardenTemperatureSensor* and *HeatingControlService*. Secondly, *GardenTemperatureSensor_DataPort1* is an input port and *HeatingControlService_DataPort2* is an output port. Finally, their measurand is outdoor temperature, which is determined by the instances of the class *inform* (e.g. *Temperature1* and *Temperature2*). *Temperature1* is the measurand of *HeatingControlService* and *Temperature2* is the measurand of *GardenTemperatureSensor*, as shown in Table 8.2.

The connection achieved between the *HeatingControlService* service and the related device *GardenTemperatureSensor* is the feedback and the evidence that proves the viability of the Sobot to achieve an automatic configuration of smart home services and devices. The intelligence of the Sobot in utilizing knowledge representation (the ontology and the Semantic Web Rule Language) facilitates its inferencing from existing knowledge elements to create new elements of knowledge (connections). The result shows the Sobot's ability to perform a series of logical connection activities between the required individuals, leading to a logical connection between the two entities (service and device), based on the available knowledgebase.

This shows the Sobot's awareness, as it successfully acquired knowledge about the environment, including the available connected resources residing in the home service gateway and their status. It was also adaptive in the sense that it was able to change the current status into a new and appropriate one as required. The Sobot behaved autonomously and proactively. It is noteworthy that it dynamically automated the activities of customised service delivery in relation to system operation and behaviour for dynamic home environments, without relying on direct user intervention.

Overall, the evaluation of the Sobot can be summarized as follows. The fact that the Sobot could successfully and efficiently configure a proper linkage between service and device provided by different suppliers based on the ontological model of the smart home, in a dynamic smart home environment, without relying on direct user intervention or causing disruption to the system, means that it was aware, flexible enough, adaptive and could produce a viable configuration to satisfy the given requirements.

The Sobot has demonstrated the potential to be a well-structured approach to auto-configuration, fulfilling the smart home requirements and helping to make the whole system more agile. Such auto-configuration ensures the smooth, seamless updating and functioning of the smart home, providing the occupants with a calm home environment. This in turn enhances the quality of the built smart home environment, satisfies users' needs and reduces overall system cost.

8.7 Summary

This chapter has presented a validation of the auto-configuration Sobot within the smart home environment, utilizing a testbed developed by the Mechatronics Research Group. The results of the test case, conducted to evaluate the effectiveness of the Sobot in identifying and responding to configuration of smart home systems, prove its viability to achieve an automatic configuration between smart home services and devices. It has been shown that all of the Sobot's functional units worked together to identify a particular configuration and responded automatically within the smart home environment. The Sobot performed a series of logical connection activities between the required individuals, leading to a logical connection between two entities (a service and a device), utilizing the smart home knowledgebase model.

Based on these results, the Sobot is evaluated as having met the performance metrics of adaptiveness, flexibility and online configuration of the smart home environment without direct user intervention, as detailed below.

Adaptiveness and flexibility: The Sobot has been found flexible and adaptive in the sense that it has dealt elegantly with the complexity of smart home environmental conditions and flexibly performed in more dynamic situations to meet its goals without disruption. It detected and interpreted changes in the relevant contextual data within the smart home environment, including new devices or services joining the environment, existing ones leaving it, and the context of any device or service changes. It validated new information or knowledge and encoded this data into an appropriate format. By utilising available knowledge locally or remotely, the Sobot was able to reason a

configuration plan based on the new information. It efficiently inferred new facts about more complex situations to meet the system's goal, based on the smart home ontological model, so that the system became adaptable to the changed environment. The Sobot automatically generated a proper linkage between the services and the required devices (i.e. GardenTemperatureSensor is connected to HeatingControlService by utilizing the ConnectToEvent object property) by automating the activities associated with customised service delivery in dynamic home environments. The Sobot was able to avoid contaminating the knowledgebase, by isolating and eliminating any inconsistent situational information. Overall, these facts demonstrate the Sobot's awareness of the digital and physical environment.

Online configuration: The Sobot executed the configuration online, while the system was working, and did not halt the performance of the whole smart home system. The system continued running even when the configuration of any changes in the smart home environment was applied.

No need for direct user intervention: The Sobot executed auto-configuration efficiently and generated a proper linkage between the devices and services without direct user intervention, so that the operation of the smart home system was not disturbed. This helped when the configuration needed was beyond users' capabilities, making life easier for them.

Chapter 9: Conclusion and Recommendations

9.1 Research Summary

The aim of this research study was to devise a novel approach capable of accurately and automatically configuring smart home systems. The main benefit of auto-configuration is that it promotes the evolution of such systems towards the provision of a more comfortable home environment. The literature review indicates that auto-configuration within the smart home environment has generally proven to be a challenging task. The challenge arises from unanticipated changes to external factors such as the status of physical devices and user's preferences, which can significantly affect the system's behaviour throughout its lifecycle. This requires considerable effort to maintain such complex systems. These challenges have stimulated the need for dynamic auto-configurable services amongst such distributed systems.

However, an evaluation of the state-of-the-art approaches reveals no satisfactory solutions to achieve auto-configuration and respond to system dynamics without direct user intervention, thus maintaining the quality of the smart home. These solutions are deemed incomplete, as they lack the ability to meet the requirements of the smart home environment, such as flexibility and avoiding the need for direct user intervention. Consequently, it is necessary to develop an efficient, agile and flexible mechanism that adapts to the new and dynamic requirements of the smart home environment without user intervention.

This thesis has proposed a novel approach named the *Knowledge-based Auto-configuration Software Robot (Sobot) for the Smart Home Environment*, which combines the advantages of recent advances in ubiquitous robotics and SW technologies to embed auto-configuration techniques into smart home environments. The essential feature of the approach is the formal description of knowledge, which allows machine understanding and processing of this knowledge within the system. A semantic model was developed and employed by an autonomic system for making intelligent decisions within smart home environments.

Ontology was utilized as a key component of the proposed approach, allowing the auto-configuration Sobot to function within a smart home system. It was specifically used as a framework to describe the various elements of the smart home environment. Such an ontology facilitates the sharing and reuse of acquired knowledge between human and computer agents to achieve semantic understanding. The knowledgebase approach was developed based on OWL and SWRL.

This new approach assigns a Sobot to automate the activities of customized service delivery for dynamic home environments. The prototype Sobot was implemented on the OSGi framework, which reduces management complexity by offering a modular runtime environment for both large distributed systems and small embedded applications. The Sobot and its components, all of which are interdependent, were developed as a bundle in the OSGi framework to transform the home environment into an intelligent system.

The case study has demonstrated the Sobot's validity and efficiency in identifying and responding to the configuration of smart home systems. The proposed approach thus contributes to the design and implementation of smart home systems. It enables services to utilise the required devices seamlessly through any network within the smart home environment, anywhere and at anytime. The intelligence embedded in the Sobot helps to deliver services automatically and cooperatively. It provides the necessary support for the realisation of the smart home concept and deals with the various complex conditions of the home environment, thus satisfying users' needs.

9.2 Research Findings

The test case reported in Chapter 8 has demonstrated the ability of the proposed approach to achieve auto-configuration within the smart home environment. The results show that the Sobot successfully inferred the location parameter of the required temperature sensor (outdoor) through the propagation rules (Rules 2-4, Chapter 8). The Sobot had to identify the location parameter of the required temperature sensor, because it was a dynamic parameter specified by the installer and subject to change. The Sobot

successfully inferred two individuals as belonging to *OutdoorTemperatureMeasurand* (Figure 8.8) from the existing ones. Based on these newly inferred facts, the Sobot automatically determined which individuals should be selected to achieve the configuration of the smart home system. Through reasoning, the Sobot inferred that *HeatingControlService* was connected to *GardenTemperatureSensor* by creating a connection between *GardenTemperatureSensor_DataPort2_Event2* and *HeatingControlService_DataPort1_Event1* via the object property *connectToEvent*. This connection satisfies the configuration rule (Rule 1, Chapter 8). Thus, the Sobot has successfully achieved the goal of configuration.

This demonstrates that the Sobot used the smart home knowledge model and relied on reasoning capabilities to determine the devices needed to perform the operations in the configuration and subsequently the services that controlled these devices. The results also show that the Sobot could successfully and efficiently configure proper linkages between devices and services provided by different suppliers, in a dynamic smart home environment. This demonstrated the Sobot ability to detect, determine and identify suitable smart home resources for executing configuration without causing any intrusion in the environment. It means that the Sobot is aware and adaptive and can produce a viable configuration that satisfies the specified requirements.

The Sobot has proved to be flexible enough to deal elegantly with a dynamic smart home environment comprising assorted devices and services, by efficiently inferring new facts about dynamic situations based on the smart home ontological model. It was also shown to be aware, as it successfully acquired knowledge about the environment including the available connected resources residing in the home service gateway and

their status. Furthermore, it was seen to be adaptive in the sense that it was able to change the current status for a new and appropriate one. The Sobot has been shown to exhibit autonomous and proactive behaviour. It dynamically automated the activities of customised service delivery for dynamic home environments without user intervention, thus satisfying the users' needs.

In conclusion, the Sobot has demonstrated the potential to be a well-structured auto-configuring approach. The research has achieved its aim to automate the activities of customized service delivery for dynamic home environments, overcoming the shortcomings of conventional methods of auto-configuration within smart home environments. Achieving auto-configuration fulfils the requirements of the smart home, thus making the whole system more agile. It ensures the up-to-date, calm and smooth functioning of the smart home, which in turn enhances the quality of the built smart home environment, satisfies users' needs and reduces the overall system cost.

9.3 Contributions to Knowledge

The major contributions to knowledge made by the work reported in this thesis can be summarised as follows:

- A novel Sobot architecture has been designed to realise an auto-configuration system for smart home environments which combines ubiquitous robotics and Semantic Web technology. The novelty of this research study lies in being the first to use ubiquitous robotics to achieve smart home auto-configuration in a way that addresses the gaps in previous related research. It also satisfies the requirements of smart home environments by increasing the agility and flexibility of their systems.

- The proposed auto-configuration Sobot approach is able to auto-configure smart home service applications provided by different suppliers. It creates viable configuration by resolving syntactic and semantic mismatches between devices and the service interface. This helps the Sobot to link the service with the required device by utilising the inferred facts, based on the smart home ontological model. It has the capacity to execute the generated auto-configuration plan without interrupting the running of the system.
- The main advantages of the Sobot over traditional applications are its self-awareness of the changing digital and physical environments and its ability to interpret these changes, extract the relevant contextual data and merge this with any new information. It deals effectively with the complexity of smart home environments and satisfies their users' needs, thus helping to improve their performance and specification. The configuration Sobot could help to minimise the overheads of the current configuration process within the smart home environment by automating it.
- A new semantic model, which can be classified as new knowledge, has been created and can be utilised by an ubiquitous robotics system for making intelligent decisions within smart home environments. In particular, the generic ontological description of smart home service ontology covers diverse applications. It can efficiently manage the whole smart home system, since it represents all of its services, resources, conditions and properties. According to the identified properties, the ontology facilitates the identification of services and the automation of their delivery in the smart home.

This approach overcomes the shortcomings of conventional methods of auto-configuration within smart home environments. It has the potential to impact on other remote service delivery areas, such as safety and telehealth in intelligent environments, through ad hoc collaboration, thus enhancing the quality of the built environment. These remote services might benefit from the new semantic model of a smart home created in this thesis, rather than creating new models from scratch.

The newly developed ontology reported in this thesis is customized for the smart home domain, in particular an auto-configuring system. The model of service and device functionalities and their related concepts have facilitated the auto-configuration of a smart home environment. Thus, the modelling of concepts such as environment and user preference could be involved. This work therefore contributes to the filling of a gap in the literature on service configuration.

9.4 Recommendations for Future Work

The area of auto-configuration is broad and still largely unexplored. This research study has investigated some issues within this area and envisages some other suitable avenues of investigation. The following paragraphs consider possible improvements to the present approach and the ramifications of this study, which are expected to prove particularly significant for future research into auto-configuration.

- Further work is specifically needed to improve the auto-configuration Sobot prototype, particularly its architecture, decision-making ability and analytical functionalities. The architecture could be improved by integrating more technologies to build more robust and efficient systems to automate smart home services. The

unified framework of elements, descriptions and configuration models could be improved to encompass the semantic richness of these elements and tasks. In particular, increasing the complexity of the inference rules will help the Sobot to achieve the configuration of more complex systems.

- The ontology could also be enhanced to better represent the features and specifications of more complex smart environments and more advanced telecare systems. This ontology would permit the identification of more complex relations amongst services and devices operating in a smart environment, or related to interference with existing devices and user profiles.
- Further scenarios should be investigated to evaluate the applicability of the proposed approach to certain areas such as the safety and security of smart home environments, utilising the ontological models created here and the individuals of the smart home devices and services (Table 8.2). For example, when the house is unoccupied, both the “away from home” security service feature and the energy management service are active. However, these services have incompatible goals, since the energy management service will turn the lights and the TV off to save energy, while the security service will switch them on to give the impression that the home is occupied. The conflicting requirements of these two services mean that they are incompatible when attempting to control smart home appliances simultaneously.

References

- [1] R. Tafazolli, C. Upstill, H. Aghvarni, R. Cooper, and W. Dutton, "A Roadmap for Interdisciplinary Research on the Internet of Things," 2012.
- [2] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé, "Vision and challenges for realising the Internet of Things," *Cluster of European Research Projects on the Internet of Things, European Commission*, 2010.
- [3] D. Evans, "The Internet of Things: How the next evolution of the internet is changing everything," *CISCO white paper*, 2011.
- [4] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, pp. 2787-2805, 2010.
- [5] S. De, B. Christophe, and K. Moessner, "Semantic Enablers for Dynamic Digital-Physical Object Associations in a Federated Node Architecture for the Internet of Things," *Ad Hoc Networks*, 2013.
- [6] C. Reinisch, M. J. Kofler, and W. Kastner, "ThinkHome: A smart home as digital ecosystem," in *Digital Ecosystems and Technologies (DEST), 2010 4th IEEE International Conference on*, 2010, pp. 256-261.
- [7] H. Nakashima, H. K. Aghajan, and J. C. Augusto, *Handbook of ambient intelligence and smart environments*: Springer, 2010.
- [8] B. Van der Vlist, G. Niezen, J. Hu, and L. Feijs, "Design semantics of connections in a smart home environment," *Design and Semantics of Form and Movement (DeSForM 2010), Lucerne, Switzerland*, pp. 48-56, 2010.
- [9] D. J. Cook, "How smart is your home," *Science*, vol. 335, pp. 1579-1581, 2012.
- [10] N. Balta-Ozkan, R. Davidson, M. Bicket, and L. Whitmarsh, "The development of smart homes market in the UK," *Energy*, 2013.
- [11] X. Zhu, Y. Yu, Y. Ou, D. Luo, C. Zhang, and J. Chen, "System Modeling of a Smart-Home Healthy Lifestyle Assistant," in *Agents and Data Mining Interaction*, ed: Springer, 2013, pp. 65-78.

- [12] P. Menschner, A. Prinz, P. Koene, F. Köbler, M. Altmann, H. Krcmar, *et al.*, "Reaching into patients' homes—participatory designed AAL services," *Electronic Markets*, vol. 21, pp. 63-76, 2011.
- [13] H.-I. Yang and A. Helal, "Safety enhancing mechanisms for pervasive computing systems in intelligent environments," in *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, 2008, pp. 525-530.
- [14] C. Fabbriatore, H. Boley, and A. P. Karduck, "Machine learning for resource management in smart environments," in *Digital Ecosystems Technologies (DEST), 2012 6th IEEE International Conference on*, 2012, pp. 1-6.
- [15] A. Dohr, R. Modre-Opsrian, M. Drobics, D. Hayn, and G. Schreier, "The internet of things for ambient assisted living," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, 2010, pp. 804-809.
- [16] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, pp. 1497-1516, 2012.
- [17] A. Valls, K. Gibert, D. Sánchez, and M. Batet, "Using ontologies for structuring organizational knowledge in Home Care assistance," *international journal of medical informatics*, vol. 79, pp. 370-387, 2010.
- [18] J. Paradiso, P. Dutta, H. Gellersen, and E. Schooler, "Guest Editors' Introduction: Smart Energy Systems," *Pervasive Computing, IEEE*, vol. 10, pp. 11-12, 2011.
- [19] M. N. Kamel Boulos, R. C. Lou, A. Anastasiou, C. D. Nugent, J. Alexandersson, G. Zimmermann, *et al.*, "Connectivity for healthcare and well-being management: examples from six European projects," *International journal of environmental research and public health*, vol. 6, pp. 1947-1971, 2009.
- [20] H. Boley and E. Chang, "Digital ecosystems: Principles and semantics," 2007.
- [21] C. Reinisch, M. J. Kofler, F. Iglesias, and W. Kastner, "Thinkhome energy efficiency in future smart homes," *EURASIP Journal on Embedded Systems*, vol. 2011, p. 1, 2011.
- [22] D. Retkowitz and S. Kulle, "Dependency management in smart homes," in *Distributed Applications and Interoperable Systems*, 2009, pp. 143-156.
- [23] Z. Etzioni, J. Keeney, R. Brennan, and D. Lewis, "Supporting composite smart home services with semantic fault management," in *Future Information Technology (FutureTech), 2010 5th International Conference on*, 2010, pp. 1-8.

- [24] G. Bekey and J. Yuh, "The status of robotics," *Robotics & Automation Magazine, IEEE*, vol. 15, pp. 80-86, 2008.
- [25] W. Feng, "Remote service provision for connected homes," 2010.
- [26] M. Arndt, S. Wille, L. de Souza, V. F. Rey, N. Wehn, and K. Berns, "Performance evaluation of ambient services by combining robotic frameworks and a smart environment platform," *Robotics and Autonomous Systems*, 2013.
- [27] K. Balasubramanian and A. Cellatoglu, "Improvements in home automation strategies for designing apparatus for efficient smart home," *Consumer Electronics, IEEE Transactions on*, vol. 54, pp. 1681-1687, 2008.
- [28] M. Sagi, D. Mijic, D. Milinkov, and B. Bogovac, "Smart home automation," in *Telecommunications Forum (TELFOR), 2012 20th*, 2012, pp. 1512-1515.
- [29] T. Perumal, A. R. Ramli, C. Y. Leong, S. Mansor, and K. Samsudin, "Interoperability for smart home environment using web services," *International Journal of Smart Home*, vol. 2, pp. 1-16, 2008.
- [30] Y. Alsafi and V. Vyatkin, "Ontology-based reconfiguration agent for intelligent mechatronic systems in flexible manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 26, pp. 381-391, 2010.
- [31] J. Kim, H.-s. Choi, H. Wang, N. Agoulmine, M. J. Deerv, and J. W.-K. Hong, "POSTECH's U-Health Smart Home for elderly monitoring and support," in *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a*, 2010, pp. 1-6.
- [32] Z. Hu, G. Privat, S. Frénot, and B. Tourancheau, "Representation and self-configuration of physical entities in extended smart grid perimeter," in *Innovative Smart Grid Technologies (ISGT Europe), 2012 3rd IEEE PES International Conference and Exhibition on*, 2012, pp. 1-7.
- [33] A. Kuutti, A. Dvoryanchikova, A. Lobov, J. L. M. Lastra, and T. Vantera, "A device configuration management tool for context-aware system," in *Industrial Informatics (INDIN), 2012 10th IEEE International Conference on*, 2012, pp. 10-15.
- [34] J. Shen, L. Wang, and Y. Sun, "Configuration of product extension services in servitisation using an ontology-based approach," *International Journal of Production Research*, vol. 50, pp. 6469-6488, 2012.
- [35] M. Horridge, D. Tsarkov, and T. Redmond, "Supporting Early Adoption of OWL 1.1 with Protege-OWL and FaCT++," in *OWLED*, 2006.
- [36] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, *et al.*, "The evolution of Protégé: an environment for knowledge-based

systems development," *International Journal of Human-computer studies*, vol. 58, pp. 89-123, 2003.

- [37] H. Knublauch, R. W. Ferguson, N. F. Noy, and M. A. Musen, "The Protégé OWL plugin: An open development environment for semantic web applications," in *The Semantic Web-ISWC 2004*, ed: Springer, 2004, pp. 229-243.
- [38] M. d'Aquin and N. F. Noy, "Where to publish and find ontologies? A survey of ontology libraries," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 11, pp. 96-111, 2012.
- [39] Y.-G. Cheong, Y.-J. Kim, S. Y. Yoo, H. Lee, S. Lee, S. C. Chae, *et al.*, "An ontology-based reasoning approach towards energy-aware smart homes," in *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, 2011, pp. 850-854.
- [40] S. Zhang, P. McCullagh, C. Nugent, H. Zheng, and N. Black, "An ontological framework for activity monitoring and reminder reasoning in an assisted environment," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-12, 2013.
- [41] N. Lasierra, A. Alesanco, D. O'sullivan, and J. García, "An autonomic ontology-based approach to manage information in home-based scenarios: From theory to practice," *Data & Knowledge Engineering*, vol. 87, pp. 185-205, 2013.
- [42] D. J. Cook, "Learning setting-generalized activity models for smart spaces," *IEEE intelligent systems*, vol. 2010, p. 1, 2010.
- [43] L. Cao, G. Weiss, and S. Y. Philip, "A brief introduction to agent mining," *Autonomous Agents and Multi-Agent Systems*, vol. 25, pp. 419-424, 2012.
- [44] L. Cao, *Data mining and multi-agent integration [electronic resource]*: Springer, 2009.
- [45] S. Dengler, A. Awad, and F. Dressler, "Sensor/actuator networks in smart homes for supporting elderly and handicapped people," in *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, 2007, pp. 863-868.
- [46] V. Riquebourg, D. Menga, D. Durand, B. Marhic, L. Delahoche, and C. Loge, "The smart home concept: our immediate future," in *E-Learning in Industrial Electronics, 2006 1ST IEEE International Conference on*, 2006, pp. 23-28.
- [47] F. K. Aldrich, "Smart homes: past, present and future," *Inside the smart home*, vol. 1, pp. 17-39, 2003.

- [48] S. Schmid, M. Sifalakis, and D. Hutchison, "Towards autonomic networks," in *Autonomic Networking*, ed: Springer, 2006, pp. 1-11.
- [49] H. Lee and J. Kwon, "The Smart Home Service System Architecture for Healthy and Safe Human Life," 2013.
- [50] S. Brownsell, D. Bradley, S. Blackburn, F. Cardinaux, and M. S. Hawley, "A systematic review of lifestyle monitoring technologies," *Journal of telemedicine and telecare*, vol. 17, pp. 185-189, 2011.
- [51] E. Aarts and J. Encarnação, "The Emergence of Ambient Intelligence," ed: Berlin, Germany: Springer, 2006.
- [52] C. Ramos, J. C. Augusto, and D. Shapiro, "Ambient intelligence—The next step for artificial intelligence," *Intelligent Systems, IEEE*, vol. 23, pp. 15-18, 2008.
- [53] D. J. Cook, J. C. Augusto, and V. R. Jakkula, "Ambient intelligence: Technologies, applications, and opportunities," *Pervasive and Mobile Computing*, vol. 5, pp. 277-298, 2009.
- [54] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 272, pp. 78-89, 1995.
- [55] A. Kameas, "Towards the next generation of ambient intelligent environments," in *Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE), 2010 19th IEEE International Workshop on*, 2010, pp. 1-6.
- [56] F. Viani, F. Robol, A. Polo, P. Rocca, G. Oliveri, and A. Massa, "Wireless Architectures for Heterogeneous Sensing in Smart Home Applications: Concepts and Real Implementation," 2013.
- [57] I. Khan, A. Mahmood, N. Javaid, S. Razzaq, R. Khan, and M. Ilahi, "Home Energy Management Systems in Future Smart Grids," *arXiv preprint arXiv:1306.1137*, 2013.
- [58] M. Erol-Kantarci and H. T. Mouftah, "Wireless sensor networks for domestic energy management in smart grids," in *Communications (QBSC), 2010 25th Biennial Symposium on*, 2010, pp. 63-66.
- [59] H. Farhangi, "The path of the smart grid," *Power and Energy Magazine, IEEE*, vol. 8, pp. 18-28, 2010.
- [60] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, *et al.*, "Smart Grid and Smart Homes: Key Players and Pilot Projects," *Industrial Electronics Magazine, IEEE*, vol. 6, pp. 18-34, 2012.

- [61] S. Ahmad, "Smart metering and home automation solutions for the next decade," in *Emerging Trends in Networks and Computer Communications (ETNCC), 2011 International Conference on*, 2011, pp. 200-204.
- [62] H. Khurana, M. Hadley, N. Lu, and D. A. Frincke, "Smart-grid security issues," *Security & Privacy, IEEE*, vol. 8, pp. 81-85, 2010.
- [63] D. Niyato, L. Xiao, and P. Wang, "Machine-to-machine communications for home energy management system in smart grid," *Communications Magazine, IEEE*, vol. 49, pp. 53-59, 2011.
- [64] J. Fiala, P. Bingger, D. Ruh, K. Foerster, C. Heilmann, F. Beyersdorf, *et al.*, "An implantable optical blood pressure sensor based on pulse transit time," *Biomedical microdevices*, vol. 15, pp. 73-81, 2013.
- [65] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, pp. 1-12, 2010.
- [66] T. Buracchio, H. H. Dodge, D. Howieson, D. Wasserman, and J. Kaye, "The trajectory of gait speed preceding mild cognitive impairment," *Archives of neurology*, vol. 67, p. 980, 2010.
- [67] A. K. Dey and D. Estrin, "Perspectives on Pervasive Health from Some of the Field's Leading Researchers," *Pervasive Computing, IEEE*, vol. 10, pp. 4-7, 2011.
- [68] S. Brownsell, D. Bradley, F. Cardinaux, and M. Hawley, "Developing a systems and informatics based approach to lifestyle monitoring within eHealth: part I-technology and data management," in *Healthcare Informatics, Imaging and Systems Biology (HISB), 2011 First IEEE International Conference on*, 2011, pp. 264-271.
- [69] D. B. McCombie, A. T. Reisner, and H. H. Asada, "Motion based adaptive calibration of pulse transit time measurements to arterial blood pressure for an autonomous, wearable blood pressure monitor," in *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, 2008, pp. 989-992.
- [70] B. Abdulrazak, S. Giroux, B. Bouchard, H. Pigot, and M. Mokhtari, *Towards Useful Services for Elderly and People with Disabilities: 9th International Conference on Smart Homes and Health Telematics, ICOST 2011, Montreal, Canada, June 20-22, 2011, Proceedings* vol. 6719: Springer, 2011.
- [71] S. Brownsell, D. Bradley, F. Cardinaux, and M. Hawley, "Developing a systems and informatics based approach to lifestyle monitoring within eHealth: part II-

analysis & interpretation," in *Healthcare Informatics, Imaging and Systems Biology (HISB), 2011 First IEEE International Conference on*, 2011, pp. 213-220.

- [72] B. Celler, W. Earnshaw, E. Ilisar, L. Betbeder-Matibet, M. Harris, R. Clark, *et al.*, "Remote monitoring of health status of the elderly at home. A multidisciplinary project on aging at the University of New South Wales," *International journal of bio-medical computing*, vol. 40, pp. 147-155, 1995.
- [73] M. J. Fisk, *Social alarms to telecare: older people's services in transition: The Policy Press*, 2003.
- [74] M. Hogan, "Technophobia amongst older adults in Ireland," 2006.
- [75] S. Brownsell, D. Bradley, R. Bragg, P. Catlin, and J. Carlier, "Do users want telecare and can it be cost-effective," in *[Engineering in Medicine and Biology, 1999. 21st Annual Conf. and the 1999 Annual Fall Meeting of the Biomedical Engineering Soc.] BMES/EMBS Conference, 1999. Proceedings of the First Joint*, 1999, p. 714 vol. 2.
- [76] S. Brownsell and D. Bradley, *Assistive technology and telecare: forging solutions for independent living: The Policy Press*, 2003.
- [77] J. Hanson, J. Percival, H. Aldred, S. Brownsell, and M. Hawley, "Attitudes to telecare among older people, professional care workers and informal carers: a preventative strategy or crisis management?," *Universal Access in the Information Society*, vol. 6, pp. 193-205, 2007.
- [78] G. Demiris, M. J. Rantz, M. A. Aud, K. D. Marek, H. W. Tyrer, M. Skubic, *et al.*, "Older adults' attitudes towards and perceptions of smart home technologies: a pilot study," *Informatics for Health and Social Care*, vol. 29, pp. 87-94, 2004.
- [79] S. Levy, N. Jack, D. Bradley, M. Morison, and M. Swanston, "Perspectives on telecare: the client view," *Journal of telemedicine and telecare*, vol. 9, pp. 156-160, 2003.
- [80] G. Williams, K. Doughty, and D. A. Bradley, "A systems approach to achieving CarerNet-an integrated and intelligent telecare system," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 2, pp. 1-9, 1998.
- [81] S. Brownsell, D. Bradley, R. Bragg, G. WiUiams, P. Catlim , and J. Carlier, "Development of Systems for Next Generation Remote Health Care", in *Journal of Telemedicine & Telecare*, 1999.
- [82] D. J. Cook and S. K. Das, "How smart are our environments? An updated look at the state of the art," *Pervasive and mobile computing*, vol. 3, pp. 53-73, 2007.

- [83] L. J. B. Joseph, N. Andy, M. James, B. Lauren "Internet of Things (IoE). The Internet of Everything Cisco IoE Value Index Study" 2013.
- [84] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer, "Easy living: Technologies for intelligent environments," in *Handheld and ubiquitous computing*, 2000, pp. 12-29.
- [85] M. Gritti, M. Broxvall, and A. Saffiotti, "Reactive self-configuration of an ecology of robots," in *Proc of the ICRA-07 Workshop on Network Robot Systems, Rome, Italy*, 2007.
- [86] G. Bin, "An Ontology-based Programming Platform for smart artefact system. ," *Unpublished MA thesis* vol. Keio University 2009.
- [87] D.-M. Han and J.-H. Lim, "Design and implementation of smart home energy management systems based on zigbee," *Consumer Electronics, IEEE Transactions on*, vol. 56, pp. 1417-1425, 2010.
- [88] G. Mantas, D. Lymberopoulos, and N. Komninos, "Security in Smart Home Environment," *Wireless Technologies for Ambient Assisted Living and Healthcare: Systems and Applications*, p. 170, 2010.
- [89] Y. Rogers, "Moving on from weiser's vision of calm computing: Engaging ubicomp experiences," in *UbiComp 2006: Ubiquitous Computing*, ed: Springer, 2006, pp. 404-421.
- [90] G. D. Abowd, A. F. Bobick, I. A. Essa, E. D. Mynatt, and W. A. Rogers, "The aware home: A living laboratory for technologies for successful aging," in *Proceedings of the AAAI-02 Workshop "Automation as Caregiver*, 2002, pp. 1-7.
- [91] G. Weiss, *Multiagent systems: a modern approach to distributed artificial intelligence*: The MIT press, 1999.
- [92] S. A. Shafer, B. Brumitt, and J. Cadiz, "Interaction issues in context-aware intelligent environments," *Human-Computer Interaction*, vol. 16, pp. 363-378, 2001.
- [93] M. Yim, Y. Zhang, and D. Duff, "Modular robots," *Spectrum, IEEE*, vol. 39, pp. 30-34, 2002.
- [94] E. Şahin and A. Winfield, "Special issue on swarm robotics," *Swarm Intelligence*, vol. 2, pp. 69-72, 2008.
- [95] Y. Chen, C.-N. Chuah, and Q. Zhao, "Network configuration for optimal utilization efficiency of wireless sensor networks," *Ad Hoc Networks*, vol. 6, pp. 92-107, 2008.

- [96] G. Xing, C. Lu, Y. Zhang, Q. Huang, and R. Pless, "Minimum power configuration in wireless sensor networks," in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, 2005, pp. 390-401.
- [97] T. Yoshito, N. Thepvilojanapong, and K. Sezaki, "Autonomous configuration in wireless sensor networks," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 88, pp. 3063-3071, 2005.
- [98] P. Busetta, A. Donà, and M. Nori, "Channeled multicast for group communications," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, 2002, pp. 1280-1287.
- [99] N. Kalra, D. Ferguson, and A. Stentz, "Hoplites: A market-based framework for planned tight coordination in multirobot teams," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 1170-1177.
- [100] Y. Al-Safi and V. Vyatkin, "An ontology-based reconfiguration agent for intelligent mechatronic systems," in *Holonic and Multi-Agent Systems for Manufacturing*, ed: Springer, 2007, pp. 114-126.
- [101] J. A. Johnsen, "A Semantic Web-driven Approach to Self-Configuring Computer Systems," Norwegian University of Science and Technology, 2006.
- [102] T. Gu, H. K. Pung, and D. Q. Zhang, "A service-oriented middleware for building context-aware services," *Journal of Network and computer applications*, vol. 28, pp. 1-18, 2005.
- [103] B. Guo, "An Ontology-based Programming Platform for Smart Artifact Systems," 2009.
- [104] G. T. McKee, D. I. Baker, and P. S. Schenker, "Task-directed configuration of networked robotic agents," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, 2002, pp. 2752-2757.
- [105] L. E. Parker and F. Tang, "Building multirobot coalitions through automated task solution synthesis," *Proceedings of the IEEE*, vol. 94, pp. 1289-1305, 2006.
- [106] R. Lundh, L. Karlsson, and A. Saffiotti, "Dynamic self-configuration of an ecology of robots," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, pp. 3403-3409.
- [107] L. Karlsson, "Conditional progressive planning under uncertainty," in *IJCAI*, 2001, pp. 431-438.

- [108] J. Aurich, N. Wolf, M. Siener, and E. Schweitzer, "Configuration of product-service systems," *Journal of Manufacturing Technology Management*, vol. 20, pp. 591-605, 2009.
- [109] P. Clements, R. Kazman, and M. Klein, *Evaluating software architectures*: Addison-Wesley Reading, 2001.
- [110] J. E. López de Vergara, V. A. Villagrà, C. Fadón, J. M. González, J. A. Lozano, and M. Álvarez-Campana, "An autonomic approach to offer services in OSGi-based home gateways," *Computer Communications*, vol. 31, pp. 3049-3058, 2008.
- [111] M. Rodríguez-Muro and D. Calvanese, "Quest, an OWL 2 QL reasoner for ontology-based data access," *OWLED 2012*, 2012.
- [112] G. Gharbi, M. B. Alaya, C. Diop, and E. Exposito, "AODA: an Autonomic and Ontology-Driven Architecture for service-oriented and event-driven systems," *International Journal of Collaborative Enterprise*, vol. 3, pp. 167-188, 2013.
- [113] T. Heider and T. Kirste, "Smart Environments and Self-Organizing Appliance Ensembles," *Mobile Computing and Ambient Intelligence*, vol. 5181, 2005.
- [114] Y.-G. Ha, J.-C. Sohn, Y.-J. Cho, and H. Yoon, "Towards a ubiquitous robotic companion: Design and implementation of ubiquitous robotic service framework," *ETRI journal*, vol. 27, pp. 666-676, 2005.
- [115] J.-H. Kim, K.-H. Lee, Y.-D. Kim, N. S. Kuppuswamy, and J. Jo, "Ubiquitous robot: A new paradigm for integrated services," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 2853-2858.
- [116] E. Garcia, M. A. Jimenez, P. G. De Santos, and M. Armada, "The evolution of robotics research," *Robotics & Automation Magazine, IEEE*, vol. 14, pp. 90-103, 2007.
- [117] M. Rashid, "Extending a networked robot system to include humans, tiny devices, and everyday objects," Örebro University, 2011.
- [118] S. Sathyakeerthy, M. Di Rocco, F. Pecora, and A. Saffiotti, "Scaling up ubiquitous robotic systems from home to town (and beyond)," in *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, 2013, pp. 107-110.
- [119] H. Lee and J. Kwon, "Ontology Model-based Situation and Socially-Aware Health Care Service in a Smart Home Environment," 2013.
- [120] P. Lin, K. Abney, and G. Bekey, "Robot ethics: Mapping the issues for a mechanized world," *Artificial Intelligence*, vol. 175, pp. 942-949, 2011.

- [121] G. A. Bekey, *Autonomous robots: from biological inspiration to implementation and control*: The MIT Press, 2005.
- [122] J.-H. Kim, Y.-D. Kim, and K.-H. Lee, "The third generation of robotics: Ubiquitous robot," in *Proc of the 2nd Int Conf on Autonomous Robots and Agents*, 2004.
- [123] S. Y. Nof, *Handbook of industrial robotics* vol. 1: John Wiley & Sons, 1999.
- [124] C. Buiu, F. Cazan, and R. Ciurlea, "Developing of a Service Robot to Recognize and Sort Waste," in *Proceedings of the 16th International Conference on Control Systems and Computer Science*, 2007, pp. 22-26.
- [125] M. Weiser, "Some computer science issues in ubiquitous computing," *Communications of the ACM*, vol. 36, pp. 75-84, 1993.
- [126] M. Tenorth, U. Klank, D. Pangercic, and M. Beetz, "Web-enabled robots," *Robotics & Automation Magazine, IEEE*, vol. 18, pp. 58-68, 2011.
- [127] X. Chen, J. Xie, J. Ji, and Z. Sui, "Toward Open Knowledge Enabling for Human-Robot Interaction," *Journal of Human-Robot Interaction*, vol. 1, pp. 100-117, 2012.
- [128] M. Inja, N. Heijne, S. Nugteren, and M. de Waard, "Project ai-the darpa robotics challenge-footloose," *Project Report, Universiteit van Amsterdam*, 2013.
- [129] Y.-H. Kim, S.-H. Cho, S.-H. Choi, and J.-H. Kim, "Software robot in a PDA for human interaction and seamless service," in *Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on*, 2007, pp. 968-973.
- [130] P. Levis and D. Culler, "Maté: A tiny virtual machine for sensor networks," in *ACM Sigplan Notices*, 2002, pp. 85-95.
- [131] S. Hadim and N. Mohamed, "Middleware: Middleware challenges and approaches for wireless sensor networks," *Distributed Systems Online, IEEE*, vol. 7, pp. 1-1, 2006.
- [132] J.-H. Kim, "Ubiquitous robot," in *Computational Intelligence, Theory and Applications*, ed: Springer, 2005, pp. 451-459.
- [133] J.-H. Kim, I.-B. Jeong, I.-W. Park, and K.-H. Lee, "Multi-layer architecture of ubiquitous robot system for integrated services," *International Journal of Social Robotics*, vol. 1, pp. 19-28, 2009.
- [134] N. Jennings and M. Wooldridge, "Software agents," *IEE review*, vol. 42, pp. 17-20, 1996.

- [135] M. Wooldridge, "Agent-based software engineering," *IEE Proceedings-software*, vol. 144, pp. 26-37, 1997.
- [136] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Handheld and ubiquitous computing*, 1999, pp. 304-307.
- [137] M. Kuna, H. Kolaric, I. Bojic, M. Kusek, and G. Jezic, "Android/OSGi-based Machine-to-Machine context-aware system," in *Telecommunications (ConTEL), Proceedings of the 2011 11th International Conference on*, 2011, pp. 95-102.
- [138] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini, "A survey of context data distribution for mobile ubiquitous systems," *ACM Computing Surveys (CSUR)*, vol. 44, p. 24, 2012.
- [139] D. Tennenhouse, "Proactive computing," *Communications of the ACM*, vol. 43, pp. 43-50, 2000.
- [140] T. Salvador and K. Anderson, "Practical considerations of context for context based systems: An example from an ethnographic case study of a man diagnosed with early onset alzheimer's disease," in *UbiComp 2003: Ubiquitous Computing*, 2003, pp. 243-255.
- [141] I. A. Essa, "Ubiquitous sensing for smart and aware environments," *Personal Communications, IEEE*, vol. 7, pp. 47-49, 2000.
- [142] E. C. Epp, "Relationship management: Secure collaboration in a ubiquitous environment," *Pervasive Computing, IEEE*, vol. 2, pp. 62-71, 2003.
- [143] T.-H. Kim, S.-H. Choi, and J.-H. Kim, "Middle layer incorporating software robot and mobile robot," in *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, 2006, pp. 253-259.
- [144] M. Weiser and J. S. Brown, "Designing calm technology," *PowerGrid Journal*, vol. 1, pp. 75-85, 1996.
- [145] M. Hohl, S. Malhotra, S. Van Ransbeeck, and C. Guedes, "Calm Technologies 2.0: Visualising social data as an experience in physical space," *PJIM Parsons Journal For Information Mapping*, vol. 1, 2009.
- [146] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach* vol. 74: Prentice hall Englewood Cliffs, 1995.
- [147] V. Jones and J. H. Jo, "Ubiquitous learning environment: An adaptive teaching system using ubiquitous technology," in *Beyond the comfort zone: Proceedings of the 21st ASCILITE Conference*, 2004, pp. 468-474.

- [148] D. King and K. Jones, "Competitive intelligence, software robots and the Internet: the NewsAlert prototype," in *System Sciences, 1995. Vol. IV. Proceedings of the Twenty-Eighth Hawaii International Conference on*, 1995, pp. 624-631.
- [149] S.-Y. Yoon, R. C. Burke, B. Blumberg, and G. E. Schneider, "Interactive training for synthetic characters," *AAAI/IAAI*, vol. 2000, pp. 249-254, 2000.
- [150] J. H. Moor, "The nature, importance, and difficulty of machine ethics," *Intelligent Systems, IEEE*, vol. 21, pp. 18-21, 2006.
- [151] P. Lin, K. Abney, and G. A. Bekey, *Robot Ethics: The ethical and social implications of robotics*: The MIT Press, 2011.
- [152] G. Veruggio, "The euron roboethics roadmap," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, 2006, pp. 612-617.
- [153] R. Beckwith and S. Lederer, "Designing for one's dotage: UbiComp and Residential Care facilities," in *Conference on the Networked Home and the Home of the Future (HOIT 2003)*, Irvine, CA: April, 2003.
- [154] K. Anderson and P. Dourish, "Situated Privacies: Do you know where you mother [trucker] is," in *Proceedings of the 11th International Conference on Human-Computer Interaction. Las Vegas*, 2005.
- [155] J. H. Moor, "Is ethics computable?," *Metaphilosophy*, vol. 26, pp. 1-21, 1995.
- [156] V. Wiegel, M. Van den Hoven, and G. Lokhorst, "Privacy, deontic epistemic action logic and software agents," *Ethics and Information Technology*, vol. 7, pp. 251-264, 2005.
- [157] J. Gips, "Creating ethical robots: A grand challenge," in *M. Anderson, SL Anderson, & Armen, C.(Co-chairs), AAAI Fall 2005 Symposium on Machine Ethics*, 2005, pp. 1-7.
- [158] J. R. Searle, "Minds, brains, and programs," *Behavioral and brain sciences*, vol. 3, pp. 417-457, 1980.
- [159] J. F. Sowa, *Knowledge representation: logical, philosophical, and computational foundations* vol. 13: MIT Press, 2000.
- [160] P. Rajeswari and T. Prasad, "Hybrid Systems for Knowledge Representation in Artificial Intelligence," *CoRR*, 2012.
- [161] D. T. H. El-Basuny, "Knowledge Representation Methods," *Principles of Artificial Intelligence, ICS-381*.
- [162] B. Market, "Bandwidth Market, Glossary," ed, 2005.

- [163] J. Durkin and J. Durkin, *Expert systems: design and development*: Prentice Hall PTR, 1998.
- [164] E. A. Feigenbaum, A. Barr, and P. R. Cohen, *The handbook of artificial intelligence*: Addison-Wesley New York, 1989.
- [165] W. J. Rapaport, "Holism, conceptual-role semantics, and syntactic semantics," *Minds and Machines*, vol. 12, pp. 3-59, 2002.
- [166] L. Chen, C. D. Nugent, and H. Wang, "A knowledge-driven approach to activity recognition in smart homes," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, pp. 961-974, 2012.
- [167] R. Davis, H. Shrobe, and P. Szolovits, "What is a knowledge representation?," *AI magazine*, vol. 14, p. 17, 1993.
- [168] D. Gašević, D. Djurić, and V. Devedžić, *Model driven architecture and ontology development*: Springer, 2006.
- [169] Z. Wemlinger and L. Holder, "The cose ontology: Bringing the semantic web to smart environments," in *Toward Useful Services for Elderly and People with Disabilities*, ed: Springer, 2011, pp. 205-209.
- [170] T. Sampalli, M. Shepherd, and J. Duffy, "A patient profile ontology in the heterogeneous domain of complex and chronic health conditions," in *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, 2011, pp. 1-10.
- [171] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, pp. 263-277, 2007.
- [172] D. Bonino and F. Corno, "Dogont-ontology modeling for intelligent domotic environments," in *The Semantic Web-ISWC 2008*, ed: Springer, 2008, pp. 790-803.
- [173] N. Porter, *Webster's Revised Unabridged Dictionary of the Engli*: G. & C. Merriam Company, 1913.
- [174] J. Davies, D. Fensel, and F. V. Harmelen, *Towards the semantic web*: Wiley Online Library, 2003.
- [175] B. Guo, D. Zhang, and M. Imai, "Toward a cooperative programming framework for context-aware applications," *Personal and Ubiquitous Computing*, vol. 15, pp. 221-233, 2011.
- [176] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge acquisition*, vol. 5, pp. 199-220, 1993.

- [177] D. Fensel, *Ontologies:: A Silver Bullet for Knowledge Management and Electronic Commerce*: Springer, 2003.
- [178] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: principles and methods," *Data & knowledge engineering*, vol. 25, pp. 161-197, 1998.
- [179] Y. Ding, D. Fensel, M. Klein, and B. Omelayenko, "The semantic web: yet another hip?," *Data & Knowledge Engineering*, vol. 41, pp. 205-227, 2002.
- [180] D. Fensel, "Ontologies: A Silver Bullet for Knowledge Management and Electronic Management," ed: Springer, Berlin, 2001.
- [181] D. L. McGuinness, "Ontologies come of age," *Spinning the semantic web: bringing the World Wide Web to its full potential*, p. 171, 2005.
- [182] N. Guarino, "Formal ontology, conceptual analysis and knowledge representation," *International Journal of Human-Computer Studies*, vol. 43, pp. 625-640, 1995.
- [183] S. Xu, J. Qi, and Z. Ma, "Ontology-Based Energy Model in Smart Home," in *Proceedings of the International Conference on Information Engineering and Applications (IEA)*, 2013, pp. 415-424.
- [184] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What are ontologies, and why do we need them?," *Intelligent Systems and Their Applications, IEEE*, vol. 14, pp. 20-26, 1999.
- [185] M. Uschold and M. Gruninger, "Ontologies and semantics for seamless connectivity," *ACM SIGMod Record*, vol. 33, pp. 58-64, 2004.
- [186] M. Paolucci, T. Kawamura, T. R. Payne, and K. Sycara, "Semantic matching of web services capabilities," in *The Semantic Web—ISWC 2002*, ed: Springer, 2002, pp. 333-347.
- [187] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana, "Web services description language (wsdl) version 2.0 part 1: Core language," *W3C Recommendation*, vol. 26, 2007.
- [188] C. Reinisch, W. Granzer, F. Praus, and W. Kastner, "Integration of heterogeneous building automation systems using ontologies," in *Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE*, 2008, pp. 2736-2741.
- [189] I. M. Dokas and C. Ireland, "Ontology to Support Knowledge Representation and Risk Analysis for the Development of Early Warning System in Solid Waste Management Operation," in *CD Proceedings of the International Symposium on Environmental Software Systems, ISESS*, 2007.

- [190] M. P. Singh and M. N. Huhns, *Service-oriented computing: semantics, processes, agents*: Wiley. com, 2006.
- [191] G. Antoniou and F. Van Harmelen, "Web ontology language: Owl," in *Handbook on ontologies*, ed: Springer, 2004, pp. 67-92.
- [192] D. Allemang and J. Hendler, *Semantic web for the working ontologist: effective modeling in RDFS and OWL*: Access Online via Elsevier, 2011.
- [193] P. Hitzler, M. Krotzsch, and S. Rudolph, *Foundations of semantic web technologies*: Chapman and Hall/CRC, 2011.
- [194] M. Stollberg, M. Moran, L. Cabral, and B. Norton, "Experiences from semantic web service tutorials," in *In Proc. of the Semantic Web Education and Training Workshop (SWET'06) at the First Asian Semantic Web Conference (ASWC 2006*, 2006.
- [195] J. De Bruijn, "Using ontologies-enabling knowledge sharing and reuse on the semantic web," 2003.
- [196] M. M. Taye, "Understanding Semantic Web and Ontologies: Theory and Applications," *arXiv preprint arXiv:1006.4567*, 2010.
- [197] I. Horrocks, C. Goble, and S. Bechhofer, "Daml+ OIL is not enough," in *En: International semantic web working symposium (Swws)*, 2001.
- [198] D. Brickley and L. Miller, "FOAF vocabulary specification 0.98," *Namespace document*, vol. 9, 2010.
- [199] D. Redavid, L. Iannone, T. Payne, and G. Semeraro, "OWL-S Atomic services composition with SWRL rules," in *Foundations of Intelligent Systems*, ed: Springer, 2008, pp. 605-611.
- [200] M. O'connor, H. Knublauch, S. Tu, and M. Musen, "Writing rules for the semantic web using SWRL and Jess," *Protégé With Rules WS, Madrid*, 2005.
- [201] P. Pauwels, D. Van Deursen, R. Verstraeten, J. De Roo, R. De Meyer, R. Van de Walle, *et al.*, "A semantic rule checking environment for building performance checking," *Automation in Construction*, vol. 20, pp. 506-518, 2011.
- [202] J. Domingue, D. Fensel, and J. A. Hendler, "Introduction to the Semantic Web Technologies," *J. Domingue, D. Fensel, & JA Hendler, Handbook of Semantic Web Technologies*, pp. 5-39, 2011.
- [203] S. Rodríguez-Valenzuela, J. A. Holgado-Terriza, P. Petkov, and M. Helfert, "Modeling Context-Awareness in a Pervasive Computing Middleware Using Ontologies and Data Quality Profiles," in *Evolving Ambient Intelligence*, ed: Springer, 2013, pp. 271-282.

- [204] L. Sommaruga, A. Perri, and F. Furfari, "DomoML-env: an ontology for Human Home Interaction," in *SWAP*, 2005.
- [205] L. Sommaruga, T. Formilli, and N. Rizzo, "DomoML: an integrating devices framework for ambient intelligence solutions," in *Proceedings of the 6th International Workshop on Enhanced Web Service Technologies*, 2011, pp. 9-15.
- [206] V. Tamma, *Ontologies for agents: theory and experiences*: Springer, 2005.
- [207] D. Preuveneers, J. Van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, *et al.*, "Towards an extensible context ontology for ambient intelligence," in *Ambient intelligence*, ed: Springer, 2004, pp. 148-159.
- [208] H. Chen, T. Finin, and A. Joshi, "The SOUPA ontology for pervasive computing," in *Ontologies for agents: Theory and experiences*, ed: Springer, 2005, pp. 233-258.
- [209] H. Chen, F. Perich, T. Finin, and A. Joshi, "SOUPA: Standard ontology for ubiquitous and pervasive applications," in *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, 2004, pp. 258-267.
- [210] H. Chen, T. Finin, A. Joshi, L. Kagal, F. Perich, and D. Chakraborty, "Intelligent agents meet the semantic web in smart spaces," *Internet Computing, IEEE*, vol. 8, pp. 69-79, 2004.
- [211] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung, "Ontology based context modeling and reasoning using OWL," in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, 2004, pp. 18-22.
- [212] G. Antoniou, *A semantic web primer*: the MIT Press, 2004.
- [213] P. Leitão, J. Barbosa, and D. Trentesaux, "Bio-inspired multi-agent systems for reconfigurable manufacturing systems," *Engineering Applications of Artificial Intelligence*, vol. 25, pp. 934-944, 2012.
- [214] J. J. Garguilo, S. Martinez, R. Rivello, and M. Cherkaoui, "Moving toward semantic interoperability of medical devices," in *High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability, 2007. HCMDSS-MDPnP. Joint Workshop on*, 2007, pp. 13-19.
- [215] M. Galarraga, L. Serrano, I. Martinez, P. de Toledo, and M. Reynolds, "Telemonitoring systems interoperability challenge: an updated review of the applicability of ISO/IEEE 11073 standards for interoperability in

- telemonitoring," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, 2007, pp. 6161-6165.
- [216] C. Stephanidis, A. Paramythis, M. Sfyraakis, A. Stergiou, N. Maou, A. Leventis, *et al.*, "Adaptable and adaptive user interfaces for disabled users in the AVANTI project," in *Intelligence in Services and Networks: Technology for Ubiquitous Telecom Services*, ed: Springer, 1998, pp. 153-166.
 - [217] K. Z. Gajos, M. Czerwinski, D. S. Tan, and D. S. Weld, "Exploring the design space for adaptive graphical user interfaces," in *Proceedings of the working conference on Advanced visual interfaces*, 2006, pp. 201-208.
 - [218] R. Kjeldsen, A. Levas, and C. Pinhanez, "Dynamically reconfigurable vision-based user interfaces," in *Computer Vision Systems*, ed: Springer, 2003, pp. 323-332.
 - [219] C. Gouin-Vallerand, B. Abdulrazak, S. Giroux, and M. Mokhtari, "A self-configuration middleware for smart spaces," *self*, vol. 3, 2009.
 - [220] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowledge engineering review*, vol. 10, pp. 115-152, 1995.
 - [221] T. Holz, M. Dragone, and G. M. O'Hare, "Where robots and virtual agents meet," *International Journal of Social Robotics*, vol. 1, pp. 83-93, 2009.
 - [222] L. Braubach, A. Pokahr, and W. Lamersdorf, "Jadex: A BDI-agent system combining middleware and reasoning," in *Software agent-based applications, platforms and development kits*, ed: Springer, 2005, pp. 143-168.
 - [223] J.-H. Kim, "Ubiquitous robot: Recent progress and development," in *SICE-ICASE, 2006. International Joint Conference*, 2006, pp. I-25-I-30.
 - [224] A. S. Rao and M. P. Georgeff, "BDI Agents: From Theory to Practice," in *ICMAS*, 1995, pp. 312-319.
 - [225] A. Guerra-Hernández, A. El Fallah-Seghrouchni, and H. Soldano, "Learning in BDI multi-agent systems," in *Computational logic in multi-agent systems*, ed: Springer, 2005, pp. 218-233.
 - [226] A. Casali, L. Godo, and C. Sierra, "A graded BDI agent model to represent and reason about preferences," *Artificial Intelligence*, vol. 175, pp. 1468-1478, 2011.
 - [227] M. E. Bratman, "Intention, plans, and practical reason," 1999.
 - [228] M. Dragone, S. Abdel-Naby, D. Swords, G. M. O'Hare, and M. Broxvall, "A Programming Framework for Multi-agent Coordination of Robotic Ecologies," in *Programming Multi-Agent Systems*, ed: Springer, 2013, pp. 72-89.

- [229] C.-H. Liu and J.-Y. Chen, "Using ontology-based bdi agent to dynamically customize workflow and bind semantic web service," *Journal of Software*, vol. 7, pp. 884-894, 2012.
- [230] I. Barakonyi and D. Schmalstieg, "Ubiquitous animated agents for augmented reality," in *Mixed and Augmented Reality, 2006. ISMAR 2006. IEEE/ACM International Symposium on*, 2006, pp. 145-154.
- [231] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge, "The belief-desire-intention model of agency," in *Intelligent Agents V: Agents Theories, Architectures, and Languages*, ed: Springer, 1999, pp. 1-10.
- [232] W. Li and Z. Qiu, "State-of-the-art technologies and methodologies for collaborative product development systems," *International Journal of Production Research*, vol. 44, pp. 2525-2559, 2006.
- [233] C. Petrie, "Agent-based software engineering," in *Agent-Oriented Software Engineering*, 2001, pp. 59-75.
- [234] M. J. Huber, "JAM: A BDI-theoretic mobile agent architecture," in *Proceedings of the third annual conference on Autonomous Agents*, 1999, pp. 236-243.
- [235] H.-K. Kim, "Convergence agent model for developing u-healthcare systems," *Future Generation Computer Systems*, 2013.
- [236] I. Dickinson and M. Wooldridge, "Towards practical reasoning agents for the semantic web," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 2003, pp. 827-834.
- [237] A. Haddadi and K. Sundermeyer, "Belief-desire-intention agent architectures," *Foundations of distributed artificial intelligence*, pp. 169-185, 1996.
- [238] V. Haarslev and R. Möller, "Racer: A Core Inference Engine for the Semantic Web," in *EON*, 2003.
- [239] X. Bai, D. Xu, G. Dai, W.-T. Tsai, and Y. Chen, "Dynamic reconfigurable testing of service-oriented architecture," in *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, 2007, pp. 368-378.
- [240] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?," *International journal of human-computer studies*, vol. 43, pp. 907-928, 1995.
- [241] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology," ed: Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, 2001.

- [242] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *Knowledge engineering review*, vol. 11, pp. 93-136, 1996.
- [243] S. Bechhofer, C. Goble, and I. Horrocks, "Requirements of ontology languages," *Deliverable 4.1 of the OntoWeb project (can be found at <http://ontoweb.aifb.unikarlsruhe.de/About/Deliverables>)*, 2003.
- [244] S. Boyce and C. Pahl, "Developing domain ontologies for course content," *Educational Technology & Society-ETS*, vol. 10, pp. 275-288, 2007.
- [245] E. M. Awad and H. M. Ghaziri, "Knowledge Management, 2004," ed: Prentice-Hall, Upper Saddle River, New Jersey.
- [246] E. Kim and J. Choi, "An ontology-based context model in a smart home," in *Computational Science and Its Applications-ICCSA 2006*, ed: Springer, 2006, pp. 11-20.
- [247] B. Medjahed and A. Bouguettaya, "A multilevel composability model for semantic web services," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, pp. 954-968, 2005.
- [248] N. Guarino, D. Oberle, and S. Staab, "What is an Ontology?," in *Handbook on ontologies*, ed: Springer, 2009, pp. 1-17.
- [249] A. Pease and I. Niles, "IEEE standard upper ontology: a progress report," *Knowledge Engineering Review*, vol. 17, pp. 65-70, 2002.
- [250] S. S. Yau, D. Huang, H. Gong, and Y. Yao, "Support for situation awareness in trustworthy ubiquitous computing application software," *Software: Practice and Experience*, vol. 36, pp. 893-921, 2006.
- [251] G. Antoniou and F. van Harmelen, "Web ontology language: OWL," in *Handbook on ontologies*, ed: Springer, 2009, pp. 91-110.
- [252] Y. Ma, B. Jin, and Y. Feng, "Semantic oriented ontology cohesion metrics for ontology-based systems," *Journal of Systems and Software*, vol. 83, pp. 143-152, 2010.
- [253] J. Bao, G. Slutzki, and V. Honavar, "A semantic importing approach to knowledge reuse from multiple ontologies," in *AAAI*, 2007, pp. 1304-1309.
- [254] S. Grimm and S. Grimm, "Knowledge representation and ontologies," in *Semantic Web Services*, 2007.
- [255] J. Wang, L. Chang, C. Zhu, and R. Dong, "Reasoning about Semantic Web Services with an Approach Based on Temporal Description Logic," in *Intelligent Information Processing VI*, ed: Springer, 2012, pp. 286-294.

- [256] Y. Ye, Z. Jiang, X. Diao, D. Yang, and G. Du, "An ontology-based hierarchical semantic modeling approach to clinical pathway workflows," *Computers in biology and medicine*, vol. 39, pp. 722-732, 2009.
- [257] Z. Li, Z. Li, X. Guo, P. Liang, K.-Q. He, and B. Huang, "A transformation approach from informal descriptions of SWRL to built-in elements of protégé4.1," in *Modelling, Identification & Control (ICMIC), 2012 Proceedings of International Conference on*, 2012, pp. 322-327.
- [258] D. Yang, R. Miao, H. Wu, and Y. Zhou, "Product configuration knowledge modeling using ontology web language," *Expert Systems with Applications*, vol. 36, pp. 4399-4411, 2009.
- [259] Q. Han, F. Gao, and H. Wang, "Ontology-based learning object recommendation for cognitive considerations," in *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, 2010, pp. 2746-2750.
- [260] S. W. Loke, "Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective," *Knowledge Engineering Review*, vol. 19, pp. 213-234, 2004.
- [261] J. F. Allen, "Towards a general theory of action and time," *Artificial intelligence*, vol. 23, pp. 123-154, 1984.
- [262] E. Serral, P. Valderas, and V. Pelechano, "Towards the model driven development of context-aware pervasive systems," *Pervasive and Mobile Computing*, vol. 6, pp. 254-280, 2010.
- [263] B. Motik, U. Sattler, and R. Studer, "Query answering for OWL-DL with rules," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 3, pp. 41-60, 2005.
- [264] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, *et al.*, "OWL-S: Semantic markup for web services," *W3C member submission*, vol. 22, pp. 2007-04, 2004.
- [265] J. Lloyd, *Foundations of Logic Programming*: Berlin: Springer-Verlag, 1987.
- [266] M. O'Connor, M. Musen, and A. Das, "Using the Semantic Web Rule Language in the Development of Ontology-Driven Applications," *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches*, 2009.
- [267] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, pp. 51-53, 2007.
- [268] O. Alliance, "OSGi technology," ed, 2012.

- [269] O. Alliance, "OSGi-the dynamic module system for Java," *accessed, May*, vol. 25, 2009.
- [270] J. S. Rellermeyer, G. Alonso, and T. Roscoe, "R-OSGi: distributed applications through software modularization," in *Proceedings of the ACM/IFIP/USENIX 2007 International Conference on Middleware*, 2007, pp. 1-20.
- [271] O. Alliance, "Osgi service platform, core specification, release 4, version 4.1," *OSGi Specification*, 2007.
- [272] S. Tomic, A. Fensel, M. Schwanzer, M. K. Veljovic, and M. Stefanovic, "Semantics for energy efficiency in smart home environments," *Applied Semantic Web Technologies*, pp. 429-454, 2012.
- [273] H. Knublauch, "Protégé-OWL API Programmer's guide," ed, 2006.
- [274] N. F. Noy, M. Crubézy, R. W. Fergerson, H. Knublauch, S. W. Tu, J. Vendetti, *et al.*, "Protege-2000: an open-source ontology-development and knowledge-acquisition environment," in *AMIA Annu Symp Proc*, 2003, p. 953.
- [275] S. Singh and R. Karwayun, "A comparative study of inference engines," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, 2010, pp. 53-57.
- [276] D. L. McGuinness and F. Van Harmelen, "OWL web ontology language overview," *W3C recommendation*, vol. 10, p. 10, 2004.
- [277] D. Ausin, F. Castanedo, and D. López-de-Ipina, "On the measurement of semantic reasoners in ambient assisted living environments," in *Intelligent Systems (IS), 2012 6th IEEE International Conference*, 2012, pp. 082-087.
- [278] A. C. Bukhari and Y.-G. Kim, "Ontology-assisted automatic precise information extractor for visually impaired inhabitants," *Artificial Intelligence Review*, vol. 38, pp. 9-24, 2012.
- [279] H. H. Shahri, J. A. Hendler, and A. A. Porter, "Software configuration management using ontologies," in *3rd International Workshop on Semantic Web Enabled Software Engineering (SWESE 2007), Innsbruck, Austria*, 2007.
- [280] G. J. Nalepa and W. T. Furmańska, "Pellet-HeaRT—proposal of an architecture for ontology systems with rules," in *KI 2010: Advances in Artificial Intelligence*, ed: Springer, 2010, pp. 143-150.
- [281] P. Kremen and Z. Kouba, "Ontology-Driven Information System Design," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, pp. 334-344, 2012.

- [282] A. F. Vilas, R. P. D. Redondo, J. J. P. Arias, M. R. Cabrer, A. G. Solla, and J. G. Duque, "An AmI-enabled OSGi platform based on socio-semantic technologies."
- [283] M. A. Zamora-Izquierdo, J. Santa, and A. F. Gómez-Skarmeta, "An Integral and networked home automation solution for indoor ambient intelligence," *Pervasive Computing, IEEE*, vol. 9, pp. 66-77, 2010.
- [284] M. Jahn, M. Jentsch, C. R. Prause, F. Pramudianto, A. Al-Akkad, and R. Reiners, "The energy aware smart home," in *Future Information Technology (FutureTech), 2010 5th International Conference on*, 2010, pp. 1-8.
- [285] O. Alliance, "About the OSGi Service Platform, technical whitepaper, revision 4.1," ed: June, 2007.
- [286] P. Dobrev, D. Famolari, C. Kurzke, and B. A. Miller, "Device and service discovery in home networks with OSGi," *Communications Magazine, IEEE*, vol. 40, pp. 86-92, 2002.
- [287] O. Gruber, B. Hargrave, J. McAffer, P. Rapiçault, and T. Watson, "The Eclipse 3.0 platform: adopting OSGi technology," *IBM Systems Journal*, vol. 44, pp. 289-299, 2005.
- [288] O. Alliance, "OSGi Alliance—Main," Website <http://www.osgi.org/Main/HomePage>, 2013.
- [289] T. Gu, H. K. Pung, and D. Q. Zhang, "Toward an OSGi-based infrastructure for context-aware applications," *Pervasive Computing, IEEE*, vol. 3, pp. 66-74, 2004.
- [290] G.-J. Ahn, H. Hu, and J. Jin, "Security-enhanced OSGi service environments," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 39, pp. 562-571, 2009.
- [291] X. Li and W. Zhang, "The design and implementation of home network system using OSGi compliant middleware," *Consumer Electronics, IEEE Transactions on*, vol. 50, pp. 528-534, 2004.
- [292] C. Lee, D. Nordstedt, and S. Helal, "Enabling smart spaces with OSGi," *Pervasive Computing, IEEE*, vol. 2, pp. 89-94, 2003.
- [293] A. L. Tavares and M. T. Valente, "A gentle introduction to OSGi," *ACM SIGSOFT Software Engineering Notes*, vol. 33, p. 8, 2008.
- [294] Y. Li, F.-y. Wang, F. He, and Z. Li, "OSGi-based service gateway architecture for intelligent automobiles," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, 2005, pp. 861-865.

- [295] M. Schaaf, "Extending OSGi by Means of Asynchronous Messaging," 2009.
- [296] I. Crnkovic and M. P. H. Larsson, *Building reliable component-based software systems*: Artech House Publishers, 2002.
- [297] G. Dewsbury, B. Taylor, and M. Edge, "Designing safe smart home systems for vulnerable people," *Dependability in Healthcare Informatics*, 2001.
- [298] E. D. Williams and H. S. Matthews, "Scoping the potential of monitoring and control technologies to reduce energy use in homes," in *Electronics & the Environment, Proceedings of the 2007 IEEE International Symposium on*, 2007, pp. 239-244.

Appendices

A.1: A Ontology Result Classified Using Pellet

1. Entity SubClassOf Action
2. Entity SubClassOf Event
3. PhysicalDevice SubClassOf Action
4. PhysicalDevice SubClassOf Event
5. ActionStatus SubClassOf Action
6. ActionStatus SubClassOf Event
7. ActionStatusEvent SubClassOf Action
8. Alarm SubClassOf Action
9. Alarm SubClassOf Event
10. AlarmEvent SubClassOf Action
11. DataPort SubClassOf Action
12. DataPort SubClassOf Event
13. DataPortGroup SubClassOf Action
14. DataPortGroup SubClassOf Event
15. Entertainment SubClassOf Action
16. Entertainment SubClassOf Event
17. Entity SubClassOf Action
18. Entity SubClassOf Event

19. Lamp SubClassOf Action
20. Lamp SubClassOf Event
21. Light SubClassOf Action
22. Light SubClassOf Event
23. Measurand SubClassOf Action
24. Measurand SubClassOf Event
25. MeasurandEvent SubClassOf Action
26. Parameter SubClassOf Action
27. Parameter SubClassOf Event
28. SetBooleanStatus SubClassOf Event
29. SetTRValve SubClassOf Event
30. FloorPlan SubClassOf Action
31. FloorPlan SubClassOf Event
32. SymbolicArea SubClassOf Action
33. SymbolicArea SubClassOf Event
34. Imp SubClassOf Action
35. Imp SubClassOf Event
36. GardenPlace Type Action
37. GardenPlace Type Event
38. GardenPlace Type OutdoorArea
39. GardenPlace Type SymbolicArea
40. GardenPlace Type Thing
41. GardenTemperatureLocation Type Action

42. GardenTemperatureLocation Type Event
43. GardenTemperatureLocation Type Parameter
44. GardenTemperatureLocation Type Thing
45. GardenTemperatureSensor Type OutdoorTemperatureMeasurand
46. GardenTemperatureSensor Type Action
47. GardenTemperatureSensor Type Entity
48. GardenTemperatureSensor Type Event
49. GardenTemperatureSensor Type Measurand
50. GardenTemperatureSensor Type TemperatureMeasurand
51. GardenTemperatureSensor Type Thing
52. GardenTemperatureSensor_TemperatureDataport Type
OutdoorTemperatureMeasurand
53. GardenTemperatureSensor_TemperatureDataport Type Action
54. GardenTemperatureSensor_TemperatureDataport Type DataPort
55. GardenTemperatureSensor_TemperatureDataport Type Event
56. GardenTemperatureSensor_TemperatureDataport Type Measurand
57. GardenTemperatureSensor_TemperatureDataport Type TemperatureMeasurand
58. GardenTemperatureSensor_TemperatureDataport Type Thing
59. GardenTemperatureSensor_TemperatureDataport_TemperatureEvent Type
OutdoorTemperatureMeasurand
60. GardenTemperatureSensor_TemperatureDataport_TemperatureEvent Type
Action
61. GardenTemperatureSensor_TemperatureDataport_TemperatureEvent Type Event

62. GardenTemperatureSensor_TemperatureDataport_TemperatureEvent	Type
Measurand	
63. GardenTemperatureSensor_TemperatureDataport_TemperatureEvent	Type
TemperatureMeasurand	
64. GardenTemperatureSensor_TemperatureDataport_TemperatureEvent	Type
Thing	
65. GardentTemperatureMeasurand Type OutdoorTemperatureMeasurand	
66. GardentTemperatureMeasurand Type Action	
67. GardentTemperatureMeasurand Type Event	
68. GardentTemperatureMeasurand Type Measurand	
69. GardentTemperatureMeasurand Type Thing	
70. HeatingControlOutdoorTemperatureMeasurand Type Action	
71. HeatingControlOutdoorTemperatureMeasurand Type Event	
72. HeatingControlOutdoorTemperatureMeasurand Type Measurand	
73. HeatingControlOutdoorTemperatureMeasurand Type TemperatureMeasurand	
74. HeatingControlOutdoorTemperatureMeasurand Type Thing	
75. HeatingControlService Type Action	
76. HeatingControlService Type Entity	
77. HeatingControlService Type Event	
78. HeatingControlService Type Thing	
79. HeatingControl_OutdoorTemperatureDataport Type Action	
80. HeatingControl_OutdoorTemperatureDataport Type DataPort	
81. HeatingControl_OutdoorTemperatureDataport Type Event	

- 82. HeatingControl_OutdoorTemperatureDataport Type Thing
- 83. HeatingControl_OutputTemperatureDataport_TemperatureEvent Type Action
- 84. HeatingControl_OutputTemperatureDataport_TemperatureEvent Type Event
- 85. HeatingControl_OutputTemperatureDataport_TemperatureEvent Type Thing
- 86. hasRuleGroup SubPropertyOf topObjectProperty
- 87. belongtoEntity SubPropertyOf topObjectProperty
- 88. connectToDataPort SubPropertyOf topObjectProperty
- 89. consistSymblicLocation SubPropertyOf topObjectProperty
- 90. containActionStatus SubPropertyOf topObjectProperty
- 91. controlDevice SubPropertyOf topObjectProperty
- 92. hasAction SubPropertyOf topObjectProperty
- 93. hasDataPort SubPropertyOf topObjectProperty
- 94. hasEvent SubPropertyOf topObjectProperty
- 95. hasLocation SubPropertyOf topObjectProperty
- 96. hasParameter SubPropertyOf topObjectProperty
- 97. inform SubPropertyOf topObjectProperty
- 98. hasAreaRelation SubPropertyOf topObjectProperty
- 99. hasPropertyPhrase SubPropertyOf: topDataProperty
- 100.isRuleEnabled SubPropertyOf: topDataProperty
- 101.isRuleGroupEnabled SubPropertyOf: topDataProperty
- 102.hasBooleanValue SubPropertyOf: topDataProperty
- 103.hasDescriptionValue SubPropertyOf: topDataProperty
- 104.hasFloatValue SubPropertyOf: topDataProperty

- 105.hasIntegerValue SubPropertyOf: topDataProperty
- 106.hasURLAddress SubPropertyOf: topDataProperty
- 107.isDynamicParameter SubPropertyOf: topDataProperty
- 108.isInputDataPort SubPropertyOf: topDataProperty
- 109.isWriteable SubPropertyOf: topDataProperty
- 110.nameAs SubPropertyOf: topDataProperty
- 111.hasPhysicalAddress SubPropertyOf: topDataProperty