



Fábio Adriano Seixas Lopes

Licenciado em Ciências de Engenharia

Eletrotécnica e de Computadores

Intelligent IoT and Dynamic Network Semantic Maps for more Trustworthy Systems

Dissertação para obtenção do Grau de Mestre em Engenharia
Eletrotécnica e de Computadores

Orientador: Doutor Ricardo Luís Rosa Jardim Gonçalves, Professor Auxiliar com
Agregação, Faculdade de Ciências e Tecnologia da Universidade
Nova de Lisboa

Co-orientador: Doutor Carlos Manuel Melo Agostinho, Professor Auxiliar
Convidado, Faculdade de Ciências e Tecnologia da Universidade
Nova de Lisboa

Júri:

Presidente: Prof. Dr. Luís Bernardo

Arguente: Prof. Dr. José Oliveira

Vogal: Dr. Carlos Agostinho



FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro 2017

Intelligent IoT and Dynamic Network Semantic Maps for more Trustworthy Systems

Copyright © Fábio Adriano Seixas Lopes, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa.

A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

For my parents and my brother.

Acknowledgements

I would like to thank all the people who supported me during the realization of my course and, in some way, contributed to this dissertation.

First, I would like to thank my family, specially my parents and brother, which supported me during these academic years and never gave up on believing in me. They always did it with an enormous willingness to help and I am very fortunate to have them in my life.

To my advisor, Dr. Ricardo Gonçalves, and the local coordinator of C2NET project, Dr. Carlos Agostinho, for giving me the great opportunity of working in a GRIS's project and believing in my capabilities. Their guidance, throughout the development of this dissertation, was a key factor to complete the objectives that I set to accomplish.

To all the people at GRIS, but specially to José Ferreira that helped me, during this dissertation, in many ways, with his great knowledge and for believing in my work, pushing me to do better and to successfully complete this dissertation.

Finally, I would also like to thank my closest friends, because they represent the true meaning of friendship and were, in many ways, part of my work and life, making them memorable and worthwhile.

As technology evolves, the Internet of Things (IoT) concept is gaining importance for constituting a foundation to reach optimum connectivity between people and things. For this to happen and to allow easier integration of sensors and other devices in these technologic environments (or networks), the configuration is a key process, promoting interoperability between heterogeneous devices and providing strategies and processes to enhance the network capabilities. The optimization of this important process of creating a truly dynamic network must be based on models that provide a standardization of communication patterns, protocols and technologies between the sensors. Despite standing as a major tendency today, many obstacles still arise when implementing an intelligent dynamic network. Existing models are not as widely adopted as expected and semantics are often not properly represented, hence resulting in complex and unsuitable configuration time. Thus, this work aims to understand the ideal models and ontologies to achieve proper architectures and semantic maps, which allow management and redundancy based on the information of the whole network, without compromising performance, and to develop a competent configuration of sensors to integrate in a contemporary industrial typical dynamic network.

Keywords: Internet of Things, Sensors, Sensor Configuration, IoT Models, Dynamic Network, Network Mapping, Semantic Maps

Com a evolução da tecnologia, o conceito de Internet das Coisas está a ganhar importância por constituir uma fundação que permite alcançar a conectividade ideal entre pessoas e coisas. Para isto acontecer e para facilitar a integração de sensores e outros dispositivos nestes ambientes tecnológicos (ou redes), a configuração é um processo chave, que promove a interoperabilidade entre dispositivos heterogêneos e providencia estratégias e processos para realçar as capacidades da rede. A otimização deste importante processo de criar uma rede verdadeiramente dinâmica deve ser baseada em modelos que permitem uma normalização de padrões de comunicação, protocolos e tecnologias entre os sensores. Apesar de ser uma grande tendência atualmente, muitos obstáculos ainda surgem quando se implementa uma rede dinâmica inteligente. Os modelos existentes não são tão amplamente adotados como seria de esperar e a semântica tende a não estar representada apropriadamente, resultando em configurações complexas e demoradas. Assim, este projeto consiste em perceber os modelos ideais e as ontologias para atingir arquiteturas e mapas semânticos adequados, que permitem a gestão e redundância baseada na informação da rede inteira, sem comprometer o desempenho, e para desenvolver uma configuração de sensores competente para implementar numa rede dinâmica típica da indústria contemporânea.

Palavras-Chave: Internet das Coisas, Sensores, Configuração de Sensores, Modelos IoT, Rede Dinâmica, Mapeamento da Rede, Mapas Semânticos

Table of Contents

Acknowledgements	vii
Abstract	ix
Resumo	xi
Table of Contents	xiii
List of Figures	xvii
List of Tables	xxi
Table of Acronyms	xxiii
1 Introduction	1
1.1 Motivation Scenario - IoT.....	2
1.2 Research Question	3
1.3 Hypothesis and Approach.....	4
1.4 Work Methodology	5
1.5 Dissertation Outline	5
2 IoT Models and Networks	7
2.1 Internet of Things.....	7
2.2 IoT Models	8
2.2.1 IoT-A Reference Architecture	8
2.2.2 W3C SSN Ontology	15
2.2.3 IoT Lite.....	18
2.2.4 Discussion.....	19
2.3 Semantics and Concepts of Intelligent Sensor Networks	21
2.3.1 Sensor Configuration	22
2.3.2 Context Awareness, Self-Configuration and Self-Adaptation	24

2.4	Dynamic Network Communication	26
2.4.1	Discussion towards Semantic Maps	28
3	Semantic Mapping	31
3.1	Overview	31
3.2	Concept Proposed	41
3.3	System Architecture	45
3.3.1	CEP	46
3.3.2	Devices	47
3.3.3	Device Knowledge Base (C2NET Ontology)	47
3.3.4	Mapping Knowledge Base (SMAP Ontology)	49
3.3.5	Semantic Mapping Module (SMAP)	50
3.4	Potential Application Scenario	52
4	Proof of Concept Implementation	57
4.1	Requirements and Functionalities	57
4.2	Technological Specifications	59
4.3	Implementation Steps	61
4.3.1	Step 1 – Designing the Architecture	61
4.3.2	Step 2 – Creation of Models and Instances on Protégé	61
4.3.3	Step 3 – Jena and Esper setup on Eclipse with Java	66
4.3.4	Step 4 – SPARQL Queries and manipulation of OWL files	66
4.3.5	Step 5 – Sensor Threads	67
4.3.6	Step 6 – Fault Detection (by Esper and by Ontology)	68
4.3.7	Step 7 – Recovery with Semantic Maps	69
4.3.8	Step 8 – Implementation of a Graphical User Interface	69
5	Testing and Hypothesis Validation	73
5.1	Testing Methodology	73
5.2	Testing Implementation	75

5.3	Hypothesis Validation	90
5.4	Scientific Validation	91
5.5	Industrial Validation	92
5.5.1	SMAP in C2NET	92
5.5.2	IoT Network Configuration in the Metalworking Case	93
6	Final Considerations and Future Work	97
6.1	Main Results	97
6.2	Conclusions	98
6.3	Future Work	100
7	References	101

Figure 1-1: Internet Of Things Impact (Kastelein, 2012).	1
Figure 1-2 - A IoT-A High Level Representation of the Reference Model and Reference Architecture (Bauer et al., 2012).	2
Figure 1-3 - : Internet Of Things Concept Illustration (Intersog, 2016).	3
Figure 2-1: IoT-A Reference Model Building Blocks(Unis et al., 2013).	8
Figure 2-2 - The IoT-A Tree (Bauer et al., 2012).	9
Figure 2-3 - Interaction of all sub-models in the IoT Reference Model (Unis et al., 2013).	10
Figure 2-4 - IoT Domain Model (Unis et al., 2013).	11
Figure 2-5 - IoT Information Model (Unis et al., 2013).	12
Figure 2-6 - IoT Functional Model (Unis et al., 2013).	13
Figure 2-7 - Interoperability Aspect of the IoT Communication Model (Unis et al., 2013).	14
Figure 2-8: The SSN Ontology (Compton, Barnaghi, & Bermudez, 2011).	15
Figure 2-9: The Stimulus-Sensor-Observation Pattern (Compton et al., 2011).	16
Figure 2-10: IoT-Lite Ontology (Bermudez-edo, Elsaleh, et al., 2015).	19
Figure 2-11 - Example of mapping between two ontologies (Ferreira, 2012).	21
Figure 2-12 - Context-Aware Sensor Configuration Model – CASCoM (Perera, Zaslavsky, Compton, Christen, & Georgakopoulos, 2013a).	23
Figure 2-13 – Context Aware Concept (Cisco, 2008).	24
Figure 2-14 – Self-Adaptation Concept (Weiss, Zeller, & Eilers, 2010).	25
Figure 2-15 - System Architecture of the CADDOT model (Perera, Jayaraman, et al., 2013). ..	26
Figure 2-16 - CADDOT Model for Sensor Configuration (Perera, Jayaraman, et al., 2013).	27
Figure 2-17 - Example of Relations (Paiva, 2015).	29
Figure 3-1 - CEP Generic Concept.	31
Figure 3-2 - Example of Detection of a Sensor Failure.	32

Figure 3-3 - Example of Solution to a Sensor Failure.....	33
Figure 3-4 Example of Sensor Malfunctioning.	33
Figure 3-5 - Semantic Mapping Model.....	38
Figure 3-6 - Device KB and Mapping KB.	40
Figure 3-7 - System Architecture.	40
Figure 3-8 - Activity Diagram of the Architecture’s Process.	42
Figure 3-9 - Sequence Diagram of the Architecture’s Process.	43
Figure 3-10 - Activity Diagram of the Design Process.....	44
Figure 3-11 - Sequence Diagram of the Design Process.....	45
Figure 3-12 - CEP Overview (Seeger, 2012).	46
Figure 3-13 – Device Knowledge Base Class Diagram.	47
Figure 3-14 – Fault Detection Class Diagram (with 3 Example Techniques).....	48
Figure 3-15 - Mapping Knowledge Base Class Diagram.	49
Figure 3-16 - Smart Factory Room Map Example.....	55
Figure 4-1 - C2NET Ontology OntoGraf.....	62
Figure 4-2 - C2NET Ontology OntoGraf Extended.	63
Figure 4-3 - Individual Property Assertions Example.....	64
Figure 4-4 - Mapping Ontology OntoGraf.	64
Figure 4-5 - Mapping Ontology OntoGraf Extended.	65
Figure 4-6 - SPARQL Query Example.	67
Figure 4-7 - EPL Statement Example	68
Figure 4-8 - SMAP Graphical User Interface.	70
Figure 5-1 – Example Scenario of Testing Implementation.	75
Figure 5-2 - Infrared Sensor.	76
Figure 5-3 - Ultrasonic Sensor.....	76
Figure 5-4 - Room 1: Start Room.	77
Figure 5-5 - Room 1: Room Started.....	77

Figure 5-6 - Room 1: Failure in a Sensor.	79
Figure 5-7 – Room 1: Semantic Maps for the Faulty Sensor.....	79
Figure 5-8 - Room 1: Result of the use of a Semantic Map.....	80
Figure 5-9 - Temperature Sensor.....	82
Figure 5-10 - Room 2: Changing the Sensor State.....	82
Figure 5-11 - Room 2: Sensor State changed.....	83
Figure 5-12 - Room 2: Inconsistency and Comparison Failures Detected.....	83
Figure 5-13 - Room 2: Use of a Semantic Map with two Destination Sensors.	84
Figure 5-14 - Room 2: Resulting Network State.	85
Figure 5-15 - Room 3: Initial Network State.....	87
Figure 5-16 - Room 3: Semantic Map.....	88
Figure 5-17 - Room 3: Network after the Semantic Map.....	88
Figure 5-18 - Room 3: Room Paused.....	89
Figure 5-19: C2NET Project Overview.	92
Figure 5-20 - Hardware for the C2NET implementation in AAMM.	94
Figure 5-21 - C2NET Implementation in the AAMM factory.	94
Figure 5-22 - C2NET AAMM’s Communication Message Example.	95

Table 2-1 – IoT Models Comparison.....	20
Table 3-1 - Semantic Mismatches (Ferreira, 2012), based on (Agostinho et al., 2011).	37
Table 3-2 - Examples for Possible Objectives of Sensing Devices in the Application Scenario.	53
Table 5-1 - Test Case 1.	78
Table 5-2 - Test Case 2.	81
Table 5-3 - Test Case 3.	86
Table 5-4 - Test Case 4	90

Table of Acronyms

<u>Acronyms</u>	<u>Definition</u>
AAMM	António Abreu Metalomecânica
ARM	Architecture Reference Model
CADDOT	Context-Aware Dynamic Discovery of Things
CAN	Controller Area Network
CASCoM	Context-Aware Sensor Configuration Model
CEP	Complex Event Processing
DB	Database
EDA	Event-Driven Architecture
EPA	Event Processing Agent
EPL	Event Processing Language
EU	European Union
FD	Functional Decomposition
FG	Functionality Groups
GRIS	Group for Research in Interoperability of Systems
GUI	Graphical User Interface
ID	Identifier
IDE	Integrated Development Environment
IEC	International Electrotechnical Comission
IEEE	Institue of Electrical and

	Electronics Engineers
IoT	Internet Of Things
IoT-A	Internet Of Things Architecture
IP	Internet Protocol
ISO	International Organization for Standardization
IT	Information Technology
KB	Knowledge Base
M2M	Machine to Machine
OSI	Open Systems Interconnection
OWL	Web Ontology Language
PIR	Passive Infrared
RFID	Radio-Frequency Identification
RW	Real World
SME	Small and Medium-Sized Enterprises
SOTA	State Of The Art
SOI	Situation of Interest
SSN	Semantic Sensor Network
SSO	Stimulus-Sensor-Observation
TTCN	Tree and Tabular Combined Notation
UDP	User Datagram Protocol
UML	Unified Modelling Language
UNINOVA	Institute for the Development of New Technology
W3C	World Wide Web Consortium
XML	Extensible Markup Language

Imagine a world where billions of objects can sense, communicate and share information, all interconnected over public or private networks. These interconnected objects have data regularly collected and analysed, providing a wealth of intelligence for planning, management and decision-making. This is the world of the Internet Of Things or IoT (Infocomm Development Authority, 2012). This concept represents various possibilities for a wide range of areas related, or to be related, in the future, to technology as we can see in Figure 1-1.

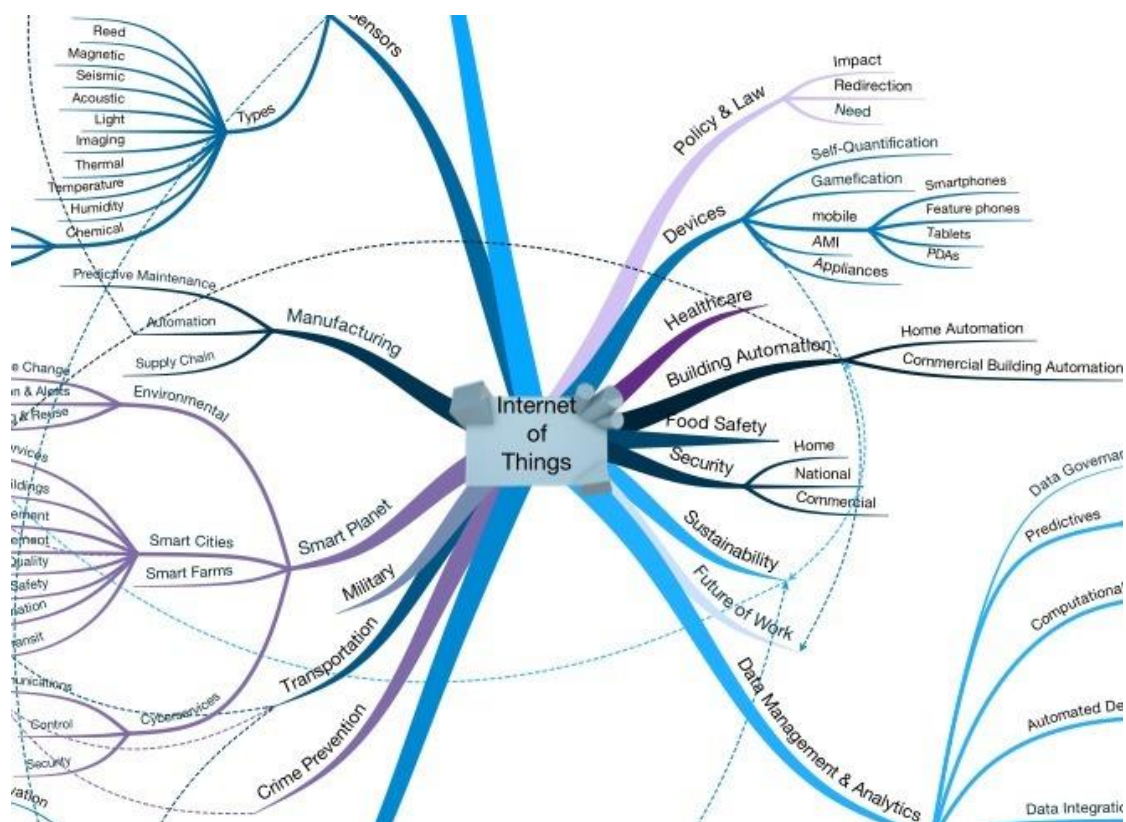


Figure 1-1: Internet Of Things Impact (Kastelein, 2012).

The Internet of Things is an emerging topic of technical, social and economic significance. Consumer products, sensors, industrial components and many other everyday objects are being combined with Internet connectivity and powerful data analytic capabilities that have the ability to transform the way we work and live (Rose Jaren, Eldridge Scott, 2015). To implement such idea, these IoT components, or IoT devices, must be somehow prepared to communicate, gather information and to share it, in order to develop a sustainable and useful network. IoT devices will only be successful when they can leave the abstract domain of experts and scientists, and can be used as standard commodities, and be deployed and setup by everyone (Chatzigiannakis et al., 2012).

To overcome the existing heterogeneity of devices, which use and demand different technologies to function, models of Reference Architectures are required. Some existing efforts to solve this problem are the IoT Architecture, the W3C SSN Ontology and the IoT Lite, reviewed in further chapters of this work. Other ontologies exist, but aim to different subjects or are not as useful to this work as the ones mentioned before.

This work aims to contribute to the contemporary methodology regarding the existing models for Reference Architectures in the Internet of Things environment by analysing them and understanding how they can contribute for a standardization of configuration and mapping of IoT devices to maximize compatibility, interoperability, safety and quality within an IoT network. In Figure 1-2, the dependencies and influences of a functional IoT model are abstractly presented.

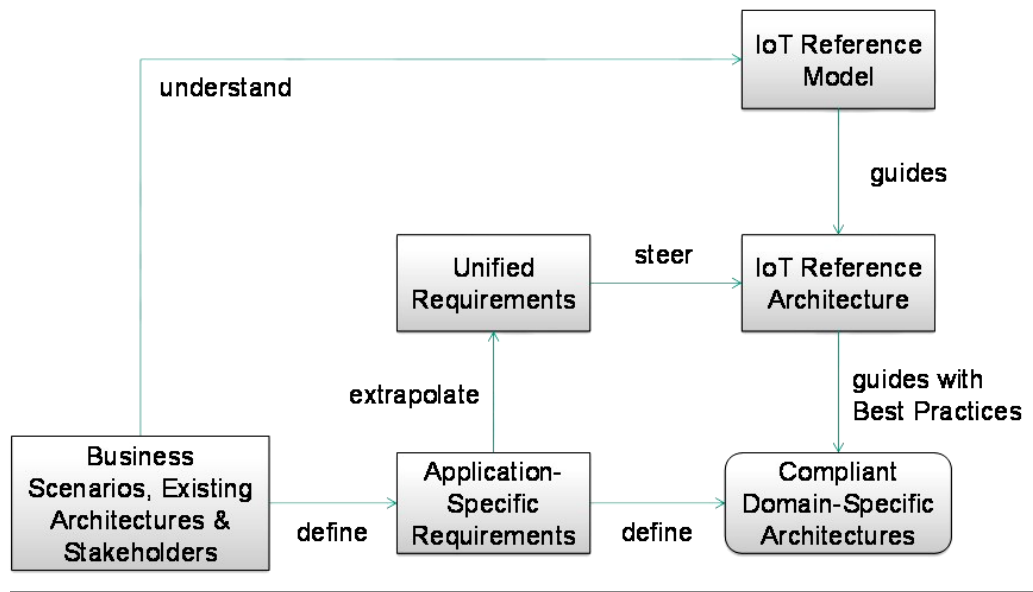


Figure 1-2 - A IoT-A High Level Representation of the Reference Model and Reference Architecture (Bauer et al., 2012).

The next section will provide some contextual information about the scenario and the motivation that lead to this topic. After that, the research question, hypothesis, approach and work methodology of this dissertation are presented.

1.1 Motivation Scenario - IoT

The Internet of Things (IoT) is becoming one of most discussed topics in technology today. It represents a concept that not only has the potential to change how technology is presented to us but also how we live. In the IoT paradigm, many of the common objects that surround us will be on the network in one form or another (Gubbi, Buyya, Marusic, & Palaniswami, 2013). As internet is becoming more widely available to the world, the cost of technology and connectivity is decreasing, more devices are being created with Wi-Fi and other sensors built into them, and smartphone use is skyrocketing. "All of these things are creating a *perfect storm* for the IoT" (Jacob Morgan, 2014).

The concept is based on machine-to-machine communications and interactions between objects, devices and people (Bermudez-edo, Elsaleh, Barnaghi, & Taylor, 2015). The term was coined in 1998 and later defined as “The Internet Of Things allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service” (Vermesan et al., 2009). In the following image (Figure 1-3), the ideology referenced by the definition of the term is illustrated.

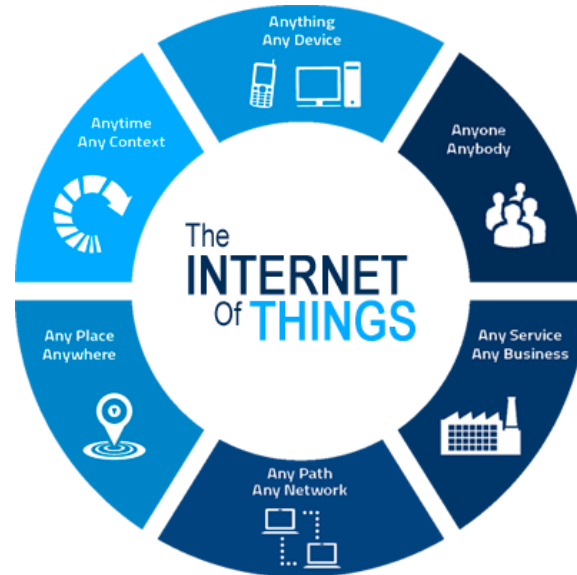


Figure 1-3 - : Internet Of Things Concept Illustration (Intersog, 2016).

In few words, IoT is the concept of connecting any device (such as cell phones, televisions, lamps and refrigerators) to the Internet and achieve broad connectivity to provide complex tasks or to make them easier. The potential is immeasurable and it is estimated that by 2020 the number of connected devices will grow to 26 billion (Rivera & Van der Muelen, 2013). The IoT can become a giant ubiquitous interoperable network of interconnected “things” (which will also include people) and has the potential to be applied to home automation or other concepts like “smart cities”, to help us reduce waste and improve energy efficiency.

The main motivation and focus of the research and development of this dissertation lays in the construction of *semantic maps* and configuration of devices, such as sensors, in a IoT domain, to achieve an intelligent and efficient network monitoring system. The idea behind the use of semantic maps is to gather and specifically organize the information of the network and its components, to allow redundancy by recovering from device errors/failures. For this, proper models must be studied, compared and implemented, to develop a useful solution for contemporary and future IoT environments.

1.2 Research Question

The general theme for this dissertation can be generally described as “*Dynamic Semantic Maps in IoT networks*”, which leads to the gathering of meta-information from the

network and its devices, specifically sensors due to its importance in dynamically adapt to the environment, to allow the creation of this specific data structures, the semantic maps, that aim to allow alternative configurations, redundancy and analyse the overall performance of the domain.

Thus, to properly define the procedures of the research work needed and to emphasize the objective of this work, it is important to formulate the research question that this thesis aims to answer during its completion.

RQ: *“Is it possible to use meta-information from network sensors to build semantic maps to support the analysis of degrees of similarity, alternative configurations and trustworthiness of systems?”*

This question provides some guidance during the development of this thesis and the answer is going to result from the necessary system implementation, validation and assessment.

1.3 Hypothesis and Approach

Based on the researched information and the research question, provided before in this chapter, this thesis is conducted regarding the following formulated hypothesis:

“If it is possible to use meta-information from network sensors and contextually analyse it, then we can dynamically build semantic maps in order to develop a more trustworthy and intelligent network, capable of reacting to the surrounding ecosystem constraints and variability.”

This statement will be challenged, implemented, tested and validated during the completion of this thesis. The results of the process will be presented and subsequently discussed, regarding this initial approach.

1.4 Work Methodology

To achieve the best outcome possible with the development of this thesis, it is important to define the necessary processes needed to an efficient scientific investigation and experimentation. The information and data here presented is based on the stages required for this particular topic, the recommendations of the thesis supervisors and classical scientific methodology bibliography (Nordgren, 2004), (Camarinha-Matos, 2012a), (Camarinha-Matos, 2012b) and (Chinneck, 1999).

According to an adaptation of the information from the sources previously mentioned, the selected main phases considered for this dissertation are:

1. **Research Questions** - Relates to a specific problem or area of knowledge, which has interest, for the author, to develop a question with the objective of improving an existing solution, finding some new aspect or solve a particular issue.
2. **Context Observations** - Includes observations and background information about the problem in study.
3. **Hypothesis** - Typically, is as an “educated guess” and is formed as a statement, that is proposed as an answer to the research question.
4. **Experimentation** - Is the phase where the experiment is planned in order to test the hypothesis. Considers several aspects such as variables, control, observation and methods of data collecting. It may result in a prototype or simulation environment to be tested, to allow the retrieval of significant data. In this case, the experimentation context is related to the C2NET project (C2NET, 2015), which is properly detailed further in this dissertation.
5. **Results Analysis**- The results are usually in the form of a statement that explains or interprets the data. It could provide a positive answer to the formulated hypothesis or it can prove it wrong. In the last case, further different hypothesis should be formulated or the problem (research question) must be reconsidered. In either case, positive or negative results are considered helpful for the scientific community.
6. **Thesis Writing and Publishing** - This last phase includes the explanation and conclusion of the entire work, in this case, in a dissertation.

1.5 Dissertation Outline

After the introduction provided in this section, this dissertation evolves into the following chapters:

Chapter 2 – IoT Models and Networks: This section presents the research of contemporary solutions and concepts for the relevant subjects of this dissertation. After the initial contextualization in the IoT paradigm, the studied models of architecture are presented and briefly discussed regarding the aspects that are important for this thesis. Finally, the main aim for this thesis is introduced, Semantic Maps, and relevant topics, such as sensor configuration and network concepts, are described and integrated in the environment of this work.

Chapter 3 – Semantic Mapping: In this chapter, an overview of the architecture considered in this dissertation is presented and explained. This architecture attempts to describe, on a high level, the solution for the module designed that will implement the concept of semantic maps.

Chapter 4 – Proof of Concept Implementation: This chapter introduces the design process of the proposed solution and includes a detailed report about the practical component of this thesis along with a thorough explanation on what it consists and why it was considered in that way.

Chapter 5 - Testing and Hypothesis Validation: In this section, the tests used to validate the formulated hypothesis and the respective analysis are presented. The contexts provided for the experiments detailed in this chapter are designed to embody real world environments, to ensure the accuracy of the provided solution and methodology, and following the demands of the foreseen integration and validation with other related research activities.

Chapter 6 – Final Considerations and Future Work: The final chapter of this dissertation is an overall comparison between all the work developed and the initial expectations, based on what was studied and the contemporary solutions for this thematic. After the aforementioned analysis and final statements, some notes about what could be improved are provided along with the already planned future work.

In this section, an overview of the researched scientific literature about this dissertation is presented. The Internet of Things sub-section is a complement to the information already mentioned in the introduction chapter and addresses the relation with the rest of the sub-sections. The IoT Models sub-section presents an overview of the contemporary approach on IoT architectures, how they function and what advantages we may take from them. Finally, the last two sub-sections of this chapter, present the gathered information about the major issues of creating a dynamic and intelligent sensor network through semantic mapping, sensor configuration and environment awareness.

The objective of this initial investigation is to form an efficient basis for the implementation and further work, explained in the further chapters, and to understand the concerning main concepts. Further specific research about the development of this dissertation may be presented in other chapters, to allow a better understanding of the mentioned and used concepts.

2.1 Internet of Things

During the previous chapter of introduction, the definition and concept of Internet of Things was explained in order to set the scenario that motivates the development of this work. With that in mind, this sub-section is only focused on the relation that emerges with the need for IoT reference architectures, models and ontologies.

Internet of Things (IoT), as mentioned before, is the communication between objects, devices and people. In the near future, the communications and information processing will be ubiquitous and performed by IoT systems (Bermudez-edo, Elsaleh, et al., 2015). With the rapid development of the internet and technology, the amount of information has increased exponentially (online and technologically generated). A lack of standardization and common vocabulary has continued to generate heterogeneity, which strongly hinders information exchange and communication (Ding & Fensel, 2001), the main purpose of the IoT. In order to be successful with the concept of Internet of Things, some base models, or ontologies, need to be developed to define how the components will work. An ontology, in computer science, is the working model of entities and interactions either generically or in some particular domain of knowledge (Stevens, 2001). These ontologies must consider that there is a wide variety of sensors or devices that communicate using different protocols and technologies, they often measure different things in very different ways and are not capable of exchanging information with other devices easily, due to being developed for specific situations.

The correct configuration, regarding efficient ontologies for these components, is believed to be the key for a successful implementation, because it provides the needed tools and contexts to successfully integrate and configure sensors in order to manage them, or give them the capability of managing themselves, and be able to monitor and act on the network as a whole.

2.2 IoT Models

A model is a definition of some slice of reality which is being observed and interpreted, that is designed through the use of abstract elements and relationships in order to correspond to real scenarios and environments (Correia, 2010).

An IoT Model aims to tackle the problems mentioned above, in a generic way. They usually propose a high-level reference architecture or ontology that establishes identities, procedures, configurations and relationships between different entities in various domains, to support any emergent idea or concept related to the Internet of Things. To understand the importance of these models, it is going to be mentioned some of the most prominent and widely available. Some other IoT models like IBM Watson, Cisco IoT System, Google Brillo, Z-Wave, Osmose, Sensei and DUL are also known, but are not as widely spread, available or relevant to the subjects of this work.

2.2.1 IoT-A Reference Architecture

In this model, the European Lighthouse Integrated Project, addresses the structural concerns related to the Internet of Things and creates a proposal for an Architectural Reference Model (ARM), starting with the definition of an initial set of key building blocks regarding functionality, scalability, performance, deployment and security, with the purpose to eventually derive into a large set of concrete IoT architectures.

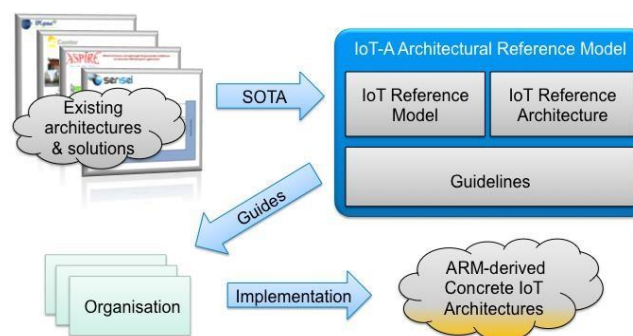


Figure 2-1: IoT-A Reference Model Building Blocks(Unis et al., 2013).

As seen in Figure 2-1, one defining choice of the IoT-A project was to base its work on the current state of the art (SOTA), rather than opting for a new approach. Due to this choice, backward-compatibility is ensured, and the solutions adopted are already established in the field. The Reference Model provides the highest abstraction level for the definition of the IoT

Architectural Reference Model, the Reference Architecture is the reference for building compliant IoT Architectures and the Guidelines are how these models, views and perspectives can be concretely used.

The IoT-A is often presented with a metaphor of a tree, not to be interpreted too strictly, as seen in Figure 2-2.

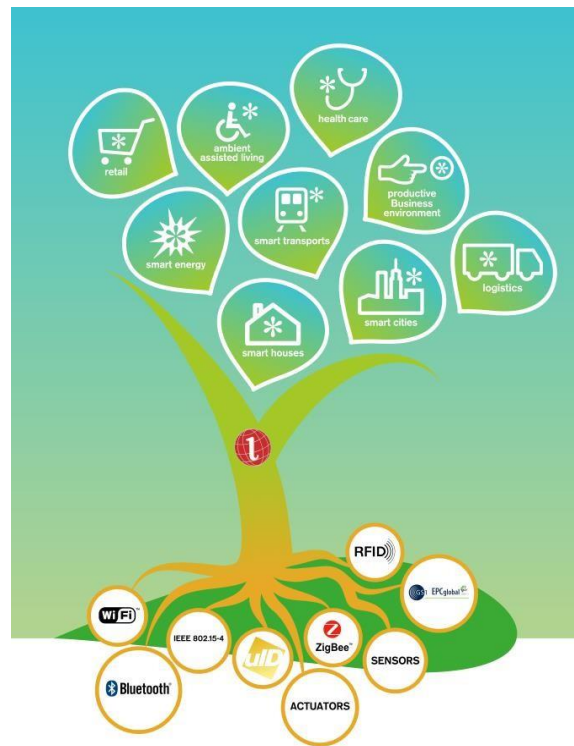


Figure 2-2 - The IoT-A Tree (Bauer et al., 2012).

The roots of the tree include selected groups of communication protocols (such as WiFi, ZigBee, IPv6 and RFID) and device technologies (such as sensors, actuators and tags). These protocols and devices are in fact the base of an efficient and interoperable IoT model, because by achieving complete understanding and commitment between them, full connectivity is reached and exchanging information can become an easy task. The leaves of the tree represent the wide variety of IoT applications that can be built from the trunk. The leaves only succeed if the roots have a proper functioning and for this functioning to be useful, the trunk must have an incisive and correct approach on the way the devices and protocols work with each other. The trunk is represented here as the Architectural Reference Model (ARM).

The IoT Reference Model aims at establishing a common grounding and a common language for IoT architectures and IoT systems (Unis et al., 2013). It consists of several sub-models: Domain Model, Information Model, Functional Model, Communication Model and the Trust, Security and Privacy Model. The Figure 2-3 describes how concepts and aspects of each model are used to support others.

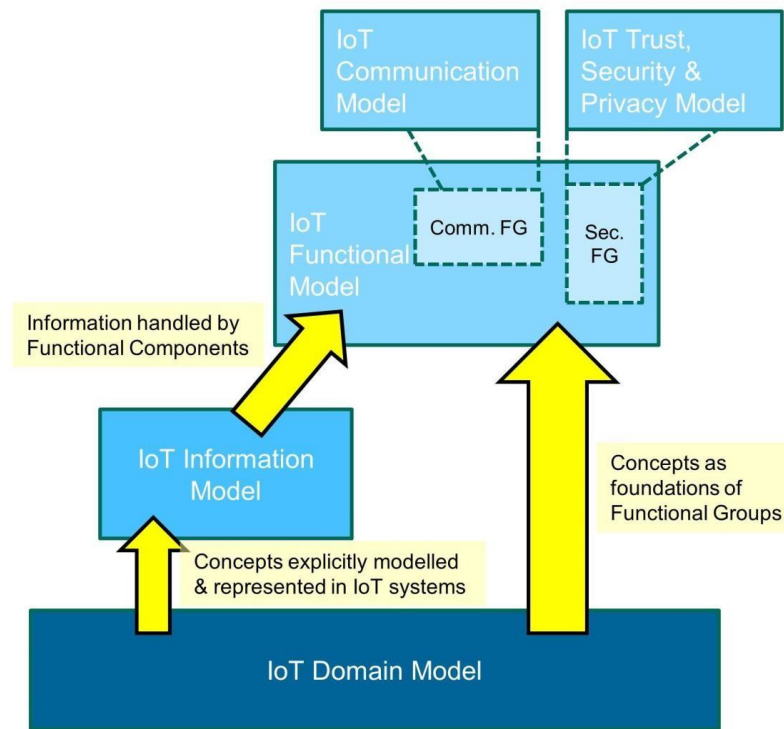


Figure 2-3 - Interaction of all sub-models in the IoT Reference Model (Unis et al., 2013).

To better understand the general architecture of this model, an explanation of some inherent concepts is presented next (Virtual/Physical Entities, Sensors/Actuator/Tags and others). These concepts are transversal to other models and are heavily mentioned during further chapters of this thesis. After this, some information about the sub-models, that form the IoT Reference Model, is provided.

- **Sensors, Tags and Actuators** - A sensor provides information about the physical environment it monitors. Tags are used to identify physical components of a system, usually they need to be read using specific sensor devices. Actuators are responsible for actions requested by the system, or user, and express modifications in the physical state of the environment.
- **Physical and Virtual Entities** - “An entity is anything that has a distinct existence” (Rannenber, Royer, & Deuker, 2009), physically or virtually. The designations mentioned are, in this context, directly associated with physical and virtual representations of certain elements of an IoT system. The representations may be directly linked, for example, a physical entity is represented by the corresponding virtual entity or to a group of virtual entities and vice versa. It can even not be any links and these connections may appear and disappear during the operation of the system.
- **Database (DB) and Knowledge Base (KB)** - A database is the storage of information, in various formats and typically organized in a way to model certain aspects of reality and supports certain processes that need such information. A knowledge base is often mentioned as a complex technology used to store structured or not structured

information to be used by a computer system. They are known to be a type of database but with a more specific purpose to systems and intelligent decision.

- **Simple and Complex Events** - A simple event is usually referred to as an action (or component), usually a time observation, which results from the operation of a system or derives from it. A complex event derives from various (two or more) simple events and requires much more processing and allows much more complex solutions.
- **Complex Event Processing (CEP)** - The CEP consists in reading and analysing information about events and to deduce conclusions about them. The complex event processing combines information from various sources or devices to infer events or patterns that suggest more complicated circumstances. The objective of this method is to identify important events (such as opportunities or threats) and respond quickly and effectively.

2.2.1.1 Domain Model

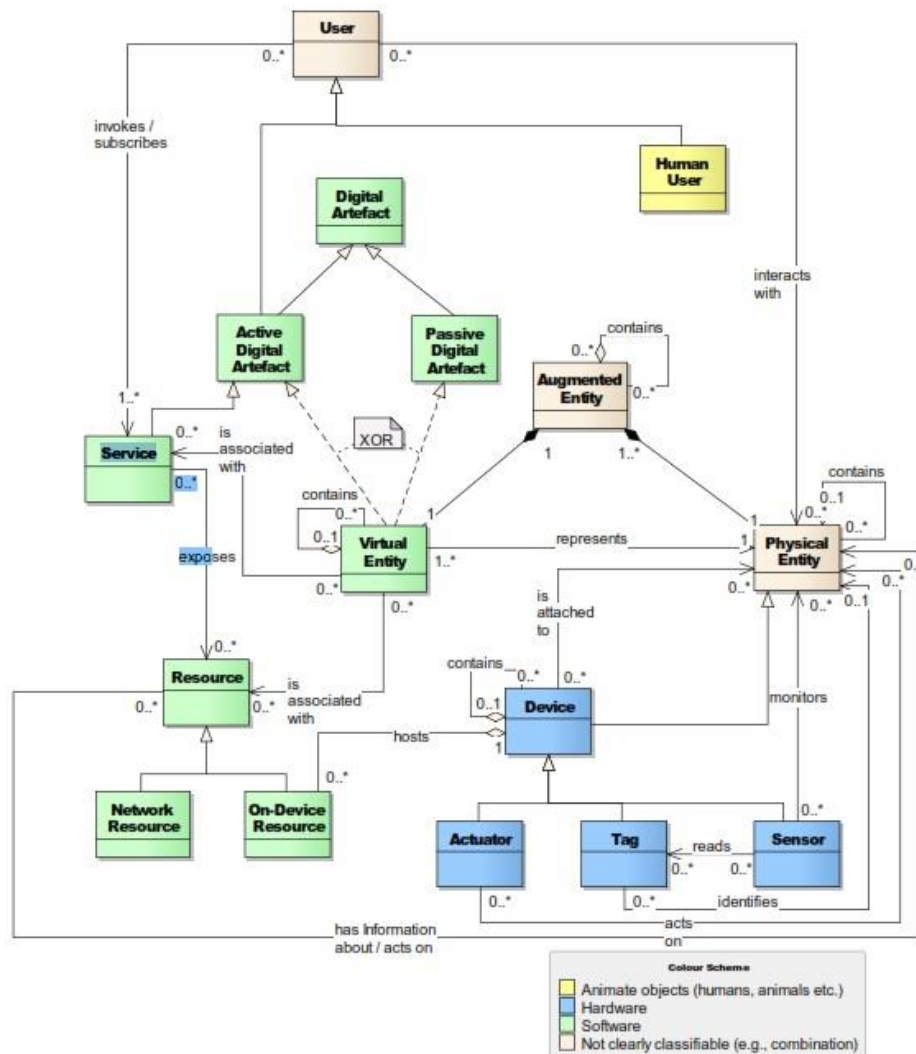


Figure 2-4 - IoT Domain Model (Unis et al., 2013).

It is the foundation of the Reference Model and describes all the relevant concepts in the Internet of Things like devices, services and virtual entities, and the relationships between them. It also provides a common lexicon and taxonomy of the IoT domain (Muller, 2008).

The main purpose of a domain model is to generate a common understanding of the target domain in question. Such common understanding is important because provides the possibility to argue about architectural solutions and to evaluate them (Unis et al., 2013).

In Figure 2-4, we can see the definition of basic attributes of the concepts related to the IoT systems and the way they interact. The software components are layered upon the hardware components to extend them to the virtual world where decision-making occurs, using the model and functional principles. The decisions and reasoning are then passed to the hardware part for the interaction with the environment.

2.2.1.2 Information Model

Defines the structure (relations and attributes) of IoT related information in a system on a conceptual level. It can be seen as a meta-model that provides a structure for the information to be handled by IoT systems (Unis et al., 2013). That structure provides representation, processing, storage and information retrieval to the system.

The IoT Information Model represents all the concepts of the Domain Model that are to be explicitly represented and manipulated in the digital world (Unis et al., 2013).

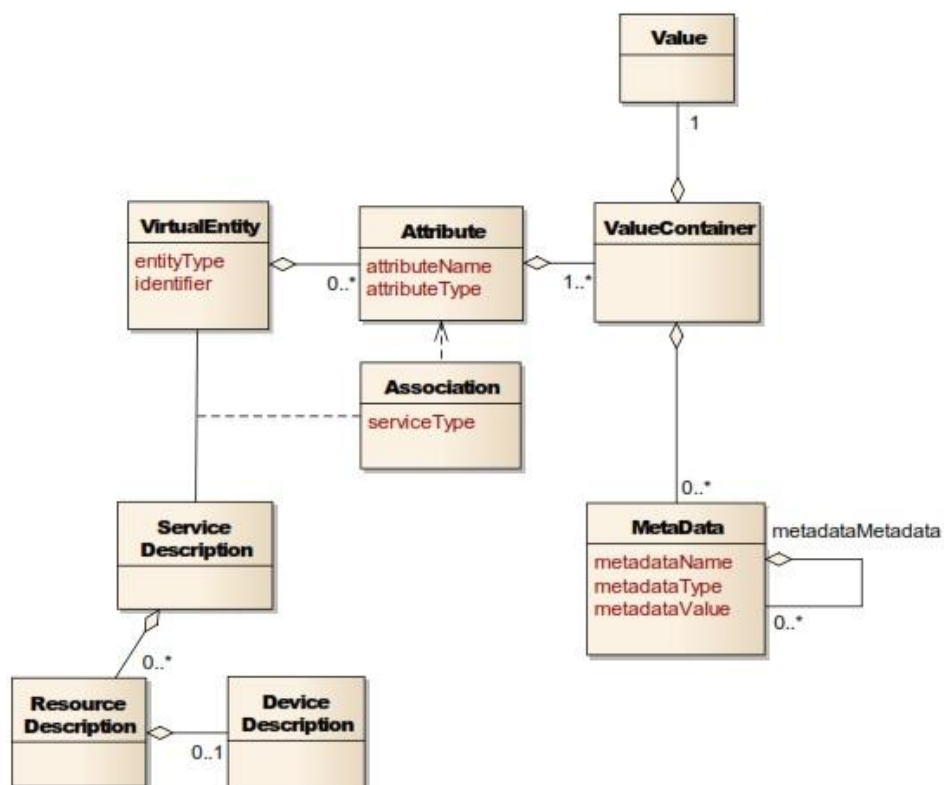


Figure 2-5 - IoT Information Model (Unis et al., 2013).

In this model, the information of the virtual entities is organized and defined using relevant context for possible applications of the various components. The resultant structure of the information represented in the Figure 2-5 shows how it is handled and processed in an IoT system.

2.2.1.3 Functional Model

Identifies groups of functionalities and their interactions as seen in Figure 2-6. The functional model is an abstract framework for understanding the Functionality Groups (FG) that are explained by Functional Decomposition (FD), which is the process to identify and relate each FG to the others. With this process, the complexity of the system is divided into smaller and more manageable parts.

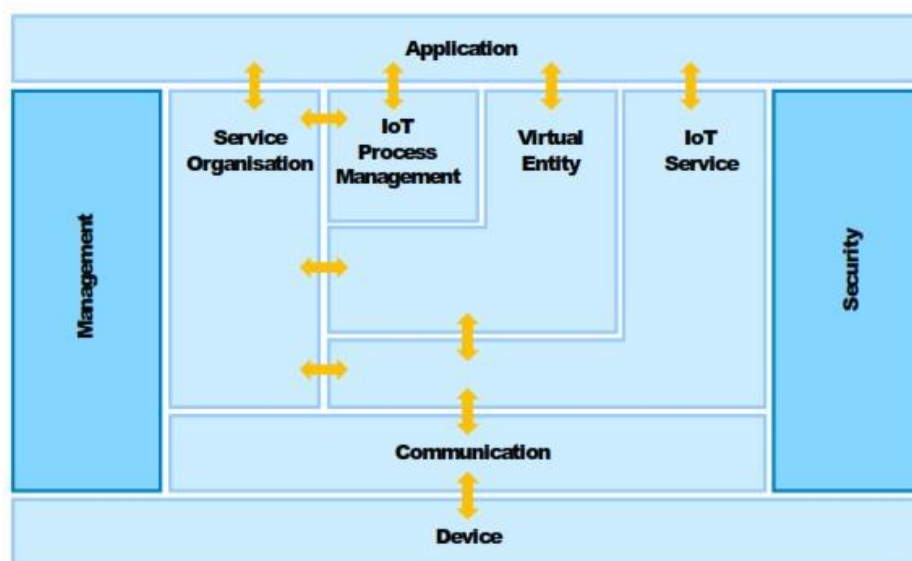


Figure 2-6 - IoT Functional Model (Unis et al., 2013).

In Figure 2-6 is depicted the derivation of the model. From the main concepts of the Domain Model (Virtual Entities, Devices, Resources and Users) others emerge like the Application, Virtual Entity, IoT Service and Device components. From the need to communicate and exchange information to support the IoT system, the Communication component is presented and included. The requirements to build services and applications are covered by the IoT Process Management and Service Organization. To address the concern about IoT Trust, Security and Privacy, the Security component is identified. Finally, the Management is a transversal component required for the management and interaction between the functional groups.

2.2.1.4 Communication Model

The Communication Model introduces concepts for handling the complexity of communication in heterogeneous IoT environments. It aims at the current paradigms for connecting elements and, for each case, to create an interoperable network capable of

efficiently exchanging information. The proposed model is based on the ISO OSI 7-layer model for networks and it highlights the aspects about interoperability among different stacks.

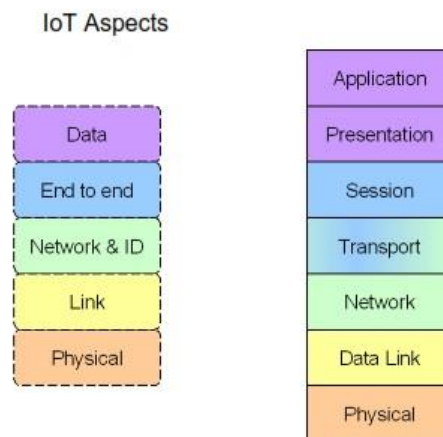


Figure 2-7 - Interoperability Aspect of the IoT Communication Model (Unis et al., 2013).

The Figure 2-7 emphasizes the interoperability aspects that the IoT Communication Model wants to develop in comparison to the ISO/OSI stack (presented on the right of the figure). The dashed lines were used for highlighting the IoT aspects relatively to the stack layers. This transversal approach allows the existing protocol stacks, after the system is modelled according to the IoT specifications and interoperable needs, to be adapted easily.

2.2.1.5 Trust, Security and Privacy Model

In practical applications, this model defines some characteristics to consider when building a safe system. The model of this aspect of IoT systems was not defined entirely but there were considered many guidelines, methods and protocols to ensure trust, security and privacy without damaging the system's efficiency. We will review the relevant concepts of this model without stepping too further into the protocols or specifications described in this model.

Trust regards the aspects of authentication when dealing with other entities and exchanging sensitive information with them. To properly achieve the compliance needed to the expected functional behaviour, all entities, protocols and mechanisms in an IoT system must be validated. Some mechanisms that allow this process are protocols for integrity and confidentiality, endpoint authentication, non-repudiation methods, behaviour policies and a trust anchor (entity to be trusted by default).

Security aspects are focused on the layers of service, communication and application. It is a very general term, but it focuses on authorization, authentication, identity management, trust, reputation and key exchange (each entity has public and private keys to communicate and encrypt messages/information).

Privacy regards non-authorized information spreading and security management in data banks. Due to the variety of entities that handle data in IoT, guaranteeing data privacy is an essential but difficult feature. To accomplish this, mechanisms of access policies, encryption/decryption algorithms, security and management credentials can be used.

2.2.1.6 Review

In this sub-section, it was presented a very broad and complete model for organizing an IoT architecture. The reference model and sub-models, define the basic concepts, terminologies and relationships in the IoT ARM. This architecture is very complete and represents one step further in defining the IoT reference architecture, but it lacks the applicability that other more specific ontologies have proposed due to a less care for specific implementations and real-world situations.

2.2.2 W3C SSN Ontology

In this model, an OWL is used, an international standard for encoding and exchanging ontologies designed to support the Semantic Web. The concept of Semantic Web is that information should be given explicit meaning, so that machines can process it more intelligently (Heflin, 2006).

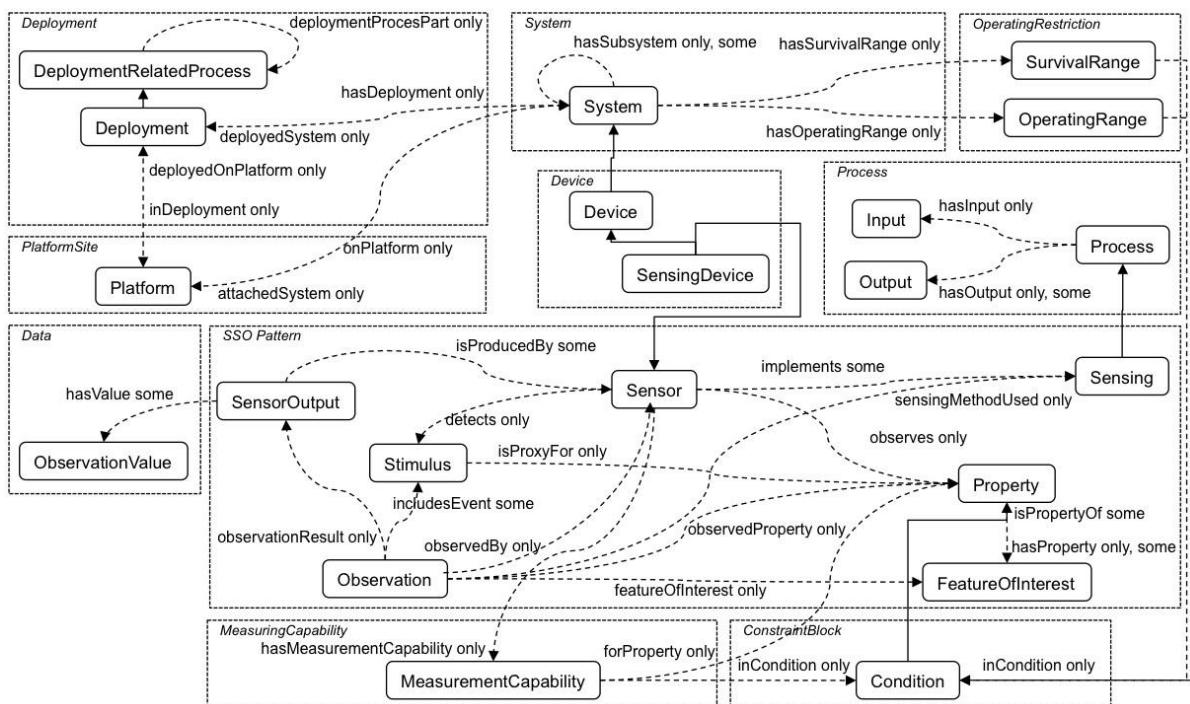


Figure 2-8: The SSN Ontology (Compton, Barnaghi, & Bermudez, 2011).

The OWL, designed to represent complex knowledge about objects, groups of objects and their relationships, is used in this ontology to describe sensors, their capabilities, measurement processes and resultant observations. This ontology contains only concepts and relations directly relevant to the sensors, leaving aside concepts related to other domains. In

this way, the ontology is better positioned to provide modularity and reusability. The key specifications of the sensor information are based on their accuracy, the observations and methods used for sensing, the concepts for operating and, related with the structure for field deployments, the deployment lifetime and sensing purpose. The ten conceptual modules, key concepts and relations are shown in Figure 2-8.

The SSN Ontology is built around a central pattern describing the relationships between sensors, stimulus and observations, the Stimulus-Sensor-Observation (SSO) pattern (Compton et al., 2011). The ontology is divided into four main perspectives:

- A sensor perspective (what senses, how it senses and what is sensed);
- A data or observation perspective (observations and related metadata);
- A system perspective (systems of sensors and deployments);
- A feature and property perspective (what senses a particular property or what observations have been made about it).

2.2.2.1 The Stimulus-Sensor-Observation (SSO) Pattern

The SSO pattern, as seen in Figure 2-9, links sensors, what they sense and the observations that result, considering three of the four main perspectives mentioned before (the system perspective is more about system organisations and deployments than sensing, but it surely relates to the SSO Pattern).

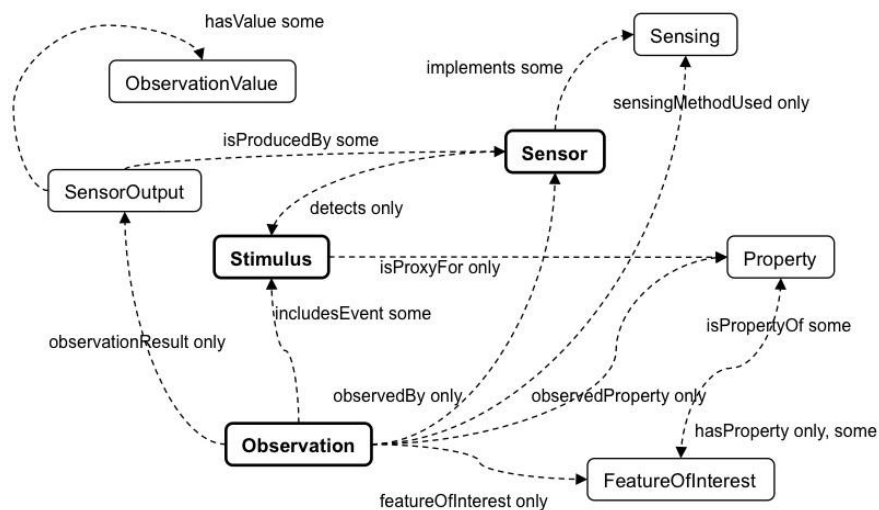


Figure 2-9: The Stimulus-Sensor-Observation Pattern (Compton et al., 2011).

Stimuli

Stimuli are changes (or events) in a environment that a sensor can detect and use to measure a property. A stimulus is a representation of an observable property, or a group of observable properties.

Sensors

Sensors are usually physical objects that observe, transforming incoming stimuli into another, often digital, form of representation. This new representation normally comprises the need to handle the obtained information in data computation. Sensors may be hardware devices, sensing systems or anything that somehow senses.

Observations

Observations are essentially the core of this pattern. For a specific sensing event, an observation can link the act of sensing, the event that is the stimulus, the observing sensor, the sensing method used, a result and an observed feature and property, with the objective of placing all of them in an interpretative context.

2.2.2.2 Perspectives

Sensor Perspective

The SSO pattern describes a sensor in terms of its stimulus, sensing method and the observations it makes (Compton et al., 2011). This perspective also includes the capabilities of the sensors. For each property that a certain sensor can observe, the performance and accuracy of the sensor might be influenced by environmental conditions related or not to the property under observation.

The accuracy is an observable property of the sensor, generally defined as the maximum difference that will exist between the actual value and the indicated value at the output of the sensor (Carr & Brown, 2000), often mentioned by a data sheet that lists properties observed in various conditions that also include precision, resolution and measurement range. A sensor may have various measurement capabilities, describing the capability of the sensor in various situations, which are themselves observable properties of the sensor's environment. A measurement capability instance collects observed properties of a sensor in the conditions specified. A sensor may have links to any number of instances for determining capability in various situations.

Observation Perspective

The observation perspective places a context for interpreting incoming stimuli, the observing event and the stimulus. The context includes the observed feature, property, observing sensor, result and method from the previously described pattern and can also report a quantitative approximation of quality of the observation, a time that the result became available and a time at which the sampling occurred.

System Perspective

The system perspective is the view representing part of a sensing infrastructure. A system has components, which are also systems that have sub-concepts like sensing devices

that have operating and survival ranges and may be deployed on certain platforms. Similarly, to how prevailing environmental conditions may affect the performance of a sensor, a system or device may have a defined operating environment, and environmental extremes may exceed the capacity of a system to survive and make further observations (Compton et al., 2011). The means for describing operating and survival ranges are the same as for sensors and measurement capabilities because they are observable properties of the systems. The operating range includes features such as power range, power sources and standard configurations. The survival range describes standard environmental conditions to which the sensor can be exposed without suffer predictable damage.

To this perspective is also relevant to mention the deployment, which is a process that includes all phases in the lifetime of an operating system such as installation, maintenance and further deactivation. A system is generally deployed on a platform. The location of platforms, systems or sensors and temporal properties of deployments can be abstractions of real-world locations or also absolute or relative locations.

2.2.2.3 Review

This OWL ontology, describes a specific straightforward approach to sensors, sensing, measurement capabilities of sensors, observations that result from sensing and deployments, or field applications, in which the sensors are used. The configuration, in this ontology, can be represented as an adaptive process which provides an interesting base for the objective of this work. The network is visualised in modules to provide better reusability and adaptation based on sensor observation. It lacks additional information regarding sensor relative entities and the domains where they are used, which means that this ontology has to be complemented with other ontologies or models in order to develop an efficient practical solution.

2.2.3 IoT Lite

The IoT Lite is a lightweight ontology, based on the SSN Ontology, to represent Internet of Things resources, entities and services (Bermudez-edo, Elsaleh, et al., 2015). By creating a representation, in a more lightweight approach than previously existed in other ontologies, it is possible to achieve an ontology able to provide shorter response time and thus create a more efficient structure for systems.

This ontology describes IoT concepts into three classes. Objects, System or Resources and Services (Bermudez-edo, Barnaghi, & Elsaleh, 2015). The devices are also split into, but not restricted to, three classes: sensing devices, actuating devices and tag devices. The services in the system are described with an availability or access control and a coverage (area covered by the IoT device). The figure 2-10 depicts the concepts of the ontology and the main relationships between them.

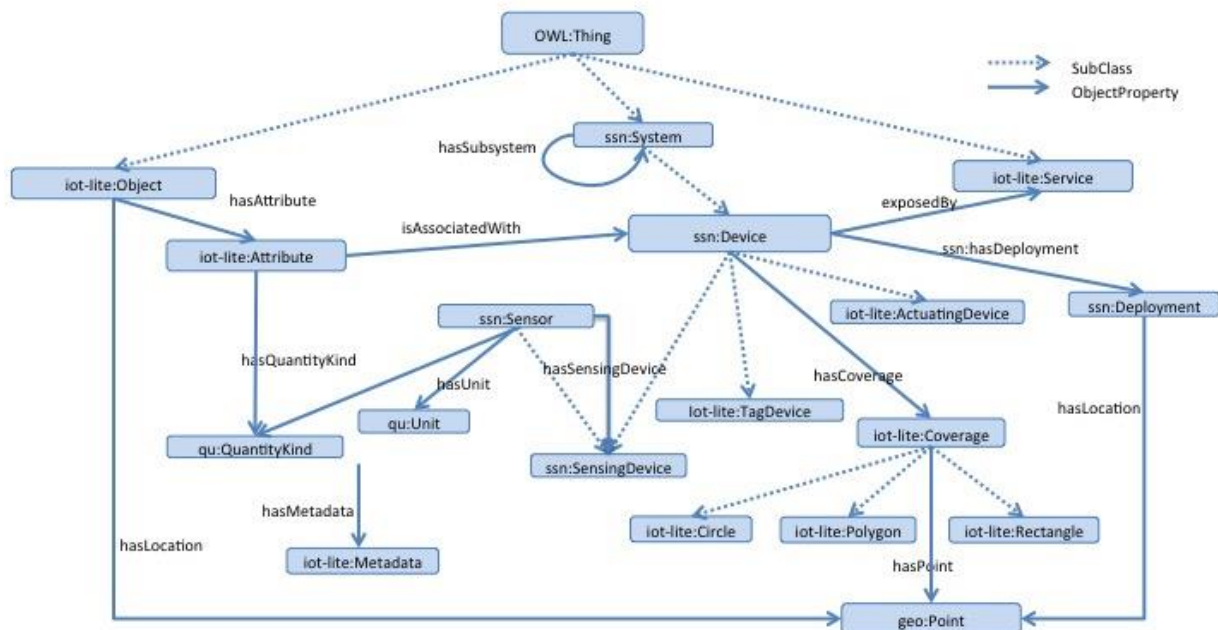


Figure 2-10: IoT-Lite Ontology (Bermudez-edo, Elsaleh, et al., 2015).

The IoT-Lite is meant to be used with a quantity taxonomy, that allows discovery and interoperability of IoT resources in heterogeneous platforms using a common vocabulary (Bermudez-edo, Elsaleh, et al., 2015).

2.2.3.1 Review

The IoT Lite Ontology is a meta ontology designed to be extended in order to represent IoT concepts. It also focuses more on sensing and establishes a high-level concept on actuation, which allows any future developments or adaptations on this area. It is a lighter view of the SSN Ontology, ideal for environments or specific situations in such environments, that require fast and easy processing. It also can be combined with ontologies representing IoT data streams (Bermudez-edo, Elsaleh, et al., 2015) like the SAO Ontology which focuses on representing, semantically, the features of a data stream in the IoT environment (Kolozi, Barnaghi, & Bermudez, 2016).

2.2.4 Discussion

Semantic technologies are viewed today as a key technology to solve the problems of interoperability and integration within the heterogeneous world of ubiquitously interconnected objects and systems (Katasonov, Kaykova, Khriyenko, Nikitin, & Terziyan, 2006). A relevant problem for IoT related semantic descriptions is that they are not as widely adopted as expected (Bermudez-edo, Elsaleh, et al., 2015) and in result of that, the IoT landscape nowadays appears to be highly fragmented. One of the main concerns users and developers have, is that semantics increases complexity and processing time and, therefore,

are unsuitable for dynamic and responsive environments such as the IoT. A model so complete as the IoT Architecture tends to be difficult to apply due to existing various aspects and constraints that the developer must deal with before even started implementing. Some more specific models like the SSN Ontology and IoT Lite, are incisive on the problem they want to solve. This author believes that by developing such ontologies for more specific solutions, or even consider them protocols (to enforce its use), creates a bigger opportunity for the scientific community to excel in the field they wish to improve, instead of dissolving into a solution, usually not giving importance to minor but still important problems, for the immensity that is the IoT environment. Nevertheless, semantic technologies are widely claimed to be a qualitatively stronger approach to interoperability than contemporary standards-based approaches (Lassila, 2005).

In Table 2-1, a comparison of the studied models is provided for better understanding. Some ontologies excel where others present weaker or inexistent solutions, which means that different applications may need different ontologies or a combination of some of them.

Table 2-1 – IoT Models Comparison.

	W3C SSN	IoT-A	IoT Lite
Interoperability	✓	✓	✓
Device Discovery and Management	✓	✓	✓
Scalability	✓	✓	✓
Management of Large Volumes of Data	✓	✓	✗
Security, Privacy, and Integrity	✗	✓	✗
Dynamic Adaptation	✓	✗	✓
Fast Processing	✗	✗	✓
Context Awareness	✗	✗	✗

The concept of combining ontologies is not new and, most of the cases, represents the more suitable solution. But adapting models to each other is an absurd idea because they represent base concepts and ideas to provide guidance for developers and should not be changed or else they would turn every already developed project obsolete. So, to provide a suitable solution, model mapping (or ontology mapping) is briefly described.

As is depicted in Figure 2-11, two ontologies may have different structures to define the same (or similar) concept, in this case the name of a person. These structures can be related using a mapping that considers the differences and describes the possible mismatch between them, providing a way to use both ontologies and integrate them, relying in the identified mismatches (Ferreira, Agostinho, Sarraipa, & Jardim-Goncalves, 2012). This topic is

going to be mentioned in further chapters due to its importance in semantic mapping (i.e. the creation of semantic maps).

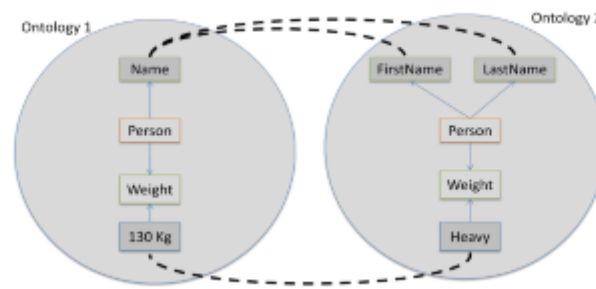


Figure 2-11 - Example of mapping between two ontologies (Ferreira, 2012).

After reviewing the different solutions, previously presented, and briefly introducing the possible mapping between them, the possibility to represent a sensor network in a model gains shape and provides a platform with new features and ideas that improve the capabilities of the network. In the next chapters, important concepts that provide such capabilities are presented, namely sensor configuration, which arises as the basis for the correct integration of the sensors and is the motto for the upcoming concepts related to the semantic maps.

2.3 Semantics and Concepts of Intelligent Sensor Networks

In a broad sense, semantics is defined as the study of meanings. Contextualized in the subjects of this work, semantics and semantic maps are terms that promote standard data formats and allow information to be shared and reused across applications, enterprises and community boundaries (Berners-Lee, Hendler, & Lassila, 2001).

Semantics provide the representation of knowledge and use models, similar to the ones studied and presented previously in this chapter, to create architectures, common data formats and nomenclatures in order to provide a homogeneous environment within the sensor network. These semantics are very useful to determine scalable solution processes that work in challenging contexts of *big data*, where the sources can be too large and heterogeneous for a developer to review all data (Knoblock & Szekely, 2013). The semantic maps are a contextualization of such semantics in the deployment environment, allowing the gathering of important meta-data from the network components and its use to provide redundancy and reliability. That information is further used for developing a more trustworthy system for the network and to build, or apply, concepts that improve the overall performance of the network, or intelligence. Those concepts, of typical intelligent sensor networks, are reviewed during this section.

2.3.1 Sensor Configuration

A sensor network perceives the environment, monitors different parameters and gathers data according to an application purpose (Pathan, Taylor, & Compton, 2010). The capabilities of a sensor network go beyond observing and forwarding raw data, it provides services and has the capability to adapt its functionalities to the environment where it is deployed. But managing today's data networks is highly expensive, difficult and error-prone and often can't rely on the initial design of the network, which does not foresee the important matter of the constraints of the environment (Kim, 2009). For this kind of processes to be easier, proper configuration is required. After proper configuration, to assure proper autonomous adaption, self-configuration of the sensors, to the new data from the changes observed in the environment, is needed.

Configuration in the IoT domain is typically an enhanced upgrade from the usually implemented configuration on stable and specific technological environments. The IoT domain requires easily exchangeable information and autonomous processes, as was previously mentioned, which requires an adaptation from the classic configuration to meet the new requirements. These requirements, in the IoT, should be predicted in the architectural model or ontology that was used to design the network.

Configuration in the IoT has six major challenges, as seen in (Perera, Jayaraman, & Christen, 2013):

- Number of sensors, an ideal configuration should be able to rapidly configure a significant number of sensors autonomously;
- Heterogeneity, devices from different brands usually communicate differently and most devices use different techniques for measurement and overall functioning;
- Scheduling, Sampling Rate and Communication Frequency, usually determined by the user's requirement and relative to the frequency (or opportunity) in which the sensors need to generate data or exchange information;
- Data Acquisition, correct methods of measurement result in better efficiency;
- Dynamicity, capability of adaptation of the network of sensors to the environment;
- Context, sensor data by its own is meaningless and needs to be analysed and contextualized to better understand the environment.

Configuration can also be divided into two possible levels: sensor-level and system-level:

- **Sensor-level configuration** aims to improve the sensor's efficiency by tuning software parameters like sampling rate, data communication frequency and sensing schedule. This kind of parameters can be self-adjusted using retrieved external information to

determine the best configuration for the environment in which the sensor is deployed. For this, some sensor networks were developed with additional sensors for detecting such situations (this topic is discussed in a further chapter). This configuration is limited to the sensor's basic configurations, what takes us to the system-level configuration.

- **System-level configuration** is based on the configuration of sensors and data processing components according to the user and system requirements. In other words, it includes the configurations files and program codes, usually manually defined, by the user. Some architectural models were already developed to enhance this configuration like the CASCoM Architecture (Figure 2-12).

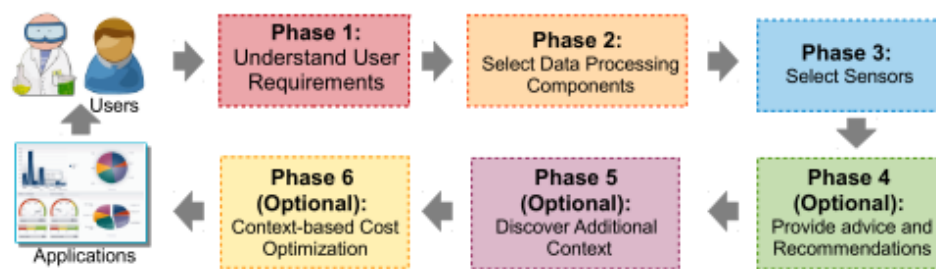


Figure 2-12 - Context-Aware Sensor Configuration Model – CASCoM (Perera, Zaslavsky, Compton, Christen, & Georgakopoulos, 2013a).

This model helps non-IT experts to configure sensors and data processing components using a single-click, quickly and easily (Perera, Zaslavsky, Compton, Christen, & Georgakopoulos, 2013b). Many complex semantic concepts play a significant role behind the system, but the idea is to make this process as easy as possible. This model is a valid approach on how a system like this should function, but only focusses on the system-level. First, after establishing the desired sensor network, a graphical user interface should be developed. This interface should function with a question-answer approach and provide a list of available tasks to perform, regarding the user input requirements. After that, the system would search for programmed components that allow the task and find sensors to produce the input requirements. If it fails to fulfil any of these previous tasks, it should provide an error information (for example, insufficient resources) and additional information for future implementations or similar possible solutions. If it succeeds, one or multiple solutions should be provided, with the associated costs, and the user chooses the final solution to apply. With the final solution, the system generates the configuration files and program codes. The sensing information retrieved is provided to the user and additional adjustments may be applied by the user or by the system, using previously developed preferences.

With the development of the IoT, users will face increasing challenges in managing larger numbers of IoT devices (Rose Jaren, Eldridge Scott, 2015). Efficient sensor network system configuration allows to considerably reduce processing time and improve general quality of information retrieved by the sensors. The processes of this network clearly beneficieate by using automatic adjustments and environmental adaptations that rely on

previous and continuous retrieved information. However, for this to work, models like the CASCoM, that allow to reduce time required for configuration of data processing mechanisms in IoT middleware, need to be further developed and implemented, not only in the system-level, but also on the sensor-level. This way, sensors will be able to self-configure, self-adapt, be context-aware without human intervention, be exchangeable and be able to perform tasks more accurately, in the highly dynamic smart environments of the IoT paradigm.

2.3.2 Context Awareness, Self-Configuration and Self-Adaptation

As explained in previous sections, semantics can play a role in assisting users to manage and query sensors and data. Indeed, as the scale and complexity of sensing networks increases, machine interpretable semantics may allow autonomous or semi-autonomous agents to assist in collecting, processing, reasoning about and acting on sensors and data (Compton et al., 2011). Configuration itself, when changed, may lead to inconsistent states resulting in operational failures and inefficiencies (Konstantinou, Florissi, & Yemini, 2002). Thus, some concepts arise as technology improves and to make valid assumptions about the models and configurations of sensors within IoT today we must understand those concepts. Some of them are context awareness, self-configuration, self-adaptation and the concept of intelligent system (in this context).

Context awareness refers to the property of a device to passively or actively determine its context. When talking about location awareness, is evident that a device only needs to determine its location (e.g. via GPS). Nevertheless, when talking about contexts, some of the times, the context itself is not clearly defined or is too dynamic. With the evolution of systems, we reached a point where automatization is key to develop a fast and efficient system, without the need for constant human supervision. That is where the context awareness is a plus for sensing systems in the IoT. If we find ways for the sensors and system to understand the surroundings, by previously creating methods to do so, they can adapt and improve their functionality. This way, we create a truly dynamic system. In Figure 2-13, an example of the context aware concept is provided, regarding an example for a context aware mobility solution provided by the company Cisco.

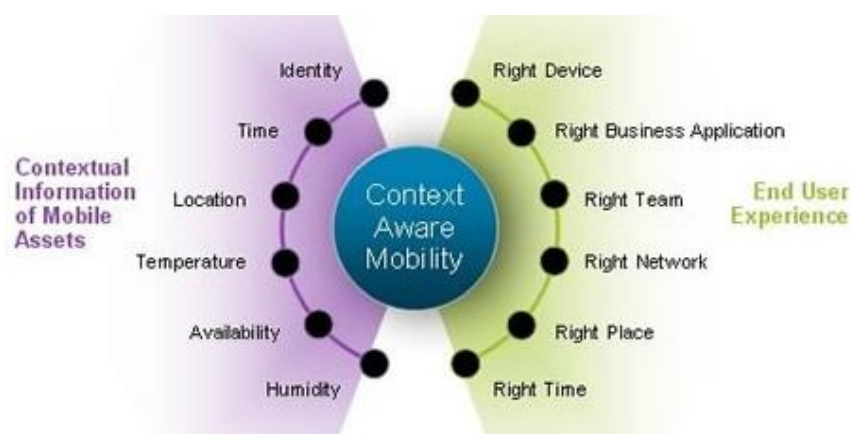


Figure 2-13 – Context Aware Concept (Cisco, 2008).

Due to the heterogeneity of devices – including sensors, actuators, storage devices, utility monitoring devices, mobile phones, network elements, and computers – and the sheer number of devices that are being connected to the internet under the IoT domain, remote or cloud-control appears to be a daunting task destined to suffer from limited scalability (Athreya, DeBruhl, & Tague, 2013). Hence, the natural direction for the IoT devices is to manage themselves using dynamic configuration, both on a software and a hardware level, and improve their resources utilization and adaptation.

Manual configuration of high numbers of nodes in a network is either impossible or highly costly, so it is desirable for the nodes to be able to configure themselves (Guardalben, Villalba, Buiati, Sobral, & Camponogara, 2011). **Self-configuration** is based on the user's specification and includes the methods for generating initial configurations for the sensors and the programming codes to apply to the system (or network). These methods can also provide guidelines to the system for maintaining stability through all the system's process.

Another relevant concept is **self-adaptation**. Today, systems must cope with variable resources, system errors and changing user priorities, while maintaining, as best as they can the goals and properties envisioned by the developer (Garlan, Schmerl, & Cheng, 2009). Self-adaptation, that is bind to the term self-management, is the capability of an entity (sensor or system) to calibrate its functionality by tuning some his parameters to correspond to the environment where it was deployed. This concept is applied to function after the configuration, including posterior self-configuration at some degree, and ensures correct functionality and management for the duration of the deployment of the device, or system. The figure 2-14 shows a self-adaptation process in the context of automotive embedded systems with the purpose to provide an illustration to the generic approach to this concept.

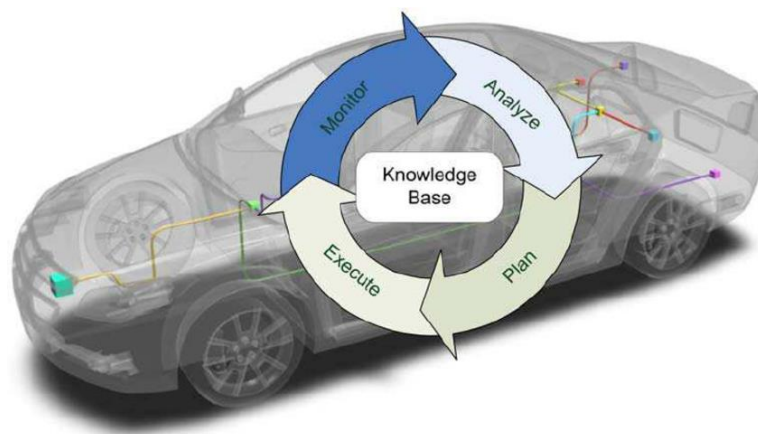


Figure 2-14 – Self-Adaptation Concept (Weiss, Zeller, & Eilers, 2010).

Systems that function based on some kind of decision-making capability or use communication to exchange data are often called **Intelligent Systems**. The terms and concepts previously mentioned are ways of improving these systems, each one with their advantages and disadvantages. Today's market thrives on efficient, fast and cheap technology and sometimes, due to bad implementation or short time for planning, security and stability of systems is despised. By building models and common architectures for technology, using

some of the concepts mention before, aspects like security, stability, portability and easy communication can be achieved and serve as platform for prospering ideas and solutions.

2.4 Dynamic Network Communication

The interconnectivity of computing and physical systems, can become “the nightmare of ubiquitous computing” (Kephart & Chess, 2003). The generated data from IoT devices is a challenge to data management and contributes to the emerging paradigm of *big data*. One of the challenges before collecting and processing information from these devices is discovering and configuring the sensors and the associated data streams.

In the context of IoT, automated discovery mechanisms and mapping capabilities are essential to network management and needed for overall communication management. Without it, the network management capabilities cannot scale, be accurate or efficient since it needs to automatically assign roles to devices based on intelligent matching against pre-set templates and attributes. It needs to automatically deploy and start active or passive performance monitors based on assigned roles and attributes, start, stop, manage and schedule the discovery process and make changes to any role or monitoring profile at any time, or create new profiles as required (Vermesan et al., 2009).

A sensor configuration process detects, identifies and configures sensor hardware and deployment platforms in a way that allows the software systems to retrieve data from sensor’s hardware when required. Communication plays a major role in this situation, allowing not only that kind of communication but also communication between the sensors, improving the performance of the system and further adaptation to the environment of the configuration during the deployment.

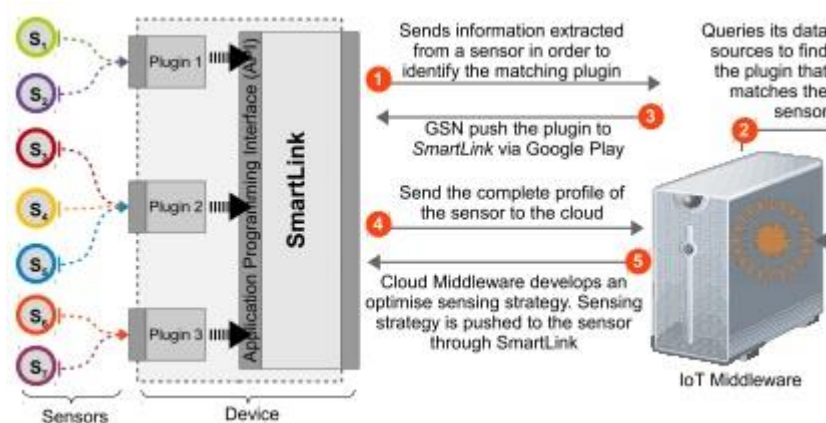


Figure 2-15 - System Architecture of the CADDOT model (Perera, Jayaraman, et al., 2013).

Smartlink, integrated in the CADDOT (Context-Aware Dynamic Discovery of Things) architecture model (Figure 2-15) is an example of a developed solution for this kind of problem. It is aimed at discovering and configuring sensors, being capable of function with

sensors deployed based on a specific location despite their heterogeneity (for example, different communication protocols, communication sequences and capabilities).

The system architecture presented above is composed by three main components: the sensors, the SmartLink tool and the cloud middleware. The Smartlink tool presented is just a mediator between the sensors and the middleware, which looks to improve the information exchange, the capability of the network and the system's organization.

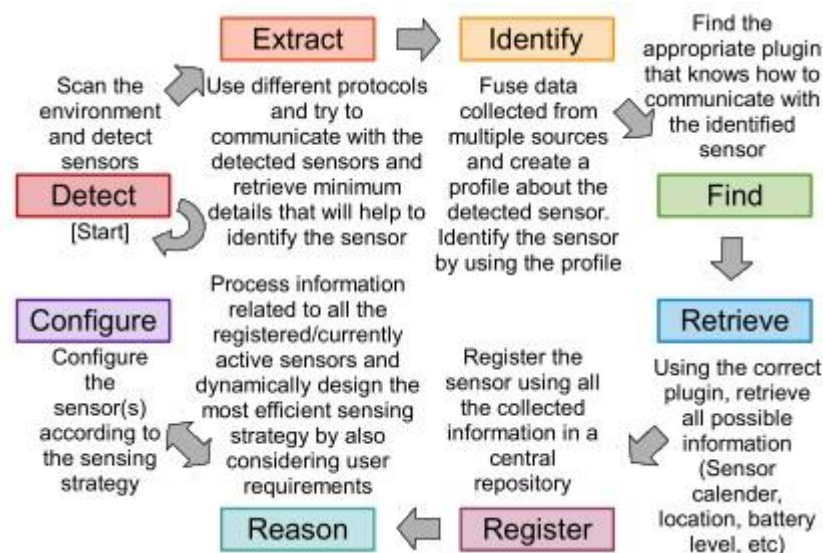


Figure 2-16 - CADDOT Model for Sensor Configuration (Perera, Jayaraman, et al., 2013).

Figure 2-16 depicts the approach of the CADDOT model for sensor configuration, on which the SmartLink tool represents a major role. The SmartLink serves as an open wireless spot, so sensors can connect to it using an ad-hoc approach (Detect), extract information from the sensors detected (Extract), send information related to the new detected sensor to the middleware (Identify), receive plugin information related to the sensor from the middleware (Find), retrieve the sensor's information with direct communication using the downloaded plugin (Retrieve) and provide the sensor's full information for registration on the middleware (Register). After that, the stages Reason and Configure are responsible for processing the information of the sensors in order to optimize its functioning.

This system is just an example of this kind of systems and may vary in other models, which usually present different methodologies and purposes.

Other important concepts, more related to the communication itself are: Network Mapping, Network Discovery and Networked Control System. They are frequently referred in models like the one that was mentioned before and are presented here for greater clarification and understanding of the concept of Dynamic Network Communication.

Network Discovery - Consists in uncovering, within a network, the subset of devices or nodes that exhibit a particular type of attribute or preference (Yee et al., 2013). This computer activity is typically used to discover the user's names and information like groups, sharing preferences and services.

Network Mapping - It is the study of the physical network connectivity. Network Mapping is used to find the devices that belong to the network (or were recently added) and to understand how they communicate. It usually defines the process before Network Discovery, because it only finds the information about what devices are connected to a network and what operative system/configurations they run.

This concept is relevant to this work because by creating an autonomous system, the idea is to read data from the environment and to adapt efficiently. Some common situations that induce this behaviour are not only environment variations but also failures and errors that may occur in the network. By recovering from such situations, the system is truly autonomous. The role of this concept is to understand and monitor the physical components of the network, the devices and connections, in a way that is possible for the network to understand and get feedback from their performance and functioning.

Networked Control System - This type of system is divided into four components:

- Sensors, responsible for retrieving information from the environment;
- Controllers, capable of forwarding commands and decision-making based upon the collected data;
- Actuators, which execute the actions requested by the controllers;
- Communication Network, that allows information exchange.

The most important feature of the Networked Control System (NCS) is the connection between the virtual and physical space, allowing the execution of various tasks remotely and sometimes autonomously. It also reduces the complexity of physical connections and cost in design and system implementation. Some types of communication typically used are Fieldbuses, Ethernet and Wireless (like for example, Bluetooth and ZigBee).

2.4.1 Discussion towards Semantic Maps

Communication plays a major role in a dynamic network, as was mentioned before, due to necessity of constantly respond to the changes monitored in the environment. The advantages of preparing a network, using proper configuration, to be dynamic relatively to the environment where it was deployed far exceed, in most cases, the rigid and specific network configurations, due to the autonomy and adaptation processes needed.

The configuration and the related processes, subjects of this research, emerge as a foundation to the dynamism and context-intelligence of an IoT network. Due to necessity of truly understanding the configuration processes, the IoT models, mentioned earlier in this chapter, provide information on how much the efficiency of any dynamic intelligent network relies on the design, standardization of configuration and reference architectures. All that

concepts and methodologies contribute to a great part of this work, the creation of semantic maps.

Semantics is the study of meaning (Merriam-Webster, 2017). The items defined by these meanings can be grouped to what is called a semantic field. Thus, the semantic field represents a lexical set of semantically related words that are related due to different aspects such as context, place or activity. The interest that drives the use of this concept in technological environments is the need, as mentioned before, to make information more computer friendly and therefore, provide the context and meaning to the words that the computers lack to have internal representation of.

RELATIONS	
<i>is-a</i>	<i>has</i>
<i>part-of</i>	<i>is-equal-to</i>
<i>is-about</i>	<i>is-similar-to</i>

Figure 2-17 - Example of Relations (Paiva, 2015).

Therefore, semantic mapping can be viewed as a strategy to represent concepts or technological instances. The semantic maps that result from the process are relations of a certain concept, by the definition of semantics. Some basic relations in semantics are shown in Figure 2-17 and they represent the basis for associating concepts and meanings. There can be different associations for any concept, like associations of class (the order of things the concept falls into), property (the attributes that define the concept) or example (examples of the concept) (Estes, 1999). In technology, since the internal representation of information gathered by technological components is not intuitively understandable by humans and vice-versa and it is also inadequate for fast learning processes, the combination of object classification and common-sense knowledge with semantic maps, represents an interesting approach to provide this type of information.

Thus, the information provided in the configuration and representation of meta-information of the components of the network in semantic maps (that will be much more detailed in the next chapters) can provide expressive information, contextual integration of the observations and correlation of the knowledge of the environment (W3C, 2012). These semantic maps become, in a way, a resource, to store knowledge information that can be used for various tasks such as redundancy, fault detection, network monitoring and error recovering. With this, associated with the concepts mentioned in this chapter, communication improves, due to the homogenous components within each reference architecture, and the concept of IoT gains shape, providing a more efficient, resourceful, interpretable and trustworthy network.

In this chapter, the architecture for the module developed in this dissertation is presented. The first section is an overview of the contents of this module, the second section delivers a description of the main concepts that originated and provided basis for this work. After that, the architecture is explored, and the contents of each module are explained. The last section contextualizes the project in an application scenario that briefly shows the possibilities for the developed solution.

3.1 Overview

Generally, systems that rely on processing the information of deployed devices and their analysis, to read or predict conditions that could trigger pre-determined rules, recur to the integration of a Complex Event Processing module. A Complex Event Processing (CEP) module analyses large flows of primitive events received from a monitored environment to timely detect situations of interest (Margara, Cugola, & Tamburrelli, 2014). This processing takes place following user-defined rules and that aspect induces a liability to the dynamism of the general system, because any change may cause an incompatibility to adapt or, in the case of a device failure, the rules may become obsolete due to a non-implementation of redundancy.

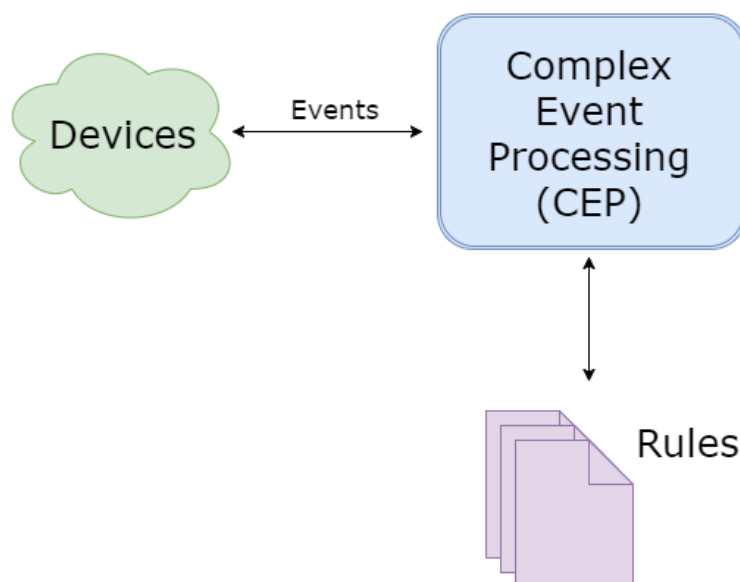


Figure 3-1 - CEP Generic Concept.

As presented in Figure 3-1, a possible CEP generic concept implementation, the CEP module listens to incoming events generated by the devices and, following the pre-

determined rules (created by the developer), detects situations that may trigger events on the devices or simply alerts the system monitor.

The idea behind understanding the CEP is to perceive its importance in the application of the models studied in chapter 2, and to provide the intended methodology that this dissertation aims to propose, the use of semantic maps. These semantic maps, as mentioned before, are intended to improve and test the trustworthiness of the system by standing as a tool to overcome errors and failures in devices or, more specifically, in sensors. The idea for the functioning method of these maps is to be a representation of similar configurations for the sensors, therefore **an alternative mapping for the CEP module to listen to the network of devices and retrieve the information needed to use the existing rules.**

Recurring to a brief example, let's say that the sensor A is damaged by unknown reasons and provides unusable information (for example, absurdly high temperature readings), as depicted in Figure 3-2. Disregarding the detection of such failure, that will be mentioned further in this chapter, the sensor A stands as a liability to the event processing and may cause several rules to become obsolete for not having the necessary information to be triggered. In this way, the system is prone to fail to comply to the initial expectations for its correct functioning. To tackle this situation, usually, human interaction is needed to provide a solution for this problem. An implementation of another temperature sensor or the use of a temperature sensor that is already present in the network are options that the system monitor may find to solve this problem. In large networks, in industrial environments, there are numerous sensors, deployed for many different uses. So, regarding that aspect, it makes sense to believe that using other sensors to solve this problem is a possible and common solution.

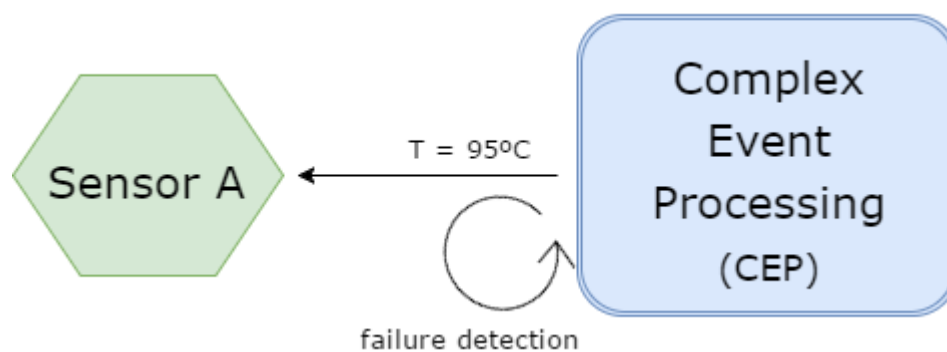


Figure 3-2 - Example of Detection of a Sensor Failure.

To find a possible sensor, one must search another temperature sensor that shares the same, or similar, localization and is apt for providing the same functionality for the role that was assigned to Sensor A. In this example, sensor B is a match. Of course, in real scenario situations, the specifications of the sensors may dictate an adaptation from the output that the CEP receives from this Sensor B but, although we mention that aspect further in this thesis, in this example that situation is disregarded.

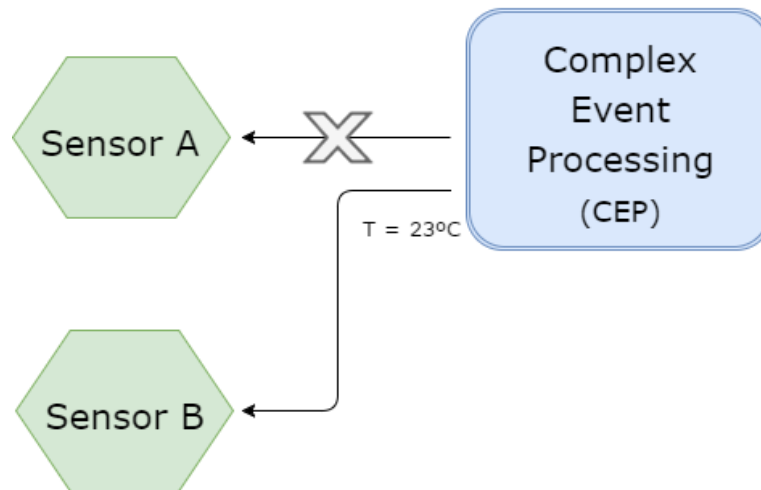


Figure 3-3 - Example of Solution to a Sensor Failure.

In Figure 3-3, it is depicted that the Sensor B provided a redundancy to the sensor A. This is a very simple solution, but is not automated in any way, that means that requires human intervention and lacks autonomous dynamic adaptation.

Other example of failure is depicted in Figure 3-4, in this case, the method to detect the failure or malfunctioning is inconsistency by proximity comparison, a fault detection method to be exploited in chapters 4 and 5.

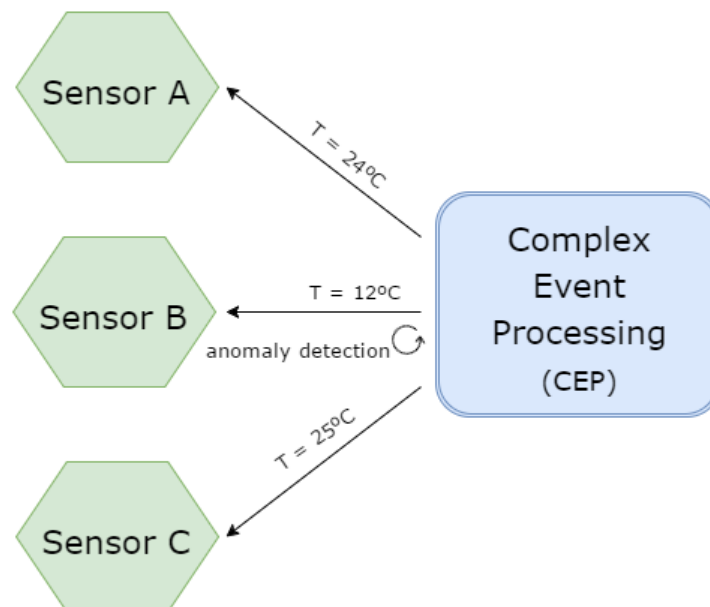


Figure 3-4 Example of Sensor Malfunctioning.

Sensor A, B and C, are all temperature sensors of which the CEP retrieves data to function, as explained previously. Sensor A and C have consistent similar output values, but Sensor B does not. It has a significantly lower value, despite not being absurd, and it may trigger some sort of event that is not determined for the current surrounding environment. This kind of anomaly detection can be trickier, but it's also important and may benefit from the solutions presented in the previous example.

In an industrial implementation, the failure of Sensor A (in the first example) or Sensor B (in the second example) may not even be notice among the high quantity of sensors readings, and that may imply an incapacity to solve the problem within an acceptable time window. This is where semantic mapping comes into play, because it represents and maps possible redundancies while providing room for developing the autonomous creation of possible maps regarding the aspects of the role of each sensor in the network. This autonomous creation of possible mappings is possible, but it always requires an initial manual configuration of the device database and relevant possible mappings by the developer, to provide consistency to the network. After the use of this methodology, the module responsible for such processes, delivers information for the CEP to update its rules database. In this way, provides a degree of autonomous functioning to the network with the capability of redundancy relative to the failure/malfunctioning of devices.

Before defining the module that may contain the semantic mapping mentioned before, it is important to understand what the semantic map needs to provide and what specifications may be important to distinguish in the sensor's role in the network.

Therefore, this dissertation proposes the following aspects for establishing associations between devices for the semantic mapping:

- Identifiers (ID's);
- Instances or Elements in the Association;
- Type of Relation, Association or Mismatch;
- Role from the original Instance in the Network;
- Output data differences between Instances;
- Importance Weights in decision-making.

The mentioned identifiers serve the purpose of the structural organization of the mapping and the sensors involved, and represent the individual unique attribute that differentiates each instance. The use of the letter "O" defines the origin sensor and the use of the letter "D" defines the destination sensor, considering the configuration targeting made by the semantic map. Naming the mapping can also be relevant and it is considered in the mapping equation, Equation 1, presented further.

The instances or elements in the association are the origin and destination sensor (considering that the sensor that precedes the mapping is the origin sensor and the one that succeeds the mapping is the destination sensor). It is important to mention that that can only be **one** origin sensor (when many fail, individual mapping must be considered) but it could exist **multiple** destination sensors, because a specific sensor may have a complex task or, using the previous example, it may not exist a temperature sensor to serve as destination

sensor and a combination of multiple sensors may provide the needed measurement that the origin sensor provided.

The type of relation, association or mismatch is a representation of the difference between each instance in a way that highlights the aspects that may have to be considered while using the maps. For better clarification purposes, this aspect is referred simply as mismatch. An example of a possible mismatch is an encoding situation, where the origin sensor measured the temperature in the Celsius scale and the destination sensor measures temperature in the Fahrenheit scale. The mathematical explanation for this kind of situation (where in this equation field, is simply named) is then complemented with one of the other parts of the equation, mentioned in the next paragraph, the output data differences between sensors.

The output data differences between sensors is the relation of output that each sensor may have, regarding its specifications, and that should be considered when analysing the data (sensors may have, for example, different measuring scales or the need to perform different multiplications to their voltage output). The output data field for the destination sensor(s) can have information, normally an expression, using multiple sensors (e.g. recurring to average or median values with different importance weights regarding the distance to the point to measure) when the mapping is from one origin sensor to multiple destination sensors.

The role from the original instance is very important, perhaps the main factor to be considered, because it represents the functionality that the origin sensor has in the network. The destination sensor does not need to have the same role, before the map is applied, but it should comply with the aspects that the role requires. And after the use of the semantic map, the role is added to the information of that sensor in the network database of sensors (explained further in this section) in order to, in further mapping, exist the necessary knowledge to use semantic maps in which the previous destination sensor is the new origin sensor.

The importance weights in decision-making is a way to differentiate several mappings for the same sensor. If a certain sensor has different semantic maps, the one with higher weight should be considered. This is possible by attributing a value that specifies the importance or effectiveness of the map and by dynamically increment or decrement that value during the functioning of the system, depending on several occurrences. For example, a sensor may have too many roles assigned and to avoid too much reliance by the network (in case of failure, requires excessive processing to treat each one of its assigned roles, individually), the value may be decremented in order to avoid further mappings to assign it as destination sensor. Other example may be that a certain sensor does not have any semantic maps associated with it, in which it is the origin sensor (in case of failure, it is not automatically recoverable by a semantic map). In that case, the use of the mapping process to that sensor should be prevented and the weight of semantic maps, that involve it as destination sensor, should be decremented.

Semantic Mapping Equation:

$$\text{MapS} = \langle \text{ID} (O, D[i]), \text{Mismatch}, \text{Role}, \text{CorrelationData}, \text{Weight} \rangle \quad \text{Equation 1}$$

The provided semantic mapping equation (Equation 1) is a summarization of the relevant fields in the semantic mapping instance, designed in this dissertation. To better understand this equation, some additional information and a brief example is provided by Equation 2.

$$\text{AtoBC_FireAlarm} = \langle (1, [2, 3]), \text{Encoding}, \text{"FireAlarm"}, \text{"(B+C)/2=A*9/5+32"}, 1 \rangle \quad \text{Equation 2}$$

In Equation 2, a concept example of an application of the semantic mapping equation is provided. It does not specify the concrete application because the semantic mapping structures are not yet defined, something that will occur further in this text. With this, it is meant to explain that to define a certain field, like the *CorrelationData*, sometimes it is not sufficient to use only an expression. Other aspect is, the equation does not follow the standard presentation of an equation (e.g. multiplication and division correct mathematical symbols) because it is presented as it would be in a typical implementation, using a string variable.



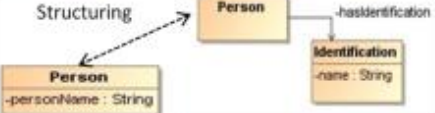
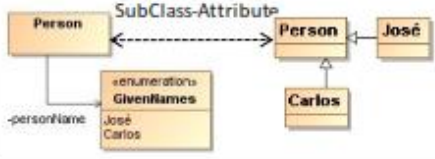
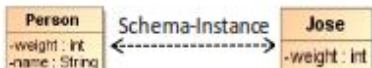
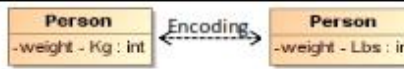
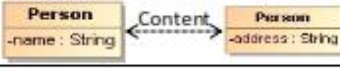


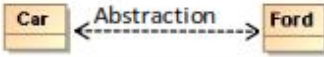
The ID (1, 2, 3) and Name (AtoBC_FireAlarm) fields don't have too much to explain and are specific to the application of the mapping. The ID=1 defines the origin sensor (A), the ID=2 and ID=3 define the destination sensors (B and C).

The mismatch field should not be ambiguously specified by a string variable and should correspond to a specific limited set of predicted possible relations. Those relations do not need to be equal to every application but should be well defined within the boundaries of it. Some possible mismatches are presented in Table 3-1. This table of mismatches belongs to the work of (Ferreira, 2012) which was based on the work of (Agostinho, Sarraipa, Goncalves, & Jardim-Goncalves, 2011), and describes them and provides an illustrative example. Each mismatch is defined as lossless when the relating element can fully capture the semantics of the related element or as lossy if a semantic preserving mapping cannot be built (Agostinho, Malo, Jardim-Goncalves, & Steiger-Garcia, 2007). In the example, provided by the Equation 2, Encoding refers to the different measurement scale between A, B and C, where B and C measure temperature in a Fahrenheit scale and A measures temperature in a Celsius scale.

The role in the provided example is "FireAlarm" and specifies that the origin temperature sensor is used to determine whether a fire alarm should be activated or not. The

name used in this field should be direct and simple, to provide an easy grasp of the role in question.

Table 3-1 - Semantic Mismatches (Ferreira, 2012), based on (Agostinho et al., 2011).

Mismatch	Description	Examples
Lossless	Naming Different labels for same concept of structure	
	Granularity Same information decomposed in or composed by (sub)attributes	
	Structuring Different design structures for the same information	
	SubClass-Attribute An attribute, with a predefined value set represented by a subclass hierarchy (or vice-versa)	
	Schema-Instance An attribute value in one model can be a part of the other's model schema (or vice-versa)	
	Encoding Different formats of data or units of measure	
Lossy	Content Different content denoted by the same concept	
	Coverage Absence of information	
	Precision Accuracy of information	
	Abstraction Level of specialisation	

The correlation data field specifies the difference already mentioned on the mismatch field as “Encoding” between the temperature scales of Celsius and Fahrenheit, and has the mathematical relation between the two scales (Equation 3) implied in the expression.

$$T(^{\circ}\text{C}) = (T(^{\circ}\text{F}) - 32) \times 5 \div 9 \quad \text{Equation 3}$$

The correlation expression, on the left side, uses the reading of the temperature B and C to provide a single reading, as it was when only the Sensor A was used by the CEP. That

resulting reading is an average of the two readings, possibly indicating that they are a bit farther from the point where Sensor A was deployed and where the measurement was meant to be done. To get the value of the reading, the expression must be solved in order to A, the only missing value (B and C represent the readings from Sensor B and Sensor C).

Finally, the weight value is initially set as 1 and may be changed during the functioning of the network as explained previously.

Other field, that was considered during an early version of this equation, but it was excluded from this equation was the rule expression, that defines the situation in which the fire alarm is activated. In this case, for example, it would be when the temperature, measured in Celsius was higher than 35 degrees. But this field would be redundant, because the module responsible for this comparison and process is the CEP. The equation, or module responsible for the implementation of the equation, should not be responsible for the acknowledgment of the validation of the rule expression.

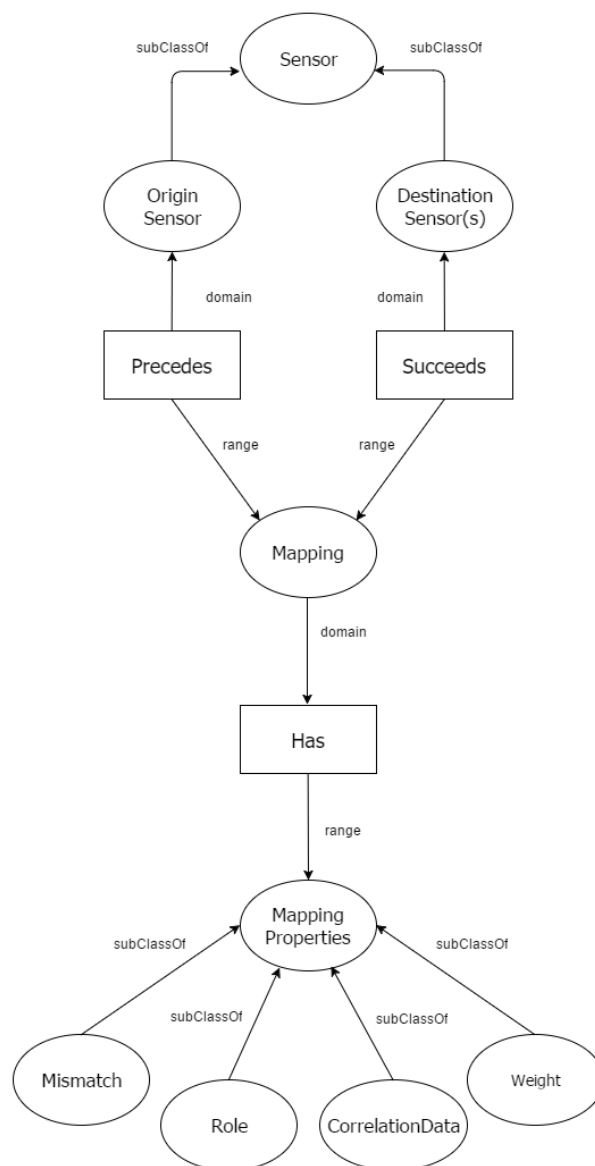


Figure 3-5 - Semantic Mapping Model.

Considering this, the mapping equation stands as a tool to provide information about the existing semantic maps, in a generic way, that can be used to update the rules database from the CEP.

With the information provided previously, we can now construct a model of the semantic mapping methodology (Figure 3-5).

This model (Figure 3-5) is specified in order to comply with some implementation aspects, to be clarified during the next chapters. The main ideas to mention are that the *Origin Sensor* and *Destination Sensor(s)* are sub-classes of the class *Sensor* and the *Mapping Properties* class has, as sub-classes, the fields mentioned before in the semantic mapping equation (Equation 1). The entities *Precedes*, *Succeeds* and *Has* are ontology properties, that are related using the *domain* and *range* specifications. The entity *Mapping* is also a class.

In an attempt to better clarify the description and nomenclature given above, this ontology model specifies, for example, that a given *Origin Sensor* (sub-class of *Sensor* and domain of the property *Precedes*) precedes a certain semantic mapping instance (represented by the class *Mapping*, range of the property *Precedes* and *Succeeds*). That certain semantic mapping instance will provide a *Destination Sensor* and that process is specified with the property *Succeeds*, in an inverse way of the mention before with the property *Precedes*, where the domain of *Succeeds* is the *Destination Sensor(s)*. The mapping instance uses the ontology property *Has*, in its domain, to imply that contains the class *Mapping Properties*, available at its range. Although this example may fail to perfectly clarify the given model (Figure 3-5), it will be easier to perceive it in the upcoming chapter regarding the implementation of this model.

Another important aspect, before unveiling the designed conceptual model for an example architecture that includes the semantic maps, is the use of a device knowledge base and a mapping knowledge base (Figure 3-6). The definitions for the concept of knowledge base is mentioned in the sub-section 2.2.1 and its use is based on the necessity of keeping an updated record on the relevant aspects of the sensors, including their current roles, and having a knowledge base where the mappings are fully specified. The mapping knowledge base is designed to be outside of the suggested model, as the device knowledge base is, because it stands as a dynamic record of the mapping information and does not have any particular relevance for other decision-making processes. These databases (knowledge base being a particular type of a database), are to be accessed only by the semantic module and updated by it, every time a change is made to the network (for example, excluding a damaged sensor, the semantic maps associated to it and updating the destination sensor with the new role in the network).

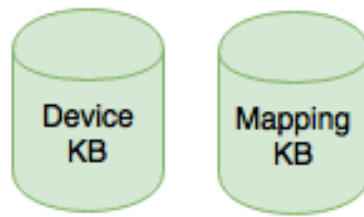


Figure 3-6 - Device KB and Mapping KB.

Finally, the system architecture is presented (Figure 3-7). The semantic mapping module, named **SMAP** (from Semantic MAPping), interacts with the user, devices, mapping knowledge base, device knowledge base and the CEP module, specifically the CEP engine.

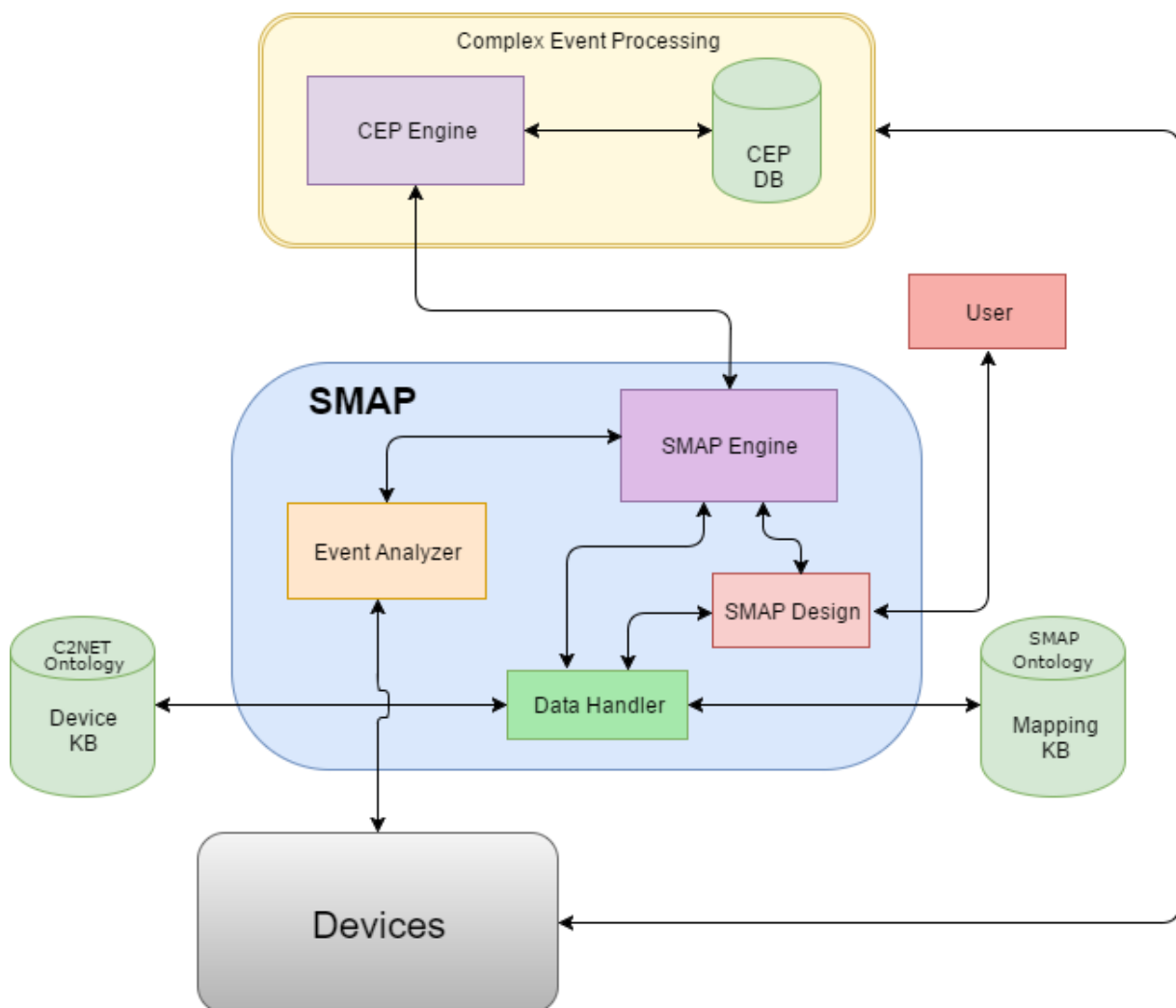


Figure 3-7 - System Architecture.

The verification of devices and the respective measurement events that reach the CEP engine, a feature mentioned earlier in this section, is assured and also treated inside the SMAP module by a fault detection process of one of its sub-modules. Along with this feature, the already mentioned interaction with the device and mapping knowledge bases provides the integrity of the information during the functioning of this module. The CEP is depicted (in

Figure 3-7) and has the CEP engine and CEP database, merely to indicate the existence of the database of events (with the user-specified rules) and that the SMAP module interacts specifically with the CEP engine, not needing to interact with that database in particular (a responsibility of the CEP engine). The sub-modules of the SMAP module are the *SMAP Engine*, the *Event Analyzer*, the *Data Handler* and the *SMAP Design*, detailed further in this dissertation.

3.2 Concept Proposed

To better comprehend the general operation of the semantic mapping module (SMAP), this sub-section defines high level diagrams that expose the flow of configuration, functioning and processes between each element of the system's architecture. First, a high-level activity diagram of a generic system, that has the semantic mapping module integrated, is presented. After that, a sequence diagram roughly provides the essential interactions, that are implemented and described further in this dissertation, between the elements of the mentioned generic system (CEP, SMAP, user, mapping and device knowledge bases). Along with the diagrams of the generic system that help to explain the runtime processes, the process of the semantic map design (or semantic mapping) is also presented using the same type of diagrams.

In Figure 3-8, the activity diagram for a system's process is presented. It is a higher level of abstraction compared to the diagram represented in Figure 3-9, but it provides an initial grip of the flow of processes. In this diagram, the process is initiated in the system itself, that in this case represents the architecture presented in Figure 3-7, without the low-layer level, the *Devices*. The system, in this case mainly referring to the CEP, uses a certain sensor, sensor A, for a specific role. During its functioning, the sensor's output is analysed by the SMAP module, also part of the mentioned system, and the CEP itself. And if it is functioning correctly, the sensor A will continuously provide data to the CEP. If not, the system should recur to semantic mapping and apply redundancy to that specific sensor. To accomplish such feature, it will query (or search) the device knowledge base to retrieve information about the origin sensor, more specifically its roles. After that it will query the mapping knowledge base to find possible semantic mapping solutions for the mentioned sensor. If it does not find any, for one or more current uses of the sensor, manual intervention is required to solve the problem and the functioning of the system should continue after that. If a mapping solution exists, the information about the destination sensor, or sensors, is retrieved from the databases and used to apply this new solution to the role, previously performed by Sensor A. The system should then use this new solution to retrieve the necessary data it requires.

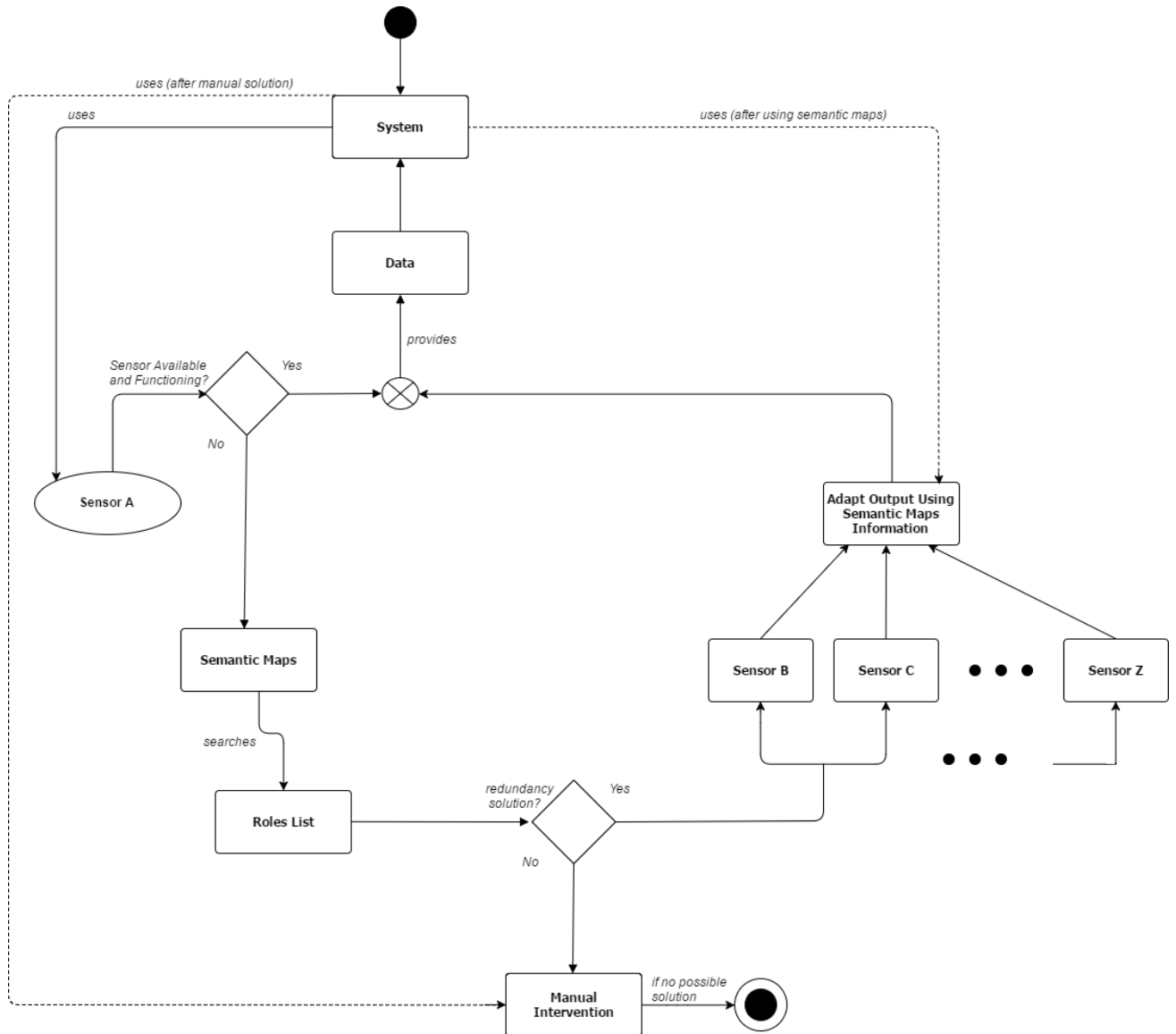


Figure 3-8 - Activity Diagram of the Architecture's Process.

Figure 3-9 depicts the sequence diagram of the system's runtime process. This diagram provides a more specific and accurate description of the processes that occur when using semantic mapping.

The elements of this diagram are the CEP, the SMAP, the mapping knowledge base and the device knowledge base. In the beginning of the sequence, it is shown the loop process, carried out by the semantic mapping module, within the sub-module called Event Analyzer, that listens to the events generated by the devices and compares them to the presumable values that the devices measure. This event listening occurs until a verification of a specific sensor functioning assumes that the device is not working properly. Some methods used to accomplish this are direct comparison with similar sensors, comparison of values by geographical interest or proximity distance and physical or expected thresholds. When a device failure is detected, the SMAP Engine takes control and activates the Data Handler sub-module to do a query in the device knowledge base to search for the device's information and

to update the database entries with its new state (not working properly, in this case) to avoid further mapping to that sensor. The SMAP Engine then receives the requested information and elaborates a request for the Data Handler to query the mapping knowledge base, searching for suitable semantic maps. The existing semantic maps (based on the role of the origin sensor) are then requested and the mapping, if any, with higher weight is provided back to the SMAP Engine. If there are no mappings relevant to apply redundancy to the origin sensor, as it was described in the activity diagram presented in Figure 3-8, manual intervention is required. When choosing a certain map, the weight field of the map may be updated to avoid situations such as too much reliability of the system on a single sensor. By now, the SMAP Engine knows the destination sensor, or sensors, and requests the Data Handler for its specific information, that uses a query in the device knowledge base, once more updating the entries with the new states and roles changes that the current mapping has caused. Finally, the SMAP module uses the mapping information to apply the correlation between the sensor output of the origin sensor and destination sensor and sends that information to CEP in order to update its current rules and events.

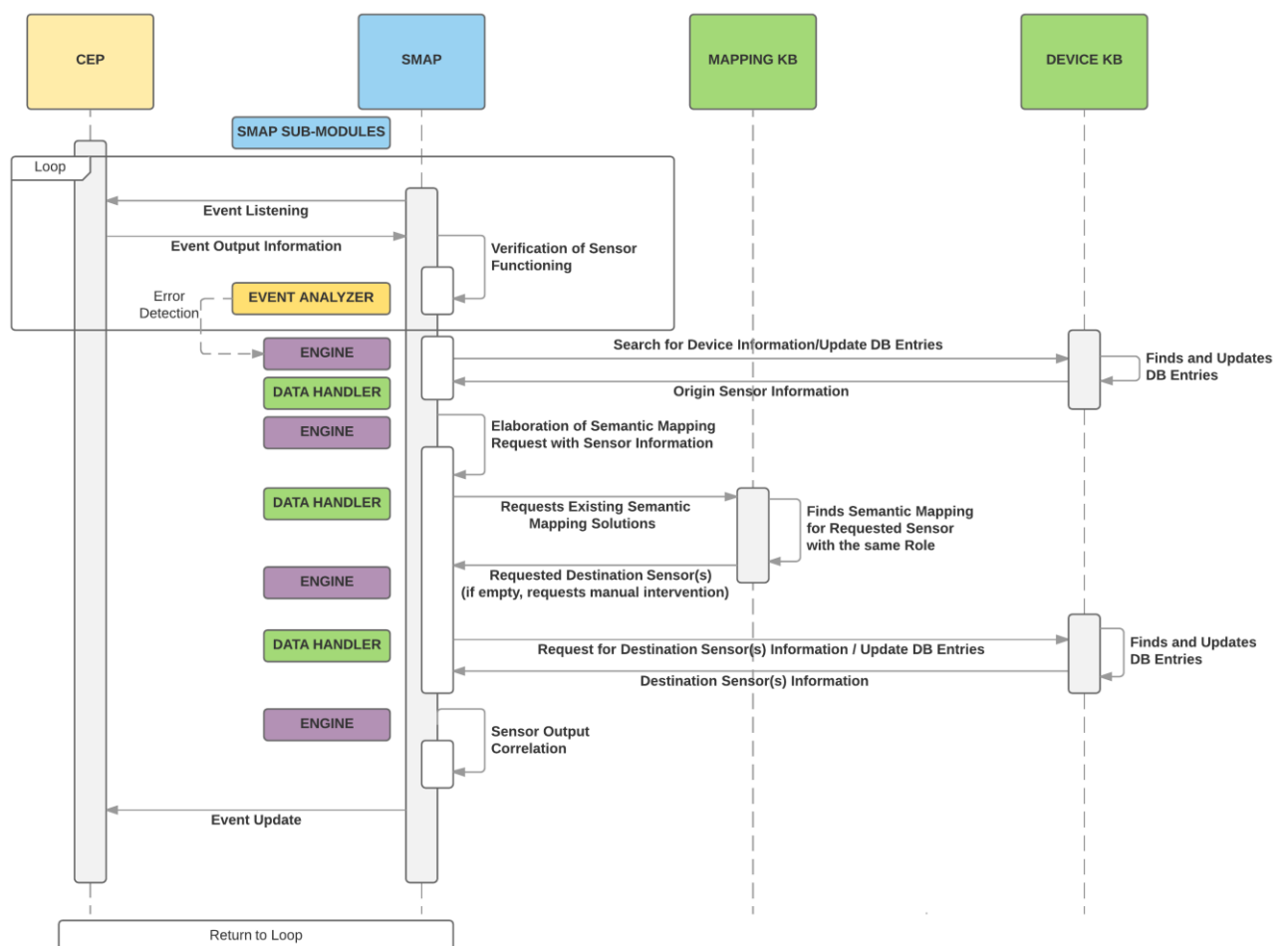


Figure 3-9 - Sequence Diagram of the Architecture's Process.

A key aspect to accomplish the previous features is to have a design process, parallel and able to run before the runtime process. This design process will allow the creation of semantic maps by the user. It is important to mention that autonomous processes may be easily created to design the semantic maps with the available information of the network and with a few core semantic maps that lay a base for the creation of others. But to simplify this stage, in Figure 3-10, the creation of semantic maps is initiated by the user and relies on the input interface used by him. The following diagram, Figure 3-11, shows the sequence diagram that highlights the processes of the input interface and each part of the module (sub-modules) and knowledge bases.

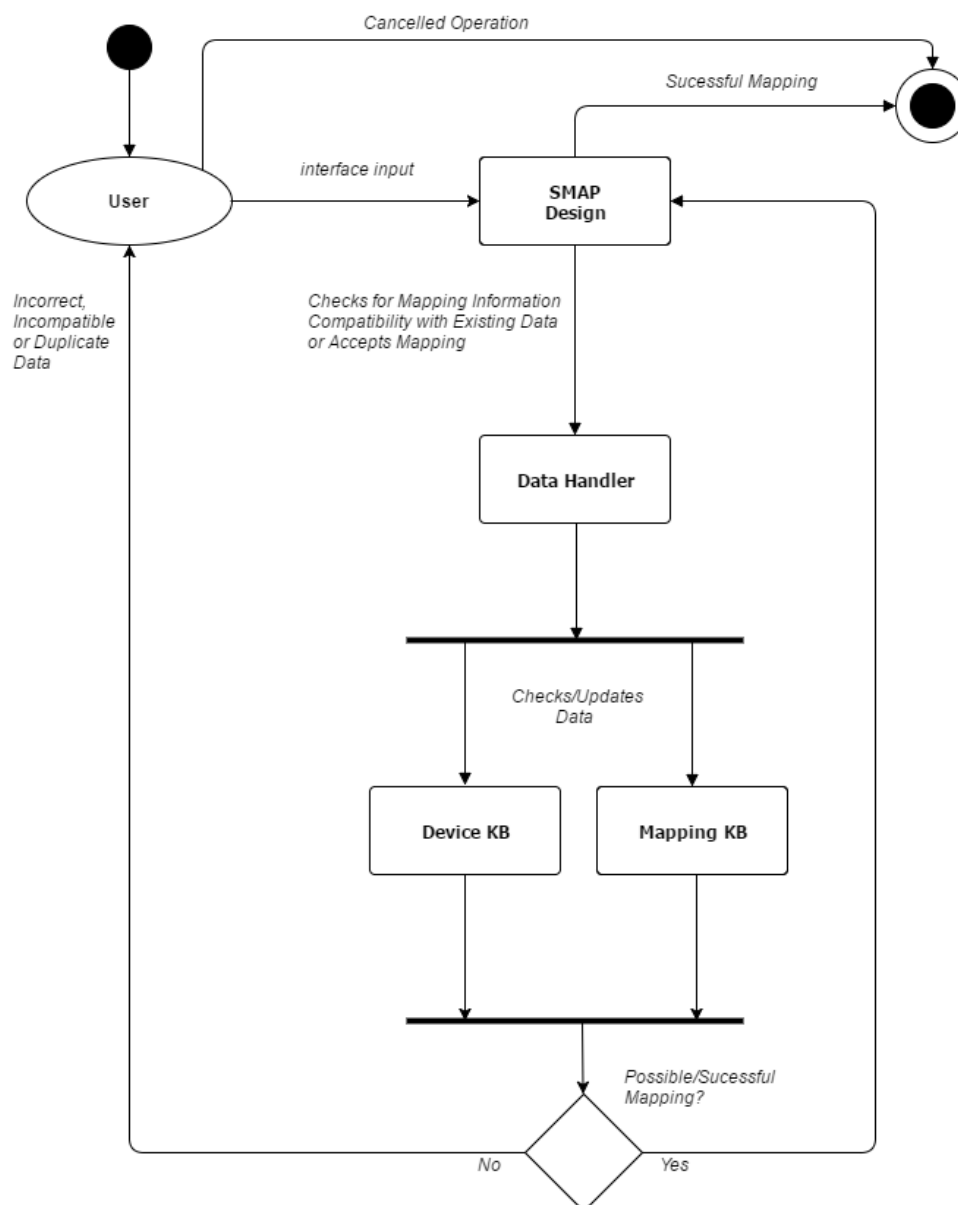


Figure 3-10 - Activity Diagram of the Design Process.

The design process (Figure 3-10 and Figure 3-11) is initiated by the user. An input interface is provided to allow him to provide the required information for the creation of a

semantic map. The SMAP Design sub-module is responsible for the interface and the coherence of the data. After ensuring the correct data format, the SMAP Design transfers the information to the Data Handler sub-module. The Data Handler is then responsible for checking if it is possible to create the semantic map and, if so, updates the information present in the device and mapping knowledge bases and informs the SMAP Design module to terminate successfully the operation. If not, the user is warned. At any time, the user can cancel the operation or initiate a new one, one per interface.

Figure 3-11 - Sequence Diagram of the Design Process.

3.3 System Architecture

The objective for this sub-section, is to provide more information about the architecture of this project. To accomplish that, each individual module is going to be more explained, still from a semi-practical approach, regarding its importance and the focus that it demands. The last sub-section presents the SMAP general objectives and sub-modules that allow those objectives to be achieved.

3.3.1 CEP

The complex event process (CEP) module, was briefly explained in the sub-chapter 3.1 and to avoid repeating the basic and generic notions of this concept, the approach in this sub-section will be simple and from a functional point of view. Its implementation is not the focus of this work, but concepts and methodologies of this module will be used to manipulate, manage and process events regarding some key testing aspects of the implementation.

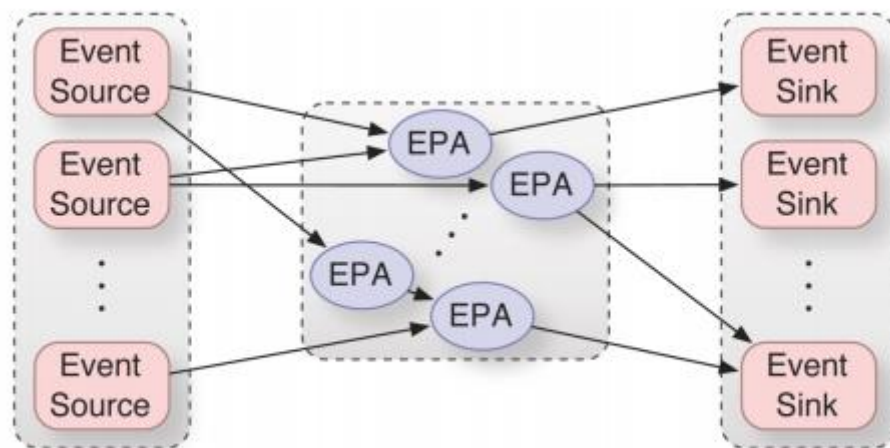


Figure 3-12 - CEP Overview (Seeger, 2012).

The standard CEP implementation is usually defined in three steps: the registration of event sources, the definition of EPA's (Event Processing Agents) and the registration of event sinks. The event sources, usually devices in the IoT paradigm, are responsible for providing data from the monitored environment, illustrated in Figure 3-12, and will typically generate a large number of events. The EPA's, the core of the event processing and defined as CEP Engine in Figure 3-7, process the input from the event sources and try to detect situations of interest (SOI). These SOI are set by pre-defined rules, integrated in the CEP DB considering the architecture presented in Figure 3-7, regarding the context of the implementation. It is important to mention that a CEP mainly focus on detecting SOI in streaming data rather than manipulating data streams (Woods, Teubner, & Alonso, 2010) and static CEP, contemporary predominant, are very context-sensitive (Hoßbach & Seeger, 2013). Regarding this information, ordinary context-based events will be discarded and the events that express importance, typically considerably less, will be submitted to analysis regarding persistent queries directed to the, already mentioned, rules. The results of such analysis, generally originate actions that take place in the event sinks (Figure 3-12), to whom the rules are specifically oriented to.

An event can be defined as a significant change of state (Chakraborty & Eberspacher, 2012) and the use of a CEP and focus on device originated events, makes the suggested architecture in this work an Event-Driven Architecture (EDA). Nevertheless, the CEP specific design does not concern to the subject of this thesis and is only considered for implementing and testing purposes. It is also important to mention that some event processing, regarding

the verification of correct sensor operation, is done by the semantic mapping module, in the Event Analyzer sub-module, allocating some working load from the CEP.

3.3.2 Devices

The devices module, represented as the low-level architecture in Figure 3-7, represents the technology that provides and generates events to be considered by the CEP. These events are usually raw information about the monitored environment and make no judgement or evaluation about it. The W3C SSN Ontology, already described in Chapter 2 (section 2.2), has a good definition to look upon when implementing something above this layer. This module is also connected to the semantic mapping module because of the need to keep an up-to-date device database that may also be updated, autonomously or manually, according to specific changes that may occur in the devices.

3.3.3 Device Knowledge Base (C2NET Ontology)

The importance of the device knowledge base is mentioned in sub-chapter 3.2 and the main objective focuses on an updated registry of the sensor's information, specifically their current roles and state of functioning. In Figure 3-13, a class diagram regarding the features of a device that are crucial to this architecture is presented.

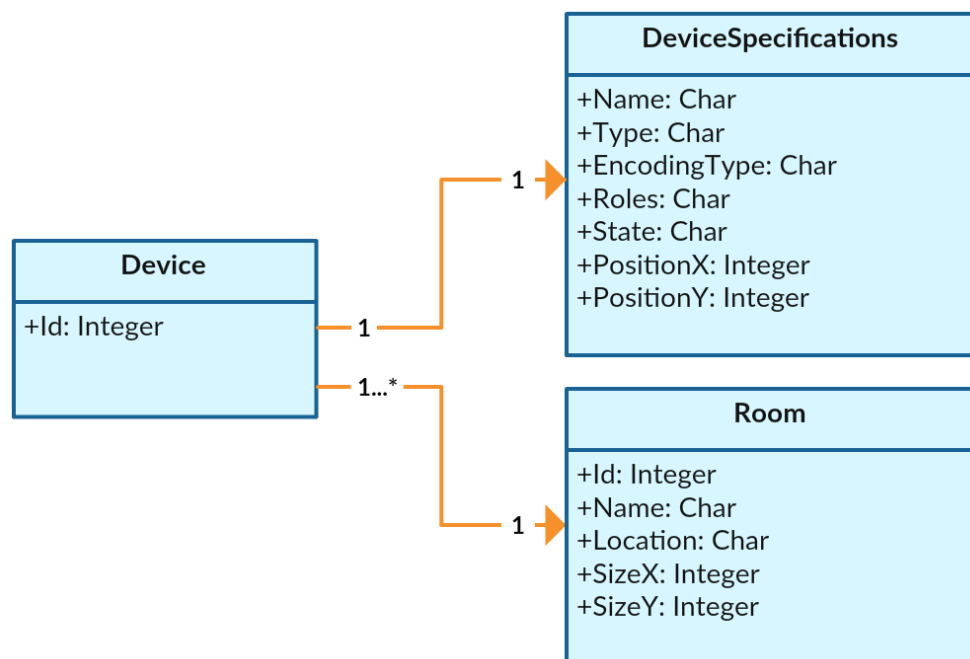


Figure 3-13 – Device Knowledge Base Class Diagram.

The *Id* is the unique identifier of the device. The device is then related 1 to 1 to its specifications, because each device has its own specifications. In the specifications, the *Name* is the manufacturer designation, the *Type* specifies the measurement objective, the *EncodingType* defines the measurement scale, the *Roles* mention the actual ongoing

applications of the sensor, which may be none, and the *State* contains a variable that expresses the current operating state.

PositionX and *PositionY* are two parameters that express the position (as coordinates) of the sensor in the room, relative to a reference point (typically the centre or a corner of the room). These parameters allow methods of fault detection in the SMAP, per example, output comparison to other nearby devices (which are explored in the implementation contemplated in chapter 4).

For each room, there can be one or multiple sensors. The room sets the perception of place and is necessary to understand where certain semantic maps are applicable or not. Per example, if a temperature sensor fails and the destination sensor is in another room, the solution provided by such semantic map should not be considered. The room is characterized in this ontology by *Id*, its unique identifier, *Name* of the room, geographical or relative *Location* and size (*SizeX* and *SizeY*). The latter attribute, is useful to understand whether a device is within the boundaries of the room and needed to successfully implement and simulate, allowing a visual component to the user by a possible, but not obligatory, graphical user interface. This structure of information was based on a pre-existing ontology and extended the work already developed for the C2NET European Research project, on which this dissertation is included. More details about this project are mentioned further in this dissertation.

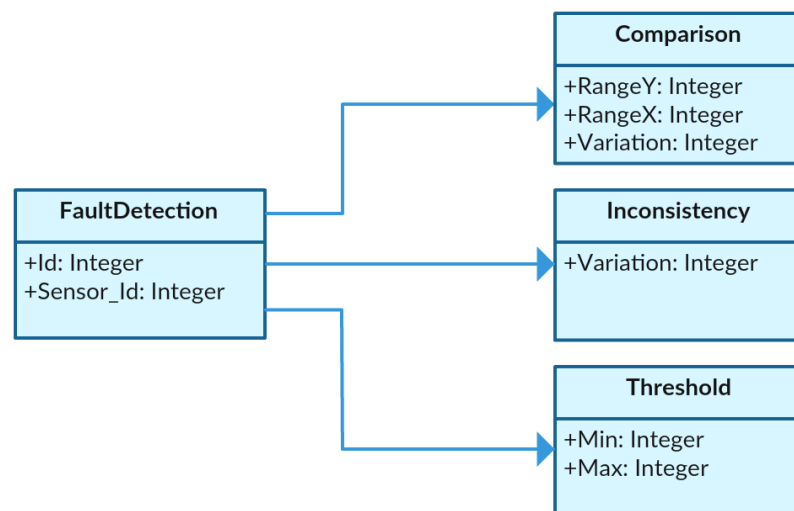


Figure 3-14 – Fault Detection Class Diagram (with 3 Example Techniques).

Other interesting concept, not mandatory for the operation of the SMAP module, that assures correct failure detection is the Fault Detection methodology, to be used by the Event Analyzer sub-module. The detection of failures and malfunctions exceeds the objective of this dissertation, so we will consider the following methodology presented in Figure 3-14. These three cases (to be used in the implementation of this project): Comparison, Inconsistency and Threshold; are a direct approach, based on some of the most common techniques, to determine the functioning of a sensor (Sharma, Golubchik, & Govindan, 2010) (Akbari, Dana, Khademzadeh, & Beikmahdavi, 2011). In this case, according to the class diagram provided

here, a certain Fault Detection instance (there can be many) is associated with a certain sensor (*Id* and *Sensor_Id*). Then, the output values of that sensor can be tested by the technique of Comparison, where there is a *rangeX* and *rangeY* to delimit the size of a rectangle around the sensor, where other nearby sensors are considered for comparing using the maximum value variation that is allowed. Other techniques are Inconsistency, that compares the current reading with previous ones from the same sensor to understand if the current reading makes sense, and Threshold, that sets a minimum and maximum value for the readings of that sensor. In this case, each sensor can have its own Fault Detection values and configurations.

3.3.4 Mapping Knowledge Base (SMAP Ontology)

The mapping knowledge base was also mentioned before, in terms of definition and objectives in the architecture. Figure 3-15 shows the class diagram for this special case of database.

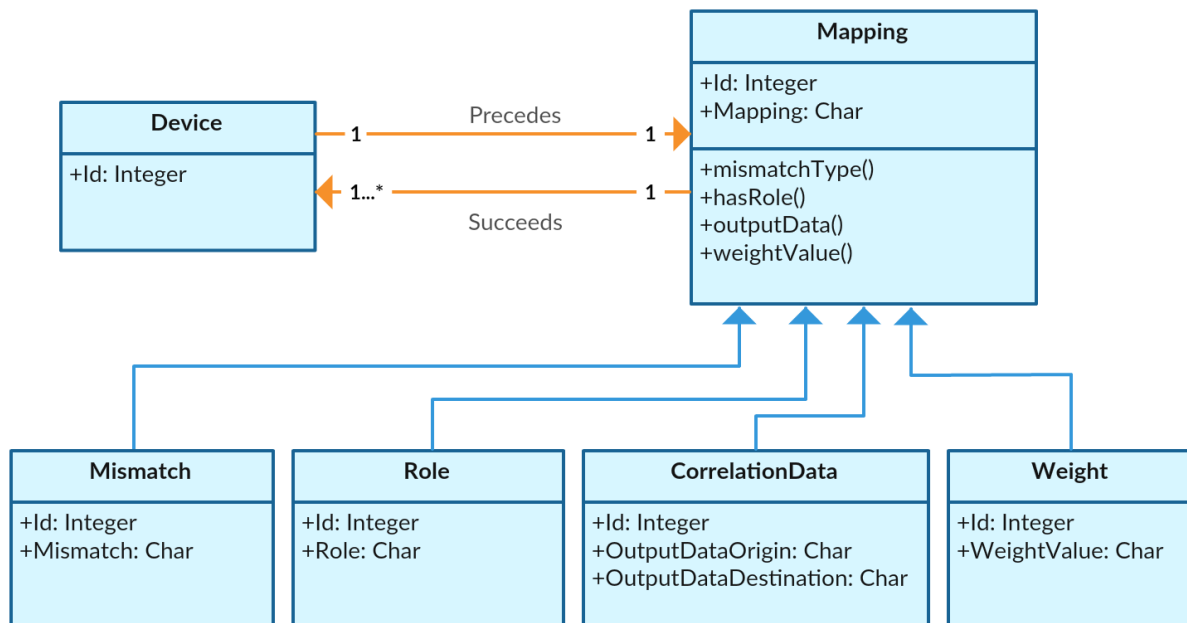


Figure 3-15 - Mapping Knowledge Base Class Diagram.

Above, sensors are described as in the class diagram represented in Figure 3-13. The connection between sensors and mapping is not as direct as may seem and is provided by the semantic mapping module. Nevertheless, it is explicit that for each mapping there is only one origin sensor and may be one or more destination sensors. The mapping properties were thoroughly detailed earlier in this chapter, sub-section 3.1, prior to presenting the semantic mapping equation (Equation 1) and is not necessary to detail them again in such way. It is important, however, to refer two interesting and unmentioned details. The output data has here two different fields (*OutputDataOrigin* and *OutputDataDestination*) to differ the output information of the devices of a certain mapping, in which the one referring to the destination may provide data to multiple sensors if that is the case. These fields can contain the same

expression (using a variable for each involved sensor) when the correlation is simple, but the existence of two separate fields beneficiates more complex cases, with multiple destination sensors. Finally, the suggested operations, more may be added according to each specific implementation, under the attributes of the class *Mapping* are just for verification and manipulation of the mapping properties.

3.3.5 Semantic Mapping Module (SMAP)

The semantic mapping module (SMAP) stands as the module that represents the concept and methodology of this dissertation. The *main functions* that this module should have, in order to comply with the architecture provided, illustrated in Figure 3-7, are:

- **Specific Event Listening from CEP** - The events are analysed according to situations of interest regarding the monitored environment, the potential of the developed semantic maps and the pre-determined mapping situations;
- **Verification of Sensors** - The SMAP module, like the CEP, will process events that interest to the objective of its existence. The main interest is to analyse the state of the sensor, to check if it is functioning properly. Along with the last point, these two functions represent a loop of event listening and processing, until some situation triggers a semantic mapping procedure. Some pre-defined failure situations can be measurements that exceed typical values, constant measurements, noisy readings or non-concordant measurements between two or more sensors of the same type, in the same area (Munir & Stankovic, 2015)(Sharma et al., 2010);
- **Search and Update Queries to Device Database about Origin Sensor** - In order to keep an updated record of the sensors and to search the correct mappings, the semantic mapping module searches the available information about the sensor, including the roles that is operating at the moment, and updates them to “none” and changes the state of operation of the device to “Error”;
- **Search and Update Queries to Mapping Knowledge Base** - After acquiring the information about the sensor, the SMAP module elaborates the request (search and update queries to the mapping knowledge base) for a semantic mapping solution. To accomplish this, it searches the role, or roles, of the origin sensor in the mapping knowledge base. If there are no semantic maps for that type of role, manual intervention is solicited. After finding the semantic maps for the needed role, the maps that have the mentioned sensor as the origin sensor are selected, by adding constraints to the previously mentioned query. The remaining maps, if more than one, have a weight component associated to them, as mentioned in 3.1, and the map that has a higher weight value is selected for the semantic mapping process;
- **Search and Update Queries to Device Database about Destination Sensor(s)** - Similarly to happens when using the device database for the origin sensor, the semantic

mapping module queries the device database for information about the destination sensor. It retrieves the current state to verify if it is indeed available (normally it is operating because the device database should have updated information). The roles may also be considered to avoid too much reliability on a solution, but again this is controlled by the weight value in sensor database and it is typically updated regarding that issue. If everything is according to the specifications for good functioning, the sensor is updated within the database with a new role and state;

- **Sensor Output Correlation** - In this phase, within the semantic mapping module, the correlation between outputs from the origin and destination sensor is considered recurring to the output data and mismatch from the semantic map information, retrieved from the mapping knowledge base. Any particular change or specification that the CEP has to deal with, in the event processing or event rules that use the destination sensor considered, is transmitted to the CEP engine;
- **Event Update in CEP** - In this final phase, the information about the destination sensor, or sensors, is provided to the CEP in order to make the necessary changes to the events, previously using the origin sensor and replacing it. The information considered in the last point, sensor output correlation, is also provided to the CEP in order to integrate them in the mentioned events.

These points, mentioned above, are the core objectives for the functioning of the SMAP module. This module has distinct operations and can easily benefit from being split in different sub-modules, not only to clarify its functioning to the observer but also to “*divide and conquer*” (as said by *Gaius Julius Caesar*), allowing for the design process of each sub-module to be direct and objective, only regarding input/output and relationships with other sub-modules/modules. The ***sub-modules***, explained briefly in section 2 of this chapter with the sequence and activity diagrams, and also illustrated in Figure 3-7 were:

- **SMAP Design** – This sub-module is where the creation of semantic maps occurs. It can be an automated process, but we will focus on using a user input interface. This interface allows to enter the required information for the creation of the semantic maps and ensures coherence and the correct information formats. The SMAP Design interacts with the Data Handler, providing it with the information for entering the semantic map into the mapping and device knowledge base. This process may fail if the Data Handler checks any type of anomaly, such as wrong or duplicate information, and the user is informed. The SMAP Design also interacts with the SMAP Engine, which is responsible for the runtime process, allowing the creation of semantic maps during the execution of the system;
- **SMAP Engine** – In order to keep all processes moderated and organized, a SMAP Engine is included in the architecture of the SMAP module. It is responsible for initializing the use of semantic maps (from an event selected by the Event Analyzer), activating the Data Handler, sending and receiving information about the instances

(sensors and maps) during the interaction with the Data Handler and conducting the process that leads to the use of the maps, sending the information of the updated network state to the CEP Engine, moderating and cope with the use of the SMAP Design by the user during the system's runtime and sending and receiving general acknowledge and confirmation information between the sub-modules;

- **Event Analyzer** – This sub-module acts only in the loop process of the event listening by the SMAP module. It listens to events generated by the devices and compares them to presumable values, to verify any situation of interest. If some situation of interest arises and the Event Analyzer concludes that a device is not working properly (using fault detection methodologies), that information is passed on to the SMAP Engine and the Event Analyzer continues to listen to the device generated events;
- **Data Handler** – In this sub-module, all the queries and processes related to the Mapping and Device knowledge bases are handled. The objective is to keep up-to-date records of the network and available semantic maps. This module is triggered by the SMAP Engine to work in specific processes (e.g. query for the origin sensor or updating a semantic map entry) and by the SMAP Design when it is the creation of a new semantic map with information generated by user input.

3.4 Potential Application Scenario

The application of the designed module, presented in this chapter, was envisioned while developing it. This module assumes the functioning described earlier but it does not demand a strict implementation and is adaptable to the different situations in the environment of the IoT technologies. Its features were designed in order to help the network, where it is implemented, to have scalability and handle the constant growth and diversity of the IoT paradigm, without being too susceptible to failures.

This dissertation objective is to demonstrate the applicability of implementing the semantic mapping module to a scenario of a real IoT environment and measure its effectiveness by deducing the level of trustworthiness, optimization and the redundancy that it provides, to justify the use of this methodology. Thus, an idea of a possible application scenario is detailed in order to set the tone for the upcoming implementation description.

Smart Factory Room

The scenario for the implementation described further in this dissertation is a smart room of a modern factory that has various machines and sensing devices that allow to control its functioning and to provide better working conditions for the employees. This type of “*aware*” room is part of what is now called the Industry 4.0 (or fourth industrial revolution) and it is integrated in the IoT paradigm, aiming to connect embedded system production technologies and smart production processes with increased connectivity and ever more

sophisticated data-gathering and analytic capabilities (Sniderman, Monika, & Cotteleer, 2016).

Table 3-2 - Examples for Possible Objectives of Sensing Devices in the Application Scenario.

Objective	Possible Devices	Example of Rule and Action
Adequate Room Temperature	Temperature Sensor	When room temperature is high, the air conditioning is activated.
Fire Detection	Temperature Sensor; Ionization and Photoelectric Smoke Detectors; Gas Sensors	When temperature and/or smoke density reach levels that indicate a fire, fire alarm is activated
Room Light Adjustment	Photoresistor, Photovoltaic or Photo Diode Light Sensors	Light in the room is adjusted according to a pre-defined level of lightning.
Security	Ultrasonic Sensor; Passive Infrared Sensor	Safety Alarm is activated when an intruder or non-authorized person is detected.
Gas Leakage or Oxygen Depletion	Various Specific Gas Sensors (like CO ₂ , CO, O)	When the concentration of certain gases in the room is life-threatening, the ventilation is activated.
Detect Safety Distance to Machinery	Ultrasonic Sensor; Passive Infrared Sensor	When safety distance to the factory's machinery is not satisfied, safety alarm is activated and the specific machine is stopped.
Tracking Products, Tools or Persons	RFID; Nano Tags; Image Capture and Recognition	Determining current states of the production process and human safety.

The justification for the choice of this context is entirely connected to the ever-growing technology integration, the needed scalability of the networks (regarding the number of sensing devices) and the demanded efficiency existing in this type of manufacturing environments of the contemporary industries. Therefore, the theme of this thesis fits in this paradigm and is prone to be more useful in situations like this, that demand handling various sensing devices with the objective of providing a more efficient and trustworthy manufacturing process.

In Table 3-2, it is provided some insight of the objectives, that the sensing devices, along with the technological modules (like the one developed in this dissertation), may improve and offer to this type of environment. The SMAP module, specifically, is intended to allow reliability to these processes, managed according to the rules specified by the developers in the network, with redundancy by using different devices, already deployed, to assure the continuous manufacturing process and wellbeing of the employees.

There are multiple and possibly infinite applications of this processes and methodologies. The objectives column, regarding the implementation of the semantic mapping module, is directly related to the role field in the mapping equation (Equation 1). Thus, the role is a designation of this objective and the right column of Table 3-2 sheds some light of a possible association of that role to a pre-determined event rule. The centre column shows possible devices, used singularly or together, to allow measuring and determine the values that will be compared to the thresholds delimited by the mentioned rules.

Finally, to clarify the environment of the implementation of this dissertation, an example of a map for a smart factory room, based on the examples provided in Table 3-2, is provided in Figure 3-16. The objective is to have a similar view, during the simulation of the program developed for the implementation, of the one provided in this map. The map shows a certain degree of redundancy of the network, by being populated with various sensors. The composition and placement are intended for demonstrative purposes and may not entirely specify the optimal design for a real scenario. Also, the map is not presented with any scale or measurement accuracy and the symbols do not follow the architectural specifications for a room, since those are not the focus of this dissertation. Other important detail to mention is that the devices presented in Figure 3-16 may belong to any object (for example, a machine), if any, but belong to the network and can be accessed to retrieve measurement information. In a similar scenario, during the implementation, some events that trigger the detection of failures in the devices and its measures are created in order to test the application of semantic maps and the redundancy it sets to achieve. The attempt to use the semantic maps does not only test the semantic mapping processes but can also determine the degree of trustworthiness of the system, by evaluating the capacity to overcome difficulties and failures. The next chapters present the specifications of the module and the concerning testing.

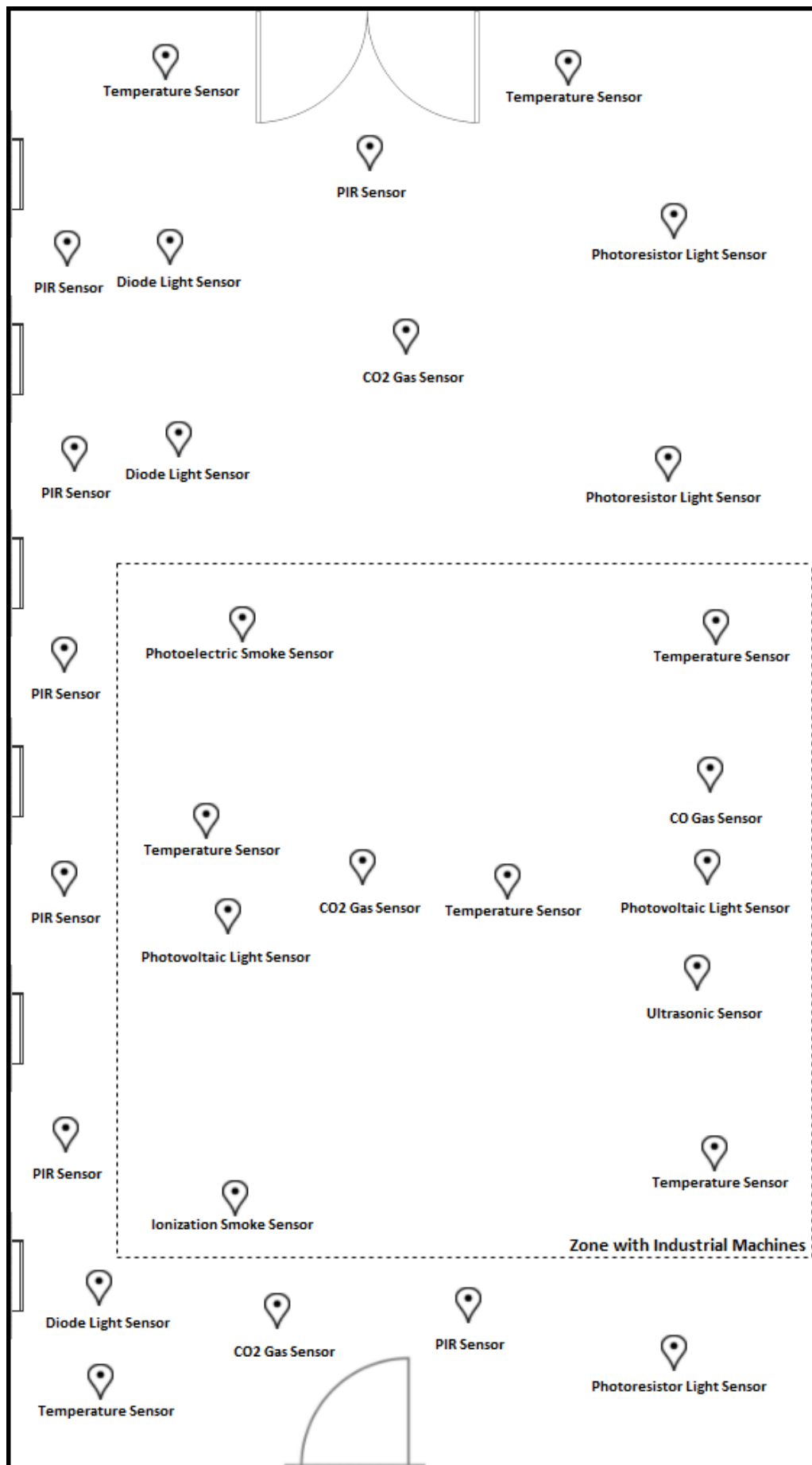


Figure 3-16 - Smart Factory Room Map Example.

4 Proof of Concept Implementation

This chapter provides information about the implementation of the architecture designed and explained previously. The sections here presented are divided in three major aspects regarding the implementation process. In the first aspect, requirements and functionalities, the needed elements and objectives for the module to be implemented and function as intended. After that, the technologic specifications are detailed, regarding the software and components that take part in this development. Finally, the last sub-section of this chapter, provides the considered implementation steps during the creation of the elements of the architecture and the virtual simulator to test everything together.

4.1 Requirements and Functionalities

In this sub-section of chapter 4, the requirements and functionalities for the SMAP module are presented. These requirements follow the context and objectives of this dissertation, mentioned in Chapter 1, and are the general questions to which the architecture, presented in the previous chapter (Chapter 3), aims to respond. Also, the functionalities that have been implemented, in part to comply to the requirements, during the creation of the architecture design are mentioned and explained. Some assumptions occur during this process, based on researched information, to achieve the initial proposed goal.

Requirements

- **The semantic mapping module should be compatible with the generic implementation of a CEP** without intervening with its functioning. The semantic mapping module should not be dependant of any of the CEP's context variable characteristics. The semantic mapping module must also be designed to be able to be applied in an already implemented architecture of a network.
- **Be able to maintain up-to-date information** in the structures of that nature within the architecture. The device knowledge base tackles the importance of having the devices information and the mapping knowledge base addresses the mapping information.
- **The semantic mapping module should detect situations of interest** upon which triggers the process of the use of semantic maps in order to apply redundancy to possible failures in devices.

- **The semantic maps should provide pondered redundancy to the system** by recurring to certain aspects of the architecture like the databases, weight-based decision making and general analysis of the current network, to provide more reliability to the system.
- **The sensors and their data should be considered as an asset to the network general operation** instead of being only used to one function or rule verification, and their data dismissed for all other purposes of the monitoring system.
- **The semantic maps should not only provide redundancy but also be helpful to establish the trustworthiness of the system** by providing a level of confidence according to the number of semantic maps that may be established in a certain network. Lower possibilities for semantic maps may be a signal of a low number of devices in the network to apply redundancy. Certain areas, with few sensors or that have a low number of semantic maps, can be considered critical in terms of possible failures.
- **Detect and react to potential failures**, not only in the device events that may originate the use of semantic maps but also potential failures in the processes of the architecture and semantic mapping module. The implementation and testing processes are important to this aspect because they aim to improve the robustness of the developed work.
- **Be able to adapt to different contexts** with the minimum number of modifications possible, to guarantee the correct operation of the methodologies of the architecture. The IoT is intended to submerge the everyday objects with technology and, because of that, many different deployments for the IoT concepts may arise. This implementation leaves some room for adaptation to specific scenarios, but the amount of variation or lack of direct nomenclature may interfere with the performance of the methodology.
- **The architecture should be interoperable** to provide easy communication between each module and with the rest of the network. All devices, disregarding its specifications, should be able to be integrated within this methodology.
- **The number of devices, events and amount of data should be scalable** to deal with the demand of real scenarios in the IoT paradigm, like *big data*. With this, the architecture should be able to grow, be receptive to upper-layer mechanisms, processes and techniques, and manage to keep an advisable level of security and data integrity.

Functionalities

- **Any device can be mapped** because the specifications of this implementation are inclusive and are generically targeted for different applications. The mapping possibilities depend only on factors from the network itself.
- **The semantic mapping module is easily implemented or integrated** in the great majority of the IoT network because it follows the specifications of the major IoT models and the concepts that constitute the IoT paradigm. This also happens because the

semantic mapping module is designed to be easily integrated to operate with an event processor, without causing any type of interference to its functioning.

- **The semantic mapping module eases some of the CEP's workload** due to being able to process the events, that express situations of interest from the monitored environment, regarding the possible use of the semantic mapping process. This way, the CEP can be configured to handle, more objectively, the events and queries that are compared to the specified rules, without being so greatly compromised by possible failures of the devices.
- **The architecture maintains an updated database of information about the sensing devices and mapping possibilities.** The process of querying and updating accordingly to the current events is responsibility of the SMAP module.
- **The sensors operational state is checked according to the events** that the CEP initially receives, or lack of them, by the semantic mapping module.
- **The existence of semantic maps and semantic mapping processes confer a degree of trustworthiness to the network that can be comparatively measured,** simply by analyzing the existing mapping conditions, the sensors involved in that mappings and the number of semantic maps.
- **The network is more capable of adapting to the failure of devices and the environment context** due the use of semantic maps. The environment context can be prone to failures due to typical unpredictable or misleading situations and the semantic maps can be a way to solve them by providing similar different configurations of devices within the network.
- **The SMAP module provides direct information to the CEP about what changes need to be made to the events** that were signaled as faulty. This information often contains details about adjustments, to the rules in the CEP database, that are needed when replacing each sensor that is used for a certain specific rule, along with the information of the destination sensor(s).

4.2 Technological Specifications

This sub-section presents the technological tools that were used during the development of this dissertation. The implementation consisted in a virtual application scenario, projected using software that was configured using developing tools and code writing, to create situations similar to real case scenarios and to test how the concept of this dissertation could work. The IDE for the code development of this project was Eclipse (to take advantage of the technologies mentioned next and the all the already available libraries) and the development programming language was Java to comply with the requirements of this

project and to take advantage of the technologies mentioned next in this section, which seem appropriate to the application desired.

Jena

The Apache Jena is an open source Java framework for building Semantic Web and Linked Data applications that supports OWL (Web Ontology Language). It provides an API to extract data from and write to RDF/XML graphs, which are represented as “models”. These models can be queried through the SPARQL standard (a semantic query language for databases in the RDF format) (Segaran, Evans, & Taylor, 2009) The SPARQL queries allow triple patterns, conjunctions, disjunctions and optional patterns (Prud’hommeaux & Seaborne, 2008) to filter the desired data.

The main reason to choose this type of framework for this work was the capability to work with Java, SPARQL queries and OWL files, generated by another software that was used for creating and schematizing ontologies (Protégé).

Protégé

Protégé is an open source modelling tool developed at Stanford Medical Informatics and has a community of thousands of users (Knublauch, Ferguson, Noy, & Musen, 2004). This tool development was largely driven by biomedical applications (Gennari et al., 2003), the system is domain-independent and has been used for many other application areas.

The modelling part and viewing part has a distinctive separation in Protégé. The model is the internal representation mechanism for ontologies and knowledge bases. The view components provide a user interface to display and manipulate the respective model (Ferreira, 2012). The model can also be viewed as something based on a simple yet flexible metamodel (Noy, Ferguson, Musen, & Informatics, 2000), which is comparable to object-oriented and frame-based systems. It can represent ontologies consisting of entities such as classes, data properties, object properties, individuals/instances, datatypes and property assertions. The tool also provides a Java API to query and manipulate models, useful for accessing information saved in the property assertions of the individuals, functioning in a similar way to a database management tool.

This tool was chosen due to the necessity to build and use ontologies based on the studied IoT architectures, the capabilities of displaying those ontologies in a clean and intuitive way and the fact that provides a connection to Java and exports an OWL file on which is possible to manipulate individuals and properties easily during the operation of the system.

Esper

Esper, from EsperTech, is a component for complex event processing (CEP) and event series analysis, available for Java applications. This tool allows the processing of large volumes

of incoming messages or events. It filters and analyses events, responding to pre-set conditions of interest (Oberoi, 2011).

Esper, in association with the Event Processing Language (EPL), provides a scalable, real-time and memory-efficient solution. EPL (technology also known as SQL streaming analytics) is a declarative language for dealing with high frequency time-based event data (EsperTech, 2016). Some typical applications are in finances, sensor networks and management processes. Esper does not require any architecture, container or any dependency.

The reason to use this tool was to have an independent component similar to a CEP, easy to implement and work with. The main advantage of Esper is the conditions, that help to automatically trigger processes related to the failure of sensors. In this case, the use of semantic maps.

4.3 Implementation Steps

Finally, this sub-section, aims to introduce the next chapter by providing some details regarding the implementation of the architecture of this dissertation. In order to organize and establish the order of work to attend during the implementation, the practical approach was divided into steps.

4.3.1 Step 1 – Designing the Architecture

In this initial step, the functional and sequence models were designed. These models were already presented in section 3.2 and aim to determine the flow of the system processing and be the basis and canvas for the next implementation steps. In this step, the structures for the device and mapping knowledge bases, also defined in Section 3.2, are considered. The architecture (Figure 3.7) of the system is very important to consider in detail, in order to fulfil the desired relations between the modules and sub-models.

4.3.2 Step 2 – Creation of Models and Instances on Protégé

In this step, the C2NET ontology was considered for developing the model in Protégé to contain the information of devices (as a knowledge base). To this ontology, some adaptations were made for this implementation in order to be capable of holding some additional features such as the perception of localization and space for the devices.

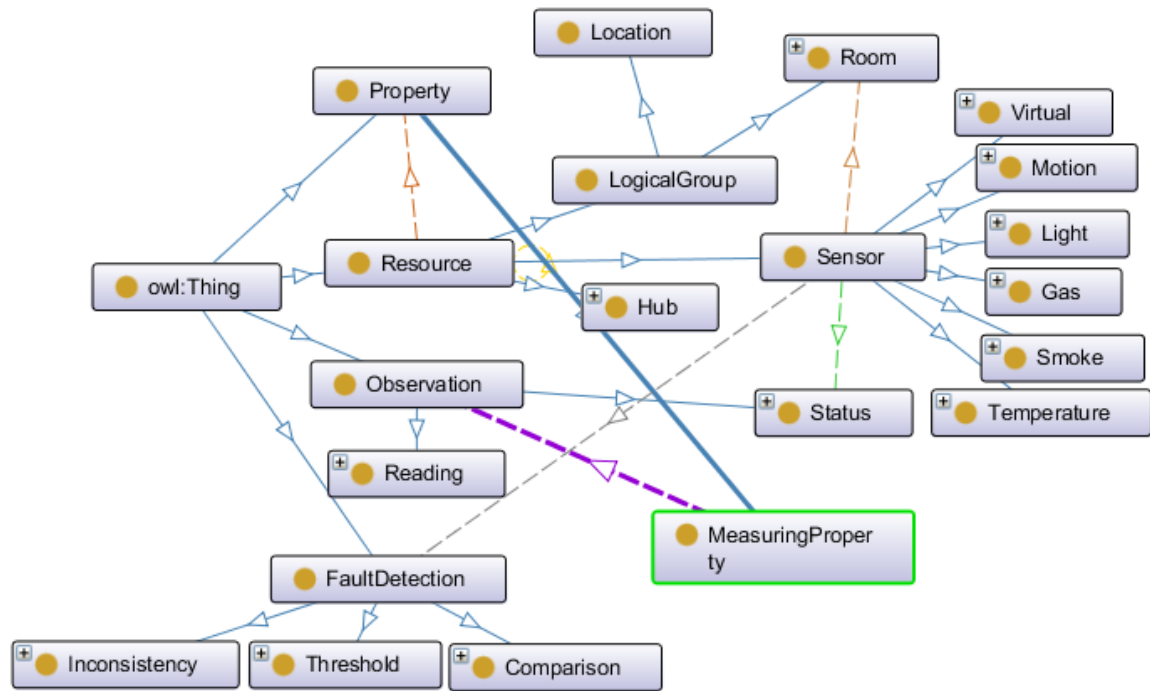


Figure 4-1 - C2NET Ontology OntoGraf.

Figure 4-1 shows a representation of the model created based on the C2NET Ontology. This representation is made by a tool integrated in the Protégé software named OntoGraf which allows the interactive visualization with filtering of classes, subclasses and individuals, and their respective relationships.

In this model, we can see the three main classes of C2NET: Property, Resource and Observation. To this application, the class of Resource is more thoroughly explored because it is where the devices (or sensors) are defined, which are the elements that this work aims to use to implement the semantic maps. Also, in the Resource class, inside the LogicalGroup subclass, the concept of Room is defined. This Room is the physical location where the sensors are deployed, giving a location context and preventing the existence of the same sensor in two different places at the same type (in typical real-world applications, using a sensor from a different room has no significant value for any task due to the distance and physical barriers between the sensor and the point to be measured). In the Observation class, there is a subclass of interest named Status, that defines the possible states for the sensor. In this implementation, it was considered four possible states: On, Off, Standby (functioning but waiting) and Error (damaged or malfunctioning). The Fault Detection class is an additional feature to be explained further in this chapter.

In Figure 4-2, a more detailed (or extended) version of this OntoGraf is presented. This version includes various individuals (or instances) that were used later for testing. These individuals allow to test and experiment with the ontology model to understand its coherence and good construction. Later, in this dissertation, these individuals are manipulated and displayed to the user in the graphical interface.

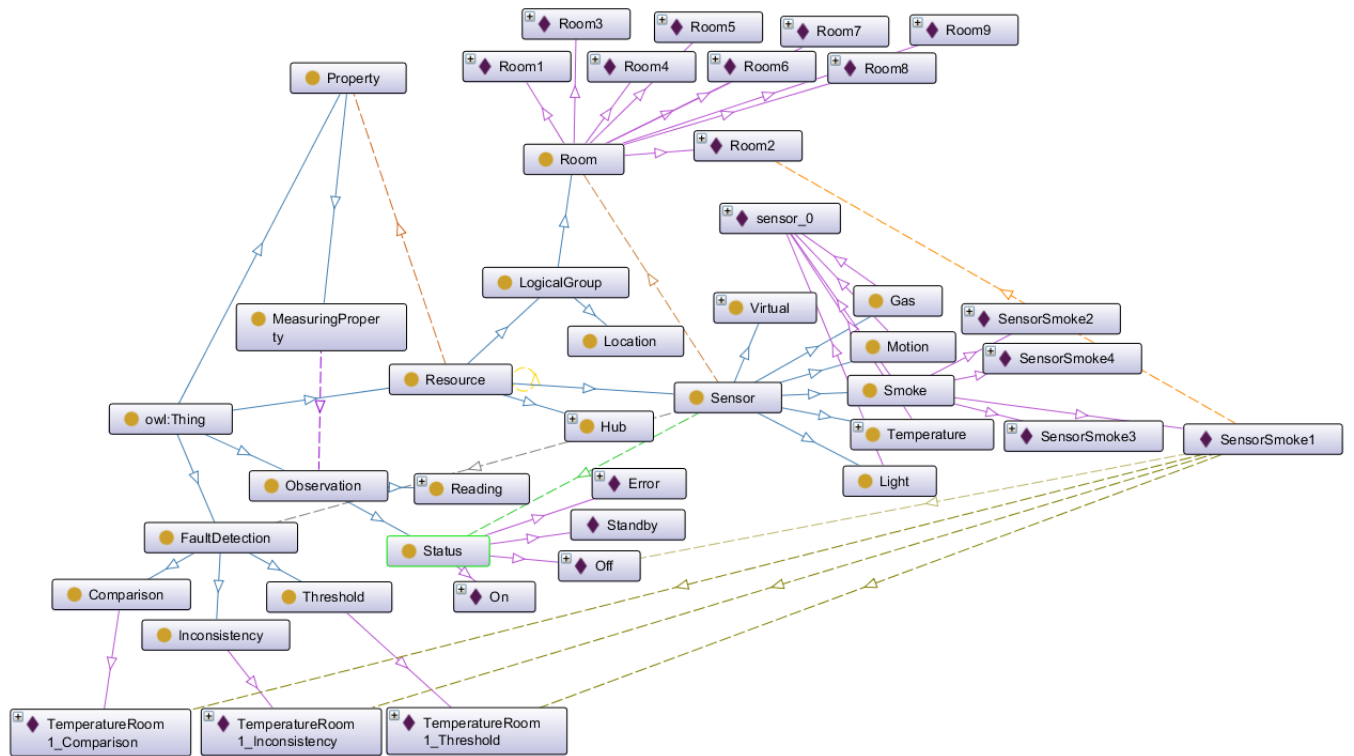


Figure 4-2 - C2NET Ontology OntoGraf Extended.

This OntoGraf may seem confusing at a first glance, due the amount of connections needed for the upcoming testing, but by noticing carefully, it is possible to verify all the relations between individuals and classes, that were planned in the previous chapters of design. It is important to remind that this representation, with all the individuals, is just a view of the content of the OWL file, to provide an explanation and demonstration of this aspect, which is not intended to be read by the user. Nevertheless, we can follow the *SensorSmoke1* connections, for example, to understand that it is a *Smoke Sensor* (Resource), from *Room2* (Room) and its current state of functioning is *Off* (State). This sensor also has, associated with it, fault detection instances related to temperature, using three methods (Inconsistency, Comparison and Threshold).

Each individual of this model will have its own property assertions (object and data), in which these relations can be directly viewed and accessed. In Figure 4-3, an example is provided. *SensorTemperature1* is currently in *Room1*, its state is *Error*, it has fault detection instances associated and has the 2D coordinates $X=2$ and $Y=1$. The unit of measure of the coordinates is arbitrary and later, in this implementation, is considered as cells in a grid representing the room (for example, $X=2$ and $Y=2$ means a cell with 16x16 pixels in the second line and second row of the grid, counting from the top left corner of the room).

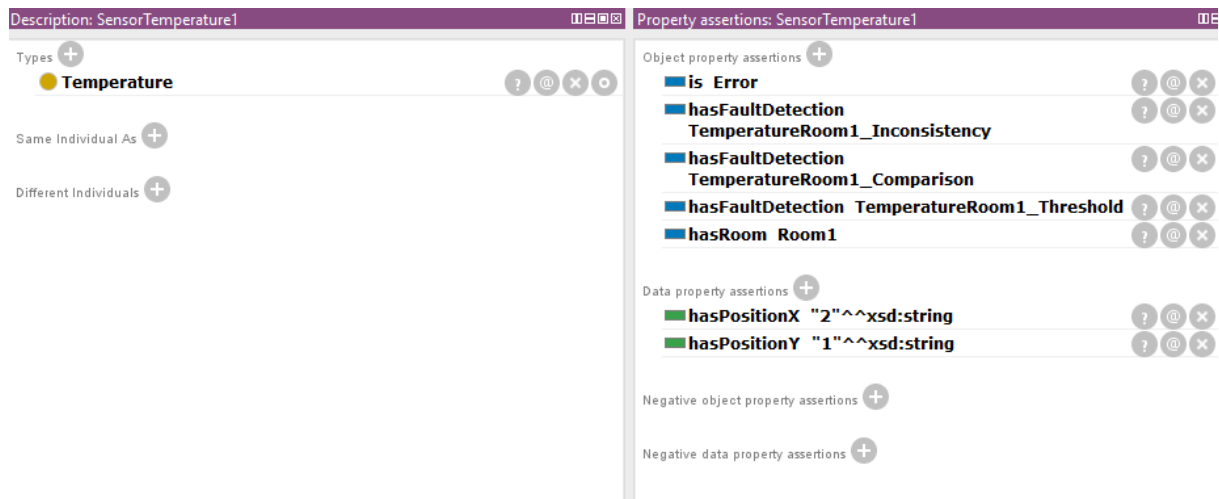


Figure 4-3 - Individual Property Assertions Example.

This dissertation also introduces a new ontology, the Mapping ontology, to be related to the C2NET ontology. This ontology serves as basis for the mapping knowledge base and it was also modelled in Protégé. The respective OntoGraf is presented below in Figure 4-4.

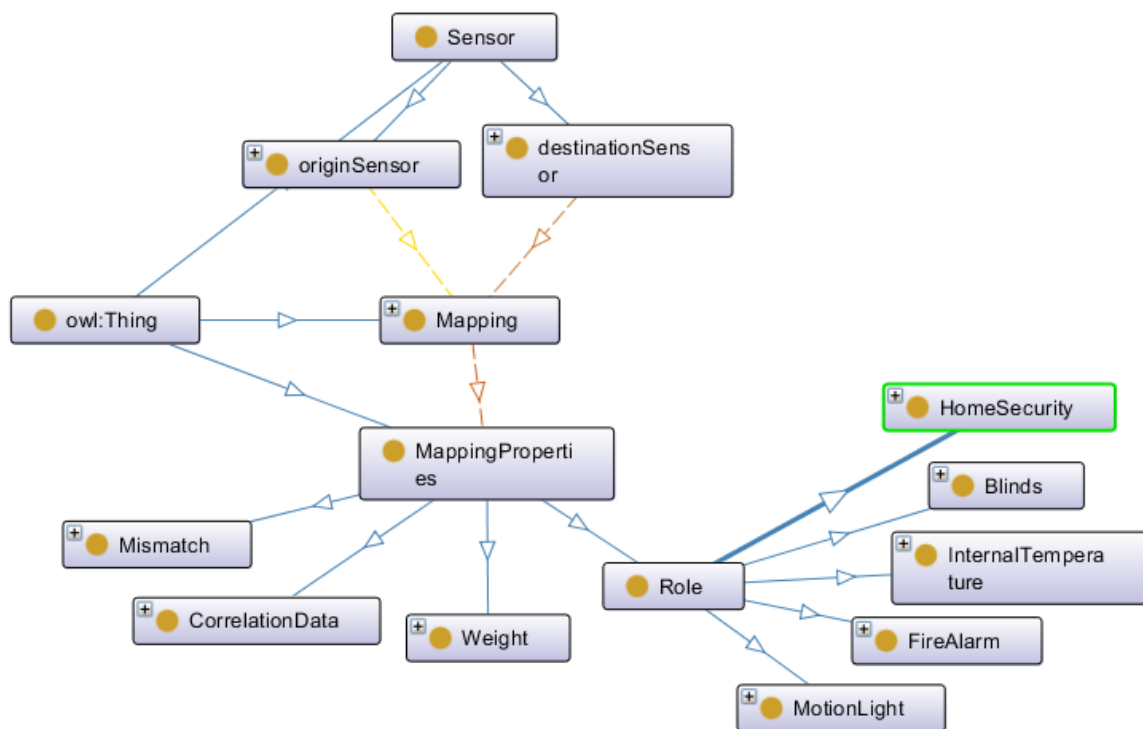
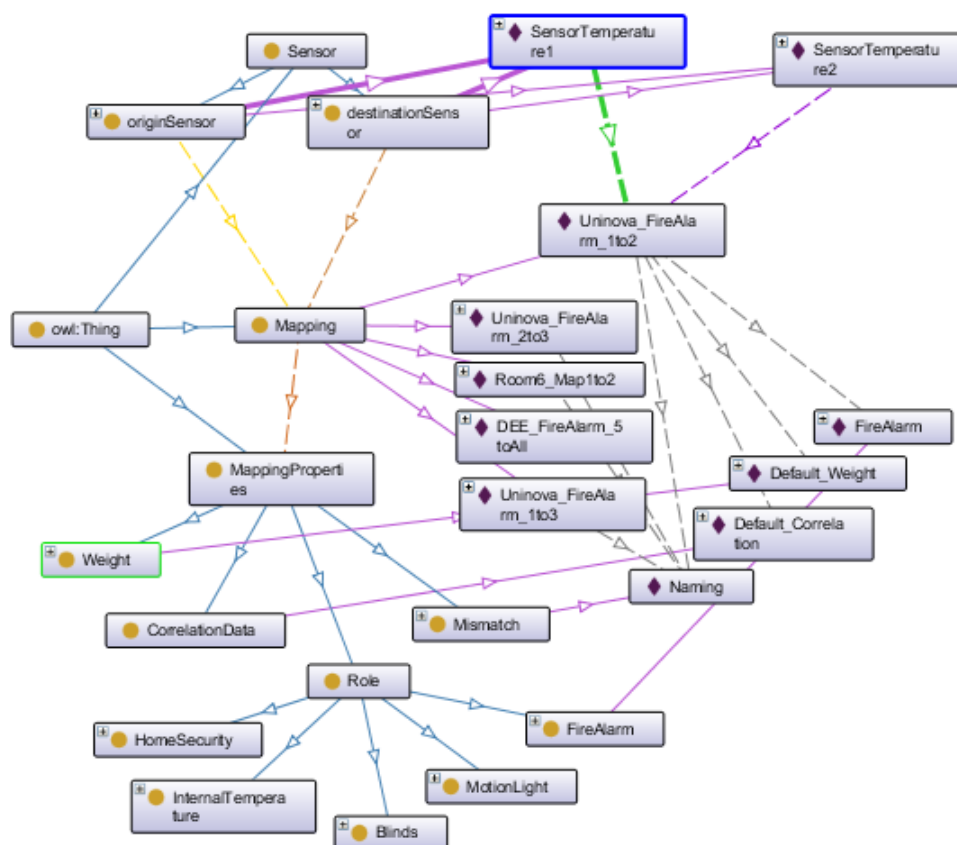


Figure 4-4 - Mapping Ontology OntoGraf.

This model (Figure 4-4) is the representation in Protégé of the model designed in section 3.1 and presented in Figure 3-5. This model is the basis for the concept and methodology of the semantic maps by describing how the sensors interact with the maps and the properties included in it. The way this model works and is represented was also mentioned in section 3.1 but is important to mention that the sensors are viewed as the origin or destination of the semantic map. There can only be one origin sensor in each map but it is possible to have various destination sensors, to aim for difficult scenarios where there

are no equivalent sensors (relative to the origin sensor) in the surroundings. The mapping properties were thoroughly detailed in Equation 1 of section 3.1, and are not going to be further explained here, but here we can see some typical examples of sensor roles in a network such as Fire Alarm, Internal Temperature and Home Security.



It was mentioned before that this Protégé tool, OntoGraf, allows filtering the elements of the model and Figure 4-5 was manipulated using that functionality, in order to display an example of the use of semantic maps. This example is also designed to be part of the testing to be presented in the next chapter. So, in Figure 4-5, two sensors (*SensorTemperature1* and *SensorTemperature2*), some semantic maps (e.g. *Uninova_FireAlarm_1to2*) and some mapping properties are visible (e.g. *Default_Weight*). These elements are instances inserted into the model and represent various scenarios, but for now we should focus on *SensorTemperature1*. This sensor is connected to both *originSensor* and *destinationSensor* subclasses, that means that it is used for more than one semantic map, because having it as both *originSensor* and *destinationSensor* in the same semantic map does not make sense (when using the semantic maps, we assume the failure of the origin sensor, making it unusable for further roles). With that in mind, all semantic maps using *SensorTemperature1*

were filtered as the Figure shows, and only one is connected to the sensor, *Uninova_FireAlarm_1to2*. As the name suggests, *SensorTemperature1* is used as origin sensor for the map. The other sensor presented in the Figure is *SensorTemperature2*, also connected to the semantic map *Uninova_FireAlarm_1to2*. Since no other sensor is connected to this map, we can safely assume that *TemperatureSensor2* is the destination sensor of the mentioned semantic map. This semantic map has then connections to mapping properties that detail it, such as *Naming* (a Mismatch) and *FireAlarm* (the Role of the origin sensor, that the destination sensor will inherit). This simple example represents a semantic map that simply applies redundancy from one temperature sensor to another, but more complex examples are shown in Chapter 5.

It is important to mention, as was hinted in Chapter 3, that these ontologies must be compatible with each other and have concordant information, otherwise they would be unusable together. In the implementation, the main classes and concepts of the ontologies are maintained to prove the proposed architecture, but the instances are manipulated in order to provide some insight of possible applications and to test them. The way this implementation does that is to use the OWL file, that Protégé produces, as database file, using the instances inserted into the ontology models as the required information for testing (sensors, rooms, semantic maps and other mentioned related information).

4.3.3 Step 3 – Jena and Esper setup on Eclipse with Java

In this step, the setup of Jena and Esper was carried. Jena allows loading the models from the OWL file in order to manipulate and query the current state of the ontology and respective instances. The implementation of the Esper libraries on Eclipse (the IDE used for the development of this implementation programming code in Java) adds automatic functions to create complex event processing methodologies, namely statements (to be tested by events) and listeners (to read the events). These libraries were explained in section 4.2 and despite being very important for this implementation, there is no relevant additional information to add for now about them.

4.3.4 Step 4 – SPARQL Queries and manipulation of OWL files

To retrieve information about the instances (elements of the ontologies), the SPARQL standard is used in this implementation, making use of Jena library functions. The queries allow filtering the results as mentioned in section 4.2 and an example is provided in Figure 4-6.

```

public static String destination(String sensor_name, String role) throws JSONException {
    Model model = FileManager.get().loadModel("D:/Faculdade/Dissertação/Protege-5.1.0/Protege Projects/thesis.owl");
    ByteArrayOutputStream JSON = new ByteArrayOutputStream();
    String test = "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
        + "PREFIX owl: <http://www.w3.org/2002/07/owl#>"
        + "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"
        + "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"
        + "PREFIX : <http://www.semanticweb.org/fabio/ontologies/2017/1/untitled-ontology-20#>"
        + "SELECT ?sensor "
        + "WHERE { "
        + "    :"+sensor_name+" :Precedes ?mapping."
        + "    ?mapping :Has :"+role+"."
        + "    ?sensor :Succeeds ?mapping."
        + "}"
        + "ORDER BY DESC (?weight) LIMIT 1";
    Query query = QueryFactory.create(test);
    QueryExecution qexec = QueryExecutionFactory.create(query, model);
    ResultSet results = qexec.execSelect();
    ResultSetFormatter.outputAsJSON(JSON, results);
    String json = JSON.toString(); // System.out.println(json);
    int a = json.indexOf("#");
    if (a != -1) {
        String b = json.substring(a);
        int c = b.indexOf(" ");
        String d = b.substring(1, c-1);
        return d;
    }
    else return " ";
}

```

Figure 4-6 - SPARQL Query Example.

In this example, the SPARQL query is intended to filter, from all the sensors in the Mapping ontology, the destination sensor(s) of a semantic map that has the origin sensor “*sensor_name*”, maps its specific role (there could be more) and has the greatest weight (in case of existing multiple maps that comply with these conditions, the one with the greater value is chosen).

To accomplish the reverse process of inserting and manipulating information present in the OWL files, a specific Java class was developed. This class consists in opening the database file (OWL file, in this case), search specific information if that is the case, and change or add the content based on the parameters of the functions developed for this class. The main functions include adding and editing sensors, maps, rules and roles. The information is not easily readable *per se*, therefore some specific conversions are made with the directives provided by the user interface and the information in the format of the database file.

4.3.5 Step 5 – Sensor Threads

Regarding the simulation integrated in the implementation of this thesis, it is obviously necessary to achieve an environment the closest possible to a real scenario, to provide a reliable idea of how this solution is viable. With that in mind, it was mentioned before that the sensors represent the lower layer of the architecture and do not depend of the monitoring system, therefore should be independent of the rest of the implementation. As (Theunis, Stevens, & Botteldooren, 2017) define, in their work, the sensor is, only, the sensing element that transforms an external physical property into an electrical response. In the simulation, the sensors are represented by independent and parallel threads that output measurements, to which the monitoring system may consider or ignore, similarly to a real network. These sensor threads use the information provided by the database files and function based on that (active when the sensor is on, suspended when the sensor is standby, etc). There is no significant logic or decision-making, like a regular sensor normally operates,

except for the output measurement that varies depending on the input parameters (it can be randomly generated or it can be a selected value).

4.3.6 Step 6 – Fault Detection (by Esper and by Ontology)

After defining the methodology of the sensing devices and the code that deals with the simulation processing, it is important to finally converge to the main focus of this work, the semantic maps. And to achieve that, the conditions that lead to the use of semantic maps must be implemented in the simulation in order to *“make things happen”*.

With Esper, is relatively easy to set up a fault detection methodology. First, the configurations for managing the complex event processing are set accordingly to the scenario. After that, the EPL statements (explained in section 4.2) are defined, using keywords to filter the type of event and time window that triggers the statement.

```
//Event Statements
//A and B above 70 (sum), but sensors match temperature by at least 5 degrees
EPStatement cepStatement_triABC_A = cepAdm.createEPL("
+ "select * "
+ "from Sensor_Reading(name='SensorTemperatureA').std:lastevent().win:time(3) as incendio1, "
+ "Sensor_Reading(name='SensorTemperatureB').std:lastevent().win:time(3) as incendio2, "
+ "Sensor_Reading(name='SensorTemperatureC').std:lastevent().win:time(3) as incendio3 "
+ "having"
+ " (incendio1.reading+5)<((incendio1.reading+incendio2.reading+incendio3.reading)/3) "
+ " or (incendio1.reading-5)>((incendio1.reading+incendio2.reading+incendio3.reading)/3) "
);
```

Figure 4-7 - EPL Statement Example

Figure 4-7 shows an example of an EPL Statement used in the implementation. This statement is triggered when the requirements of the expression are met, in a time window of the last three measurements of the three sensors of this example (*SensorTemperatureA*, *SensorTemperatureB* and *SensorTemperatureC*).

After defining the statements, event listeners for each specific statement must be created to allow the detection of the event of interest.

The use of Esper is recommended for all the reasons mentioned in section 4.2.3, but has some restrictions because of the pre-defined filtering methods and the fact that it is relatively difficult to automatically generate the statements. Considering this and noticing that an approach that can directly relate, inside the ontology models, specific fault detection methods with each device, can be useful, an experimental ontology was developed. This ontology is not aimed to substitute the use of Esper, but simply to provide an alternative way of carrying this process, to compare the effectiveness of both methodologies.

The ontology for fault detection was mentioned and explained in section 3.3.3, and is integrated inside the C2NET ontology model. With this method, each fault detection instance is associated with an ID and, possibly, other data properties, similarly to the sensors, and can be related by a class property with the sensors. The beneficial points are to consider these methods inside the ontology and database, allowing direct correspondence to the sensors,

allowing easy and automated manipulation of the conditions during the implementation (due to being present in the always updated knowledge bases and not in the initial programming code) and allowing the creation of typical fault detection profiles such as a “*fault detection instance for temperature sensors for fire alarm roles*” (which can be directly associated with a new sensor in the network, instead of manually creating a statement). Of course, this method has predictable disadvantages, and the most significant ones are the fact that demands the inclusion in the initial design of the ontology model (which may be a time-consuming process, relative to the use of Esper) and the runtime processing time which is surely greater than using the optimized components of Esper.

Nevertheless, in this dissertation, both approaches were implemented and tested.

4.3.7 Step 7 – Recovery with Semantic Maps

Finally, the last functional step of the implementation is here presented. After all the “*fuss*” discussed in the previous sections, it all concludes in the recovery of a network failure using semantic maps.

Despite the method of fault detection (using Esper or by ontology), the recovery of a sensor failure is conducted in the same way, as presented in section 3.2. The program searches, by query, the origin sensor information, queries again for existing mapping solutions and again for the destination sensor(s) (if any semantic map was found). In the simulation, the origin device is switched to *Error* mode, the colour of the representation of the sensor turns black and the device becomes non-targetable by other semantic maps, and the destination sensor(s) turns green, acquiring the role from the origin sensor. The knowledge bases are updated, and the event processing is updated too.

4.3.8 Step 8 – Implementation of a Graphical User Interface

All the steps presented before, converge in the implementation of a GUI. This was the selected way of showing the architecture processing working while providing user input information. The importance of such feature was mentioned in section 2.3.1 and the interaction between the user and interface follows the pattern defined by the CASCoM system-level configuration model explained in the same section of this thesis and depicted in Figure 2-12.

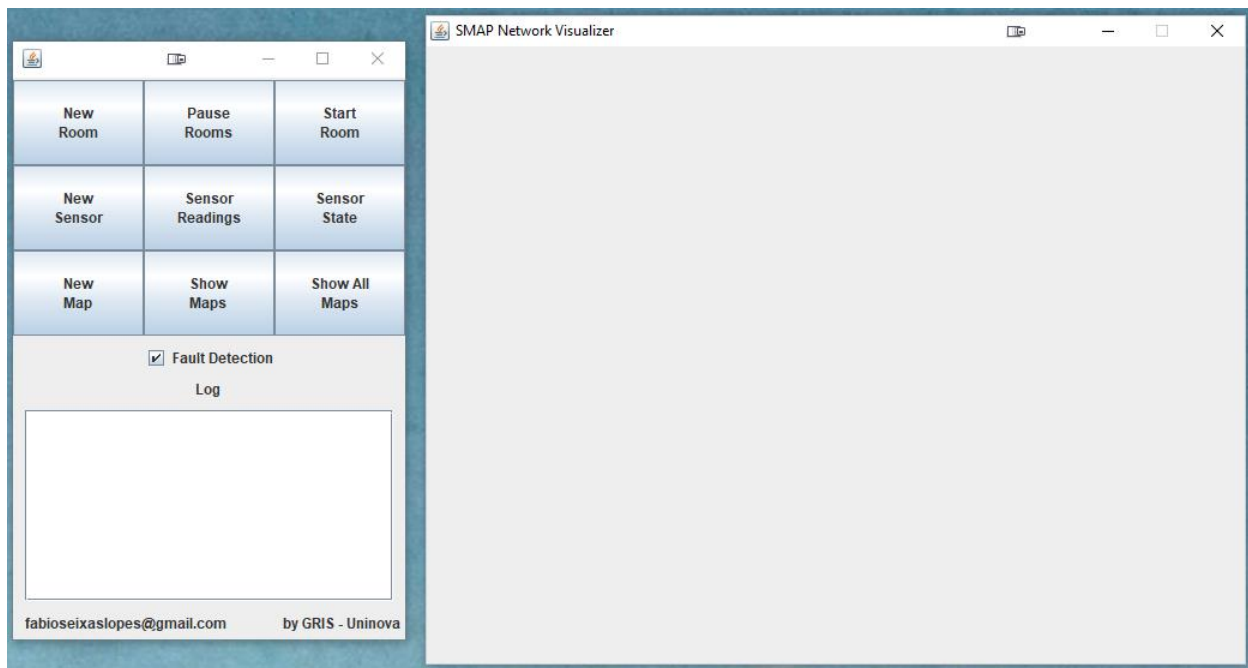


Figure 4-8 - SMAP Graphical User Interface.

The SMAP graphical user interface (depicted in Figure 4-8) is divided into two windows: the control window and the network visualizer. To allow a better comprehension of the functions in each one of them, small descriptions are presented in this section, that will be complemented with the use of those same functions during the operation of the sensor network, in the scenarios of the next chapter of testing and validation.

Control Window:

- **New Room** – Allows the creation of a new room;
- **Pause Rooms** – The functioning sensors, if any, inside the rooms, switch their operating state to *Standby*;
- **Start Room** – Starts an existing room, starting the threads of any sensor with its operating state as *On*;
- **New Sensor** – Allows the creation of a new sensor;
- **Sensor Readings** – The user can choose the measurement output of any sensor of the room (randomized values are an available option);
- **Sensor State** – The state of any sensor of the room can be changed with this button. The possible operating states are: *On*, *Off*, *Standby* and *Error*;
- **New Map** – Allows the creation of a new semantic map;
- **Show Maps** – Shows the semantic maps for a selected sensor of the room (only semantic maps that have that same sensor as origin sensor) with a line connecting to the respective destination sensor(s);

- **Show All Maps** – Shows all semantic maps for all sensors in the room;
- **Fault Detection** – This box, turns on and off the fault detection ontology method, explained in step 6 of this implementation, for testing purposes;
- **Log** - Gives testing feedback and displays the measurements of active sensors, of the current room, in real-time.

Network Visualizer Window:

- **Room** – The window adapts to the size of the room (scaled), and shows any background associated with it. The rooms are divided into cells by an invisible grid of 16x16 pixels in which the sensors are shown;
- **Sensors** – The sensors are represented by circles, within the mentioned cells (in which, for example, X=3 and Y=1 means a cell with 16x16 pixels in the first line and third row of the grid, counting from the top left corner of the room), with one of the four colours that represent their current operating state (*On* is green, *Off* is red, *Standby* is yellow and *Error* is black). There can be more than one sensor in the same cell. When clicking on the cell, an additional window for each sensor in that position appears, called Sensor Information, and gives additional data such as the sensor types and their last three measurements. The sensors that are *On* also have a small grey circle flashing in the centre as a visual aid to help understand which sensors are providing measurements;
- **Semantic Maps** – The semantic maps are represented by lines connecting the origin sensor to the destination sensor(s) and only appear in the graphical user interface when one of the two buttons for that effect is selected.

This GUI was protected against the most common wrong data formats and fields, when the user is inputting information recurring to the Control Window, avoiding user induced errors in the simulation.

The next chapter provides testing and validation using the program implemented with these steps. Some aspects, such as the GUI, are easily depicted when recurring to practical demonstrations of its functionalities.

This section of the thesis addresses the testing of the implementation by validating the requirements and functionalities of the system, that were mentioned previously in chapter 4. The testing methodology used is explained along with the testing implementation that was developed. Following that, the validation of the hypothesis mentioned in chapter 1 is detailed and the validation in terms of scientific and industrial importance is addressed.

The reader is reminded that testing is an operational way to check the correctness of a system implementation by means of experimenting with it (Tretmans, 1999) and represents a process with its own effectiveness. And since the testing of realistic systems can never be exhaustive, because systems can only be tested during a restricted period, testing cannot ensure complete correctness of an implementation. It can only show the presence of errors, not their absence (Tretmans, 2001).

5.1 Testing Methodology

The methodology presented in this chapter, for the testing of the implementation, is intended to provide insight of the effectiveness of the solution proposed in this dissertation. Therefore, it is relevant to use proper methods, already proven and established in other academic publications, to ensure reliable results. These methodologies and standards were defined and revised throughout the years, based on the expertise of using them and their practical results, sometimes recurring to a superimposition of a multitude of these to cover complex implementations (Ferreira, 2012).

There are many testing methodologies in software engineering, such as unit testing, conformance testing, abstract concepts like black box testing, etc (White, 1987). But for testing, not only in this environment but also generally, the functional and structural processes are distinguished (Myers, Thomas, & Sandler, 2004). Other distinguishable characteristic is between dynamic tests (execution of code) and static tests (e.g. code review) (Anderson, 2011), but only dynamic tests are considered here.

Structural testing is based on the internal structure of a computer program, where all code is analysed, and each line is executed at least one time, covering all the possible paths of the execution of the program. This analysis is also known as the white-box testing, where tests are derived from the program code (Anderson, 2011).

Functional testing is based on observing, externally, the program execution, regarding only the objectives intended to be tested. This type of testing is known as black-box testing, because the internal content of the program is ignored (Zeller, 2017).

Since the functional tests derive from the specifications, the main goal is to analyse if the implementation is in fact working as expected. Consequently, due to the nature of these testing methods, while structural testing is used in early stages of the software development, functional tests are the main focus in later stages of development (Ferreira, 2012).

With the information presented here, and by analysing some similar publications, the methodology chosen for this testing procedure is based on a known standard, named Tree and Tabular Combined Notation (TTCN, or Test Notation Standard).

Tree and Tabular Combined Notation (TTCN)

The TTCN is a standardised notation by the ISO/IEC 9646-1 for the specification of tests for software systems and has been developed within the framework of functional testing. The tests are defined through tables which are divided in a general description, constraint, behaviour and verdict.

The test is defined as a sequence of events which represent the steps of the testing process on a given system. Each event has its own sequence level, visible by text indentation, and can be one of two types: action or question. Actions are preceded by an exclamation point and represent the actions performed on the system. Questions are preceded by an interrogation point and represent evaluations of the output of the system after one or more actions are completed. The answer can be positive or negative, therefore, multiple questions can exist at the same indentation level, covering all possible outputs of the system. After completing a TTCN test table, a verdict must be deliberate. The possible outcomes are: *“Success”*, *“Failure”* or *“Inconclusive”*. This verdict is based on the sequence of events and is conditioned by the output of the system and question events. Example scenarios can be found in (TTCN-3, 2013).

Regarding the presented methodology, Figure 5-1 shows an example scenario for the testing implementation. Different scenarios will be presented in the next section, but it is relevant to describe and show a diagram representing a variant of them, to allow a better comprehension of the overall testing methodology.

In this example, the CEP uses three sensors for a specific role (temperature sensor A, CO₂ gas sensor X and CO gas sensor Y). The gas sensors measure a normal concentration of a given gas, but the temperature sensor A measures a very high temperature. After the SMAP verifies the operation of the temperature sensor A, notices the malfunctioning of it and provides a semantic map. This semantic map provides a solution to substitute the temperature sensor A with a new temperature sensor, B. The information about the needed update of the rules database in the CEP is passed from the SMAP to the CEP engine. After that, the CEP starts using the temperature sensor B instead of the temperature sensor A, and the SMAP changes the state of the temperature sensor A, in its knowledge bases, as faulty (Error, in the SMAP sensor state nomenclature).

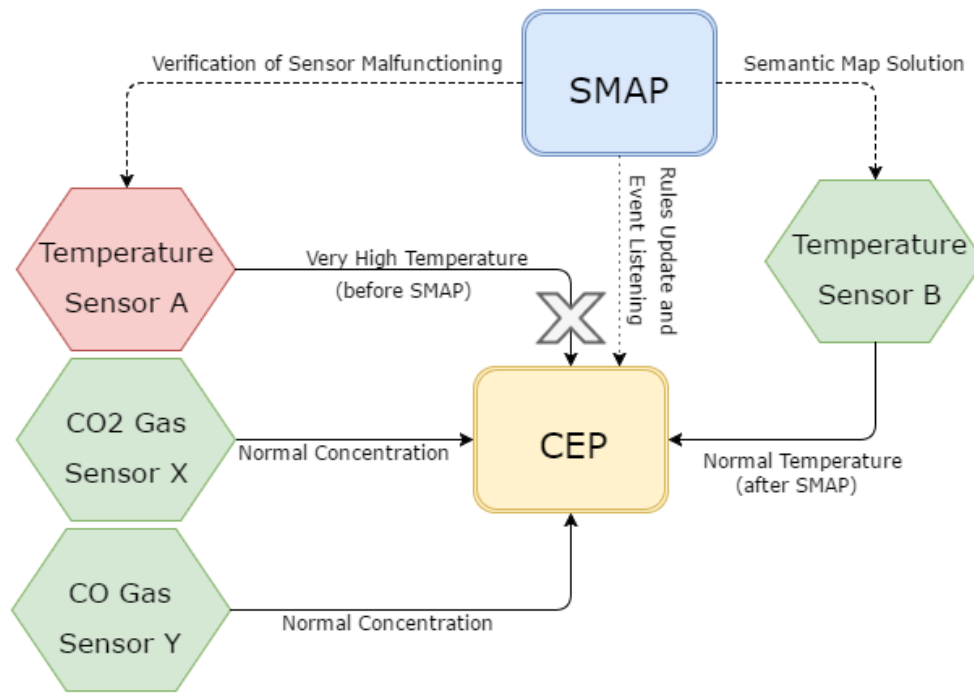


Figure 5-1 – Example Scenario of Testing Implementation.

Most of the processing occurs inside the SMAP module, during the search for the appropriate semantic map, but it is disregarded in this stage due to the testing being from a functional perspective. The representation of the Figure 5-1 is not going to be repeated to the upcoming scenarios and it is intended to serve only as an aid to understand the testing process.

5.2 Testing Implementation

This section presents the testing of the implementation presented in chapter 4, using the methodology of the previous section (5.1). The scenarios presented here are intended to show the functionalities of the developed program, using the graphical user interface to manipulate and show the current state of the network in each scenario. This section is organized by three “Rooms” in which the scenarios are designed to provide feedback of the functionalities. “Room 1” and “Room 2” present a general overview of all processes and are representations of real case scenarios for two specific situations, implemented for the validation of the C2NET project in a factory of a company called “*António Abreu Metalomecânica, Lda*”¹, located in Vila Nova de Famalicão, in the north of Portugal. “Room 3” presents an extended theoretical application intended for testing purposes and provides some additional information.

Room 1: SMAP Initialization and Use of Semantic Maps

As can be seen in Figure 5-5, this room has four motion sensors:

¹ <http://www.aametalomecanica.com/>

- AAMM_Pass_1: Ultrasonic sensor, active sensor on the top left corner of the SMAP Network Visualizer;
- AAMM_Pass_2: Infrared sensor, inactive sensor on the top left corner of the SMAP Network Visualizer, redundant to sensor AAMM_Pass_1;
- AAMM_Pass_3: Ultrasonic sensor, active sensor on the bottom right corner of the SMAP Network Visualizer;
- AAMM_Pass_4: Infrared sensor, inactive sensor on the bottom right corner of the SMAP Network Visualizer; redundant to sensor AAMM_Pass_3.



Figure 5-2 - Infrared Sensor.

These sensors represent the four motion sensors implemented in a station, of the mentioned factory, responsible for detecting the passage of manufactured pieces. The type of infrared sensors (model GP2Y0A02YK0F) used in the implementation in the mentioned factory, is illustrated by Figure 5-2 and the type of ultrasonic sensors (model HC-SR05) is illustrated by Figure 5-3.



Figure 5-3 - Ultrasonic Sensor.

The sensors are placed in pairs, in the top left corner and the bottom right corner (as appears in the SMAP Network Visualizer of the following figures), and, initially, only one

sensor of each pair is active, the other one is only for redundancy purposes (as can be seen in Figure 5-5). This redundancy is achieved with the use of semantic maps, represented by a blue line connecting the involved sensors, as can be seen in Figure 5-6.

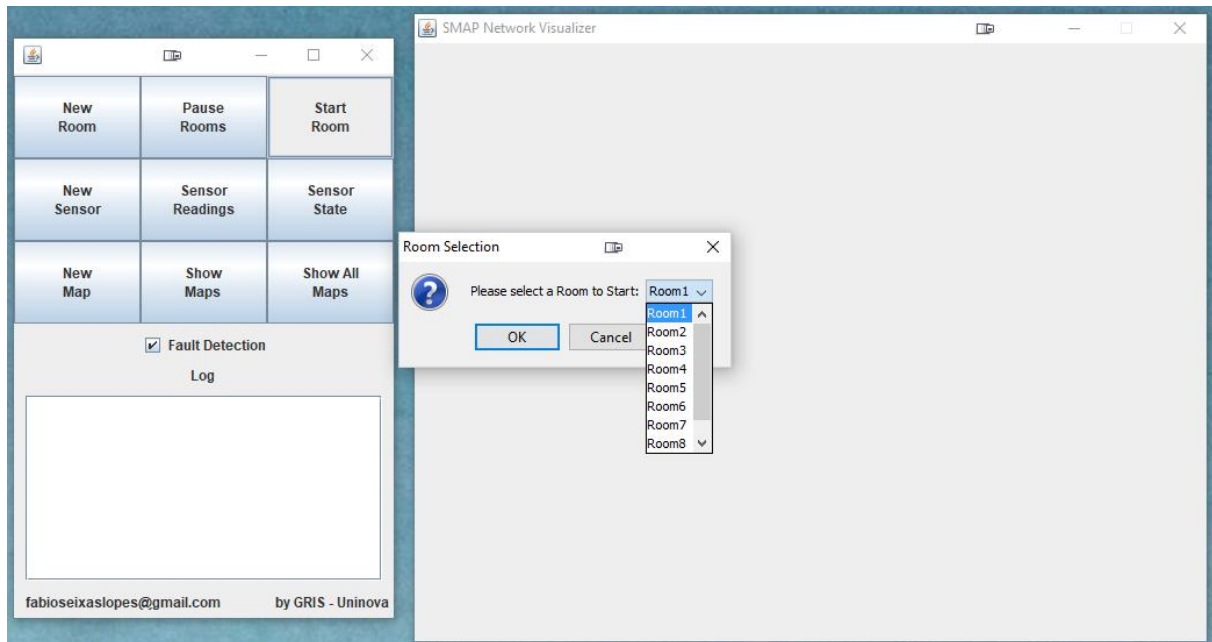


Figure 5-4 - Room 1: Start Room.

Figure 5-4, shows the graphical user interface room selection process. The dropdown list of rooms results from a query to the C2NET ontology to get all the existent rooms in the database file created for this implementation, using Protégé.

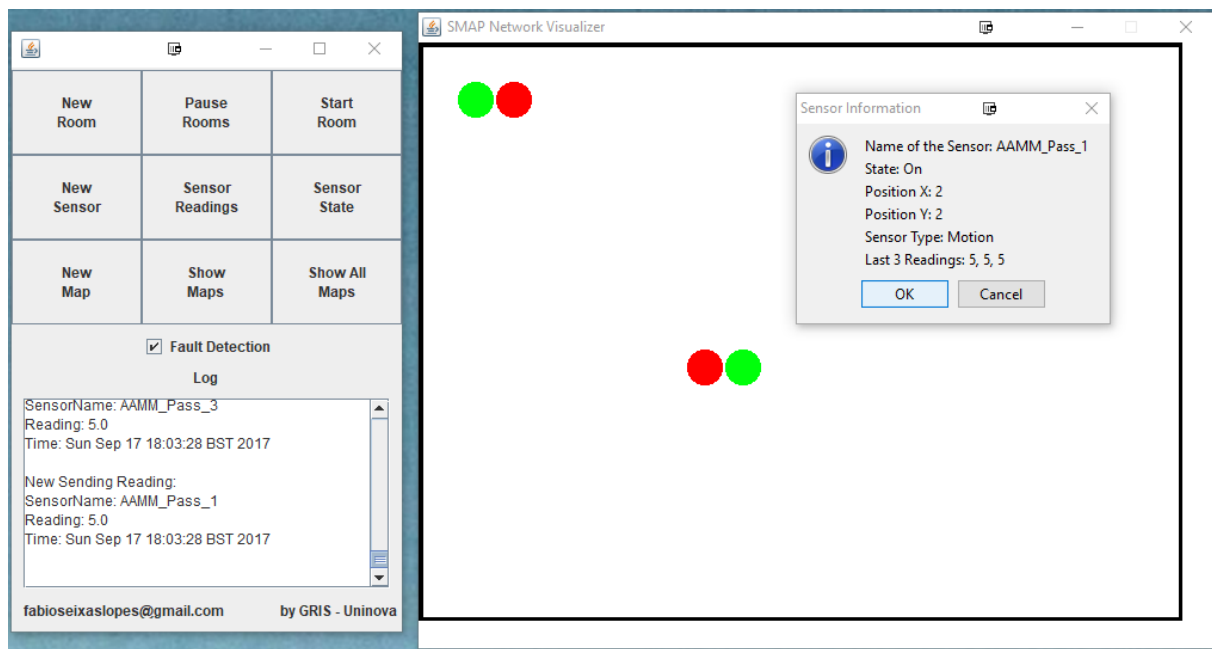


Figure 5-5 - Room 1: Room Started.

Following that, Figure 5-5 shows *Room 1* functioning with the Log showing the measurements in real time from the two active sensors (only two are green), the SMAP

Network Visualizer showing the existent sensors in the room (four motion sensors, two *On* and two *Off*) and by clicking in the green circle, on the top left corner, we get access to the sensor information from sensor *AAMM_Pass_1*, displayed in an additional box as seen in the mentioned figure.

In Table 5-1, the Test Case 1 is provided. This TTCN test table is based on the previous two Figures and introduces this testing implementation with the start of a room and respective sensors. By analysing the table and Figure 5-5, we can conclude that this step was a *Success*.

Table 5-1 - Test Case 1.

Test Case 1		
Test Case: Room Start Group: Room 1 Tests Purpose: Check if Room 1 starts and the active Sensors run.		
Behaviour	Constraints	Verdict
! Press Start Room ? Rooms Loaded ? Room Started ? Sensors Started ! Click in the Sensor ? Got Sensor Information OTHERWISE OTHERWISE OTHERWISE OTHERWISE		 Success Failure Failure Failure Failure

Next, in Figure 5-6, the button to show maps from *AAMM_Pass_1* was clicked and the interface showed redundancy to the nearby sensor, *AAMM_Pass_2*, with a blue line connecting the centers of the respective circles. To increase the complexity of this testing procedure, the “New Map” function was used to create two new semantic maps from *AAMM_Pass_1* to *AAMM_Pass_3* and *AAMM_Pass_4*, individually (this can be seen in Figure 5-7 and Figure 5-8, where two new blue lines appear, connecting the *AAMM_Pass_1* to the mentioned destination sensors). After that, the “Sensor Readings” button was used to change the sensor output of *AAMM_Pass_1* to 300 units. The fault detection methods were active, so, the program notices the failure in the sensor, by the Threshold fault detection method, and asks for recovery using semantic maps (this is depicted by Figure 5-6).

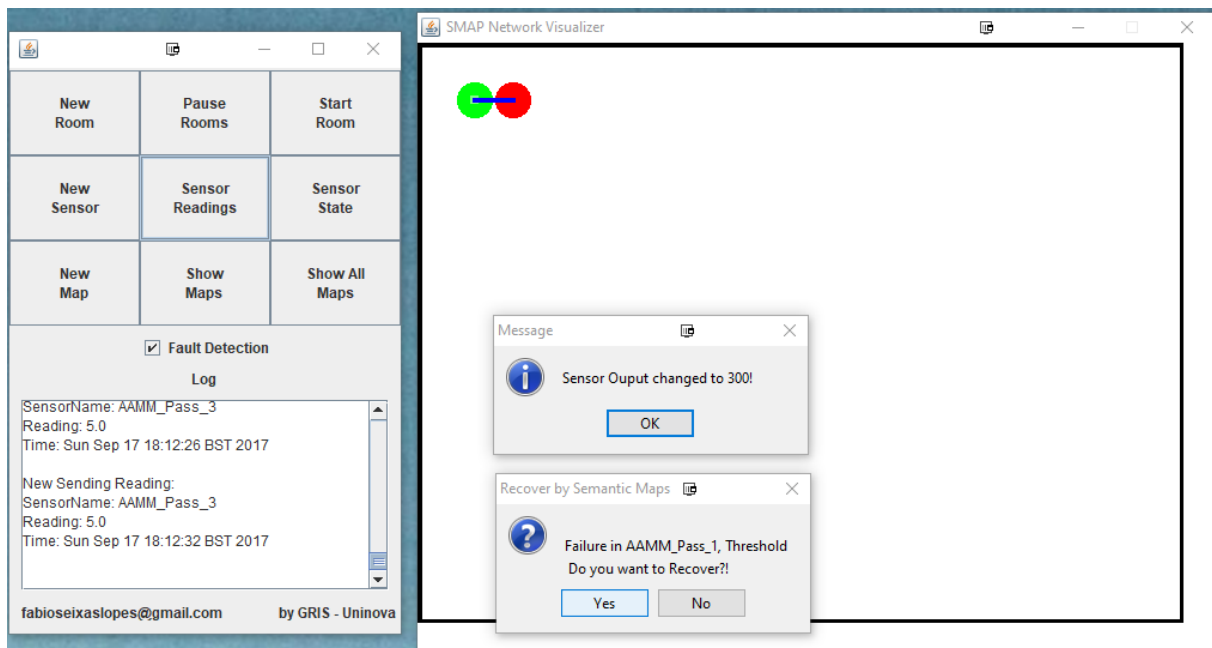


Figure 5-6 - Room 1: Failure in a Sensor.

In Figure 5-7, we can see the options presented to the user to recover from the faulty sensor, *AAMM_Pass_1*. The semantic maps “*AAMM_1to2*”, “*AAMM_1to3*” and “*AAMM_1to4*” are selectable and have as destination sensors, *AAMM_Pass_2*, *AAMM_Pass_3* and *AAMM_Pass_4*, respectively.

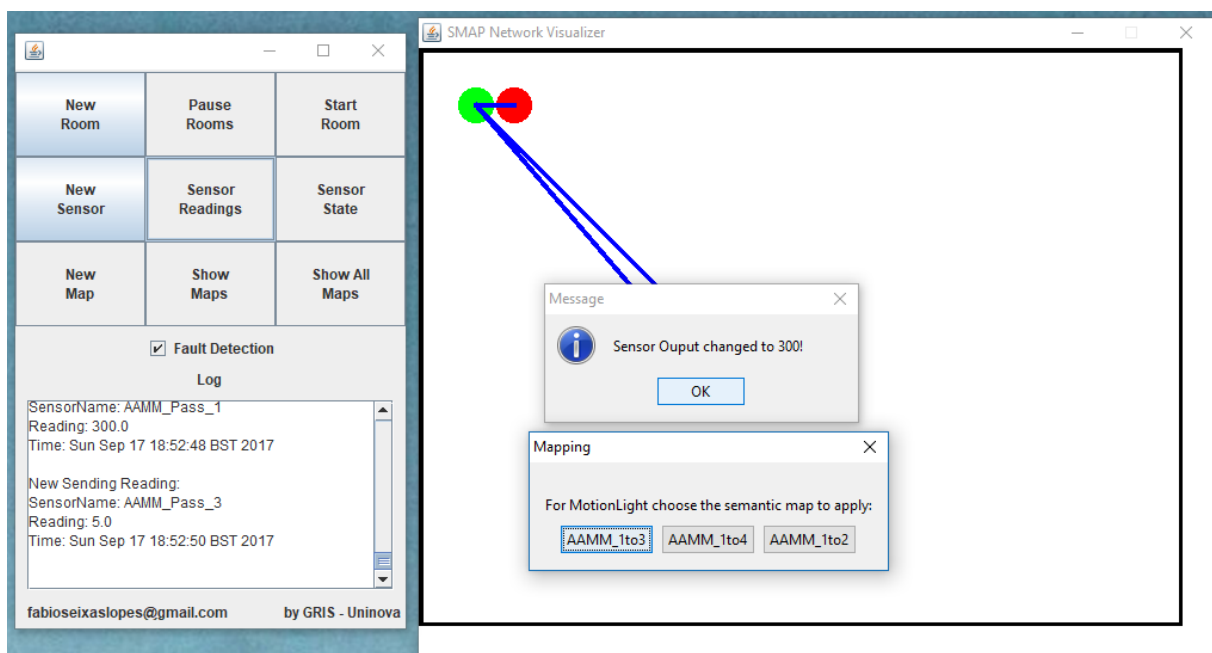


Figure 5-7 – Room 1: Semantic Maps for the Faulty Sensor.

In Figure 5-8, we can see that the user chose the semantic map for the *AAMM_Pass_2* (visible in the Log box, as active), and the origin sensor circle turned black to indicate the state of *Error*, this disables further semantic maps to target it as destination sensor. The destination sensor turned green and, in the Log, we can see its real-time measurements (the measurements from *AAMM_Pass_1* stopped because it is no longer active). This semantic

map selection process is active for this test, but it can be automated using the previously mentioned weight component, present in every semantic map.

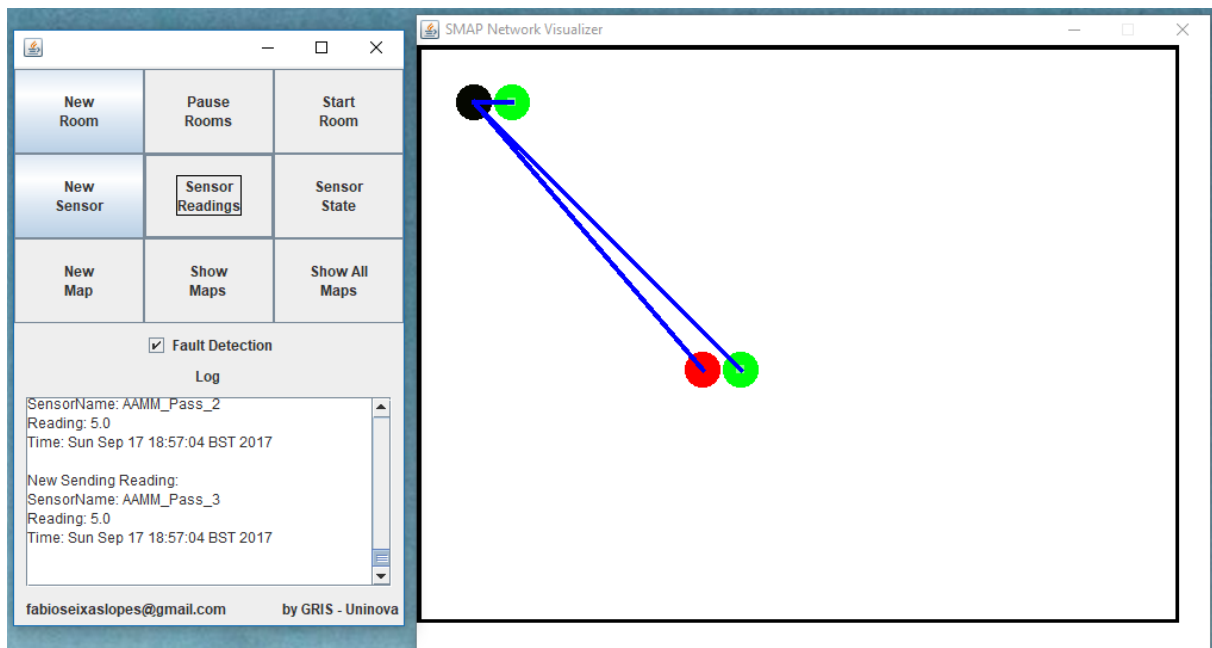


Figure 5-8 - Room 1: Result of the use of a Semantic Map.

The following illustration, Table 5-2, is the test case, relative to the example scenario just provided. By analysing the table of the test case and Figures 5-6, 5-7 and 5-8, we can conclude that the implementation of the use of a semantic map was a *Success* and the objective of this step was accomplished.

Table 5-2 - Test Case 2.

Test Case 2		
Test Case: Use of Semantic Maps Group: Room 1 Tests Purpose: Check if the program handles the use of a Semantic Map, maintaining the network functioning.		
Behaviour	Constraints	Verdict
! Press Show Maps ? Semantic Maps Loaded ! Change Output ? Output Changed ? Malfunction Detected ! Choose Semantic Map ? Network Recovery OTHERWISE OTHERWISE OTHERWISE OTHERWISE		Success Failure Failure Failure Failure

Room 2: Sensor State Changes and Network Recovery

As can be seen in Figure 5-10, this room has four temperature sensors:

- AAMM_Temp1: Thermocouple sensor, active sensor on the top left corner of the SMAP Network Visualizer;
- AAMM_Temp2: Thermocouple sensor, inactive sensor on the bottom left corner of the SMAP Network Visualizer;
- AAMM_Temp3: Thermocouple sensor, inactive sensor on the top right corner of the SMAP Network Visualizer;
- AAMM_Temp4: Thermocouple sensor, inactive sensor on the bottom right corner of the SMAP Network Visualizer.

This room represents the second scenario from the mentioned factory. It is a station with an industrial oven where the manufactured pieces dry after being painted. In strategical locations of the oven, four temperature sensors are used for calculating the average temperature of the room (this calculation is done by the CEP). The type of temperature sensors used, in the factory, was the nickel-alloy type E illustrated by Figure 5-9.



Figure 5-9 - Temperature Sensor.

The objective is to have enough active sensors to do this calculation properly and, if possible, leave others as non-active to allow redundancy. Initially, only one sensor, *AAMM_Temp1* is *On* (as can be seen in Figure 5-10).

In Figures 5-10 and 5-11, a basic functionality, essential from the demonstrative and experimental point of view, is shown. That functionality is just the activation of a certain sensor, in this case *AAMM_Temp2*, to better manipulate the simulation environment and force the desired following testing procedure.

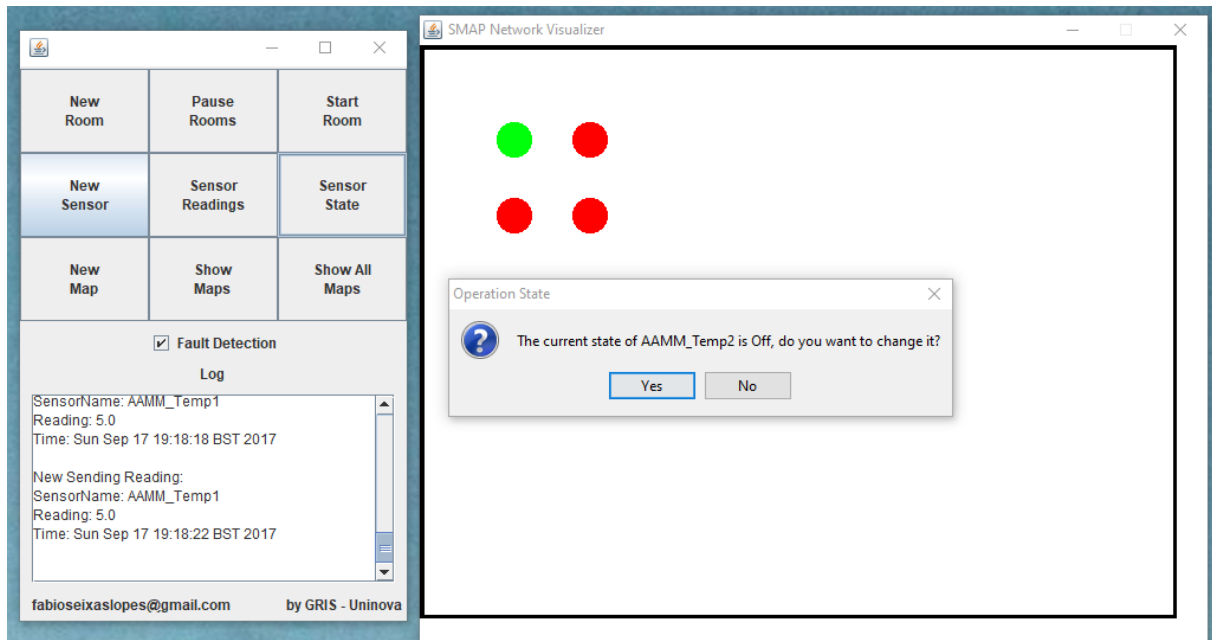


Figure 5-10 - Room 2: Changing the Sensor State.

Figure 5-10 shows the screen after clicking the “Sensor State” button and choosing the desired sensor, *AAMM_Temp2*. The sensor current state is presented (which also can be seen in the SMAP Network Visualizer) and in this case, it’s *Off*. After that, the interface shows a message dialog which requests the new state for the selected sensor and, in this test, the

On state is selected. And finally, Figure 5-11 shows *AAMM_Temp2* as active and, in the Log, we can see the current sensor measurements.

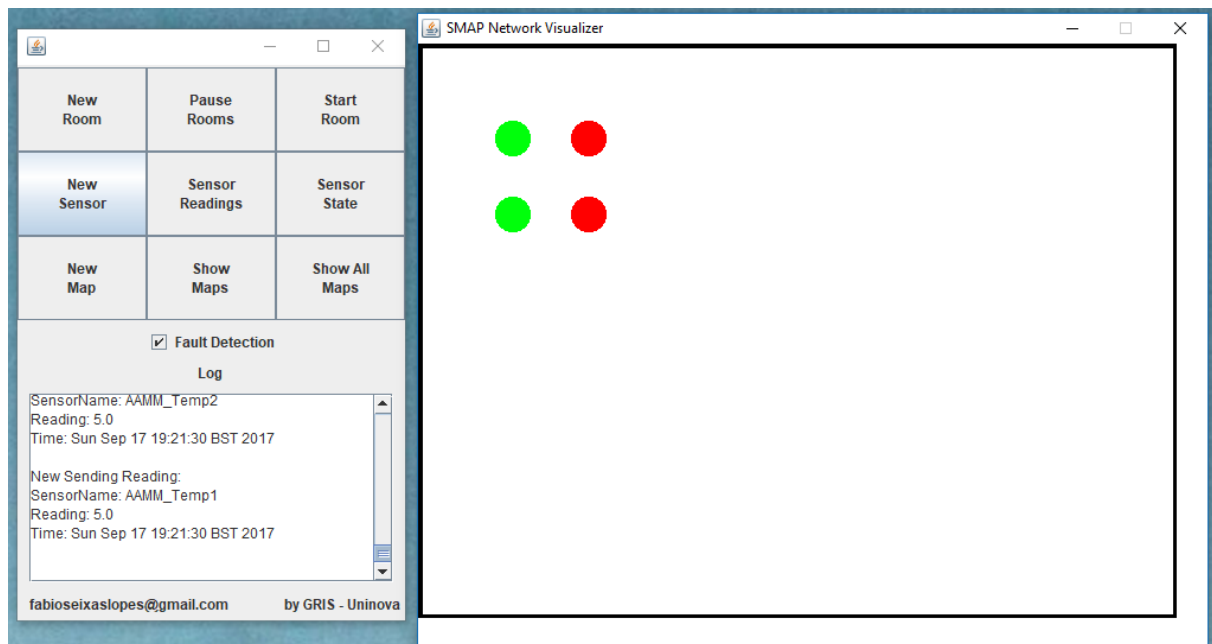


Figure 5-11 - Room 2: Sensor State changed.

Since this previous case of testing is a very simple procedure, the correspondent Test Table, Test Case 3, includes this and the following procedure which represents the redundancy from *AAMM_Temp2* to *AAMM_Temp3* and *AAMM_Temp4* (a semantic map with one origin sensor and two destination sensors) using a failure that invokes the Inconsistency fault detection method. This procedure is shown by Figures 5-12 to 5-14.

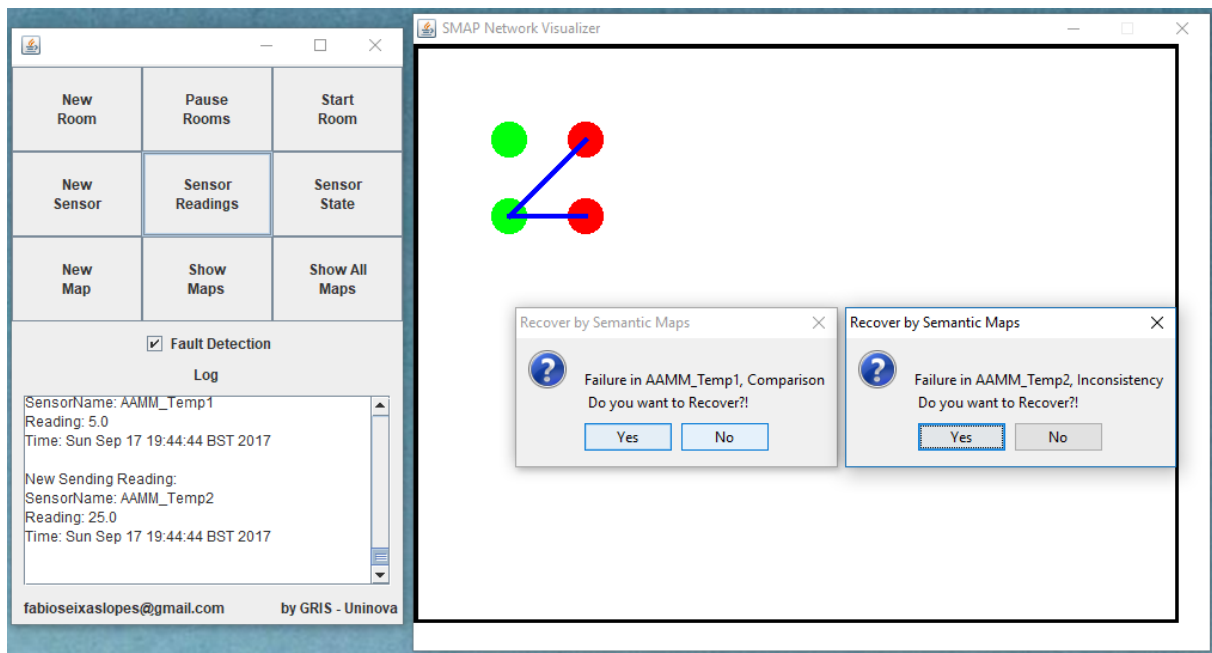


Figure 5-12 - Room 2: Inconsistency and Comparison Failures Detected.

Figure 5-12 depicts the procedure and result of changing the reading value of the sensor, in this case, the *AAMM_Temp2*. It is visible in the SMAP Network Visualizer, that the three sensors share redundancy by a semantic map (depicted by the blue lines connecting the center of the circles). The new sensor output value is within the threshold of the sensor, contrarily to the previous example of fault detection. But the significant change from 5 units to 25 units triggers the Inconsistency method.

As seen in the message dialogs from Figure 5-12, the Inconsistency method was activated, but the Comparison fault detection mechanism was activated too. This occurred because the Comparison mechanism compares the output values of sensors related by geographical interest or proximity distance, using certain proximity thresholds. The developed program can compare innumerable nearby sensors (depending on the processing capability of the hardware running this interface). In this case, the four sensors can be considered due to their proximity but only the active sensors are compared for logic reasons. And with the change made previously, the difference between the two active sensors is 20 units, above the stipulated for this room (10 units), and for that reason, the Comparison method was activated. In order to proceed with the use of the semantic map mentioned before, the user clicks “No” on the Comparison method and “Yes” on the message dialog presenting the activation of the Inconsistency fault detection rule.

Recapitulating, in Figure 5-12, the recovery by semantic maps is activated by the Inconsistency fault detection method that compares the current reading of a certain sensor with its previous ones, considering a limited acceptable variation. In this case, the current sensor reading is 25 and all the previous ones are of 5 units. The limit variation stipulated for this simulation is under 20 units, so the recovery process is automatically initiated.

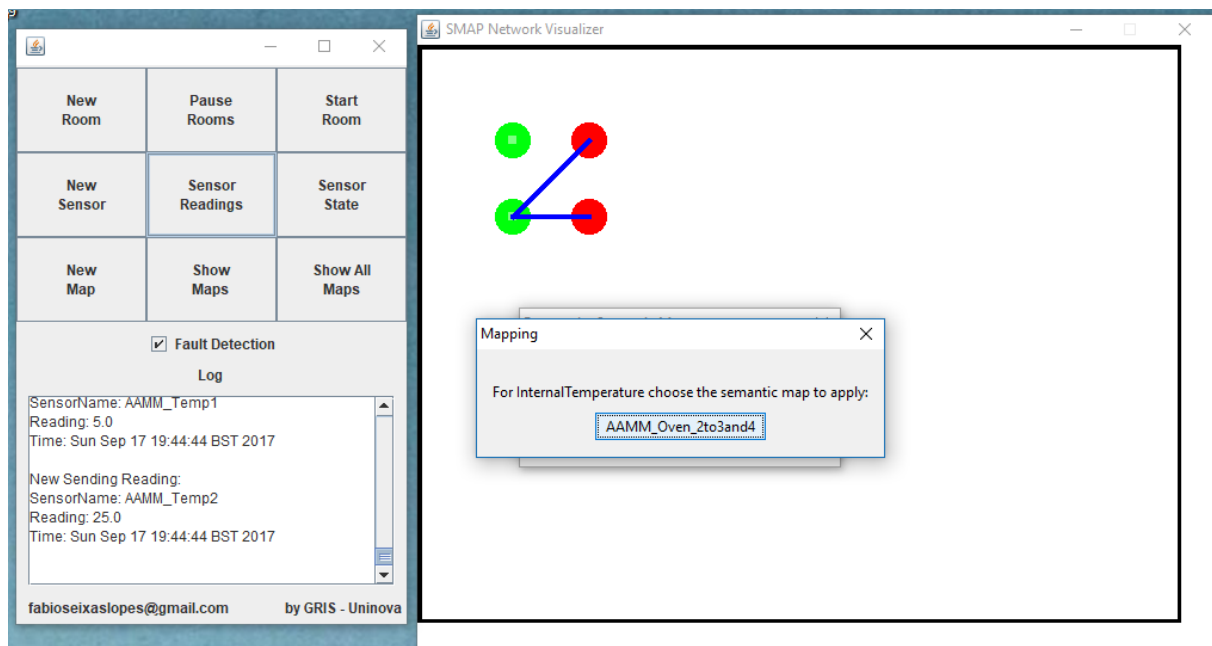


Figure 5-13 - Room 2: Use of a Semantic Map with two Destination Sensors.

As seen in Figure 5-13, there is only one semantic map available for this situation, *AAMM_Oven2to3and4*, and the destination sensors are *AAMM_Temp3* and *AAMM_Temp4*. At this point, the readings from *AAMM_Temp2* are being disregarded because the simulator has undergone into the semantic map redundancy procedure, after the user has chosen to do so.

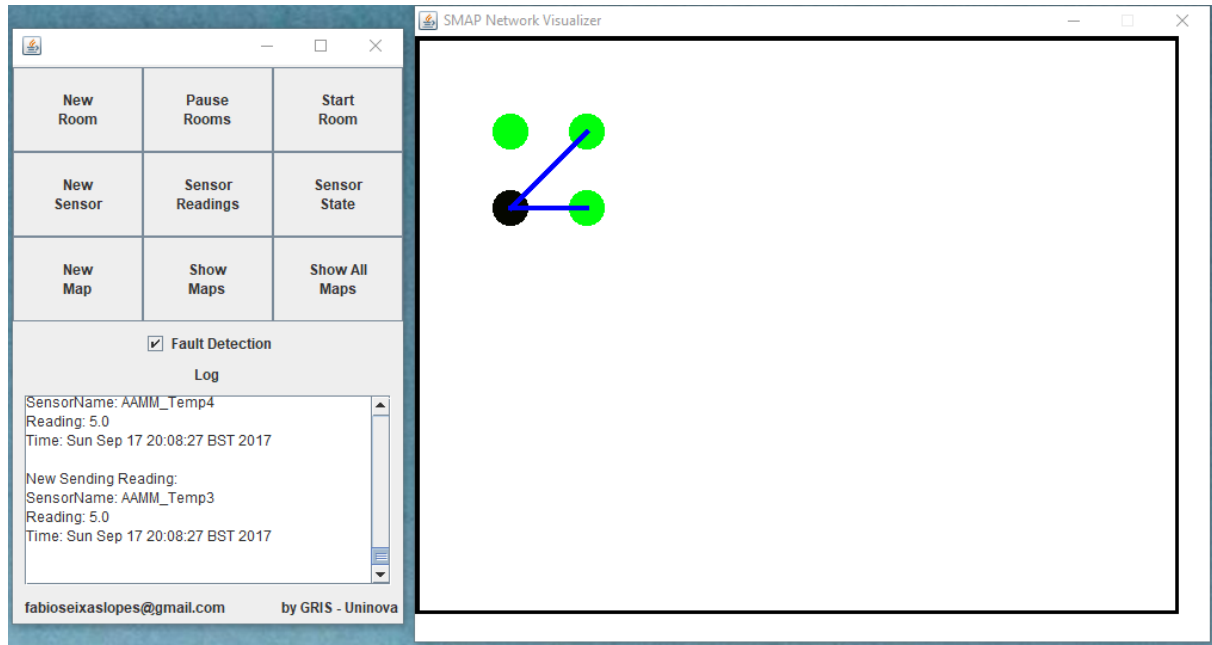


Figure 5-14 - Room 2: Resulting Network State.

Finally, in Figure 5-14, we can see the resulting network state, where the *AAMM_Temp2* state is now *Error* and the destination sensors are now *On*. In the Log box, the output values of the three active sensors are now being displayed.

The correspondent table, Test Case 3, relative to the example scenario just provided, is presented in Table 5-3. By analysing the table of the test case and Figures 5-10 to 5-14, we can conclude that the implementation of this scenario was a *Success* and the objective of this step was accomplished. In the table, to what is referred as Sensor 2, it is meant to be relative to the *AAMM_Temp2* sensor.

Table 5-3 - Test Case 3.

Test Case 3		
Test Case:	Use of Semantic Maps	
Group:	Room 2 Tests	
Purpose:	Check if the program handles the manipulation of sensor states and a different use of the semantic maps	
Behaviour	Constraints	Verdict
! Press Show Maps ? Semantic Maps Loaded ! Change Sensor 2 State ? Sensor 2 State Changed ! Change Sensor 2 Output ? Sensor 2 Output Changed ? Failure Detected ! Choose Semantic Map ? Network Recovery OTHERWISE OTHERWISE OTHERWISE OTHERWISE OTHERWISE		Success Failure Failure Failure Failure Failure

Room 3: Theoretical Scenario for Interface Testing

As can be seen in Figure 5-15, this room has nineteen thermistor temperature sensors, where only one is active, *SensorTemperature5*, and is represented by the green circle. All other eighteen sensors are, initially, inactive and their location inside the room is arbitrary.

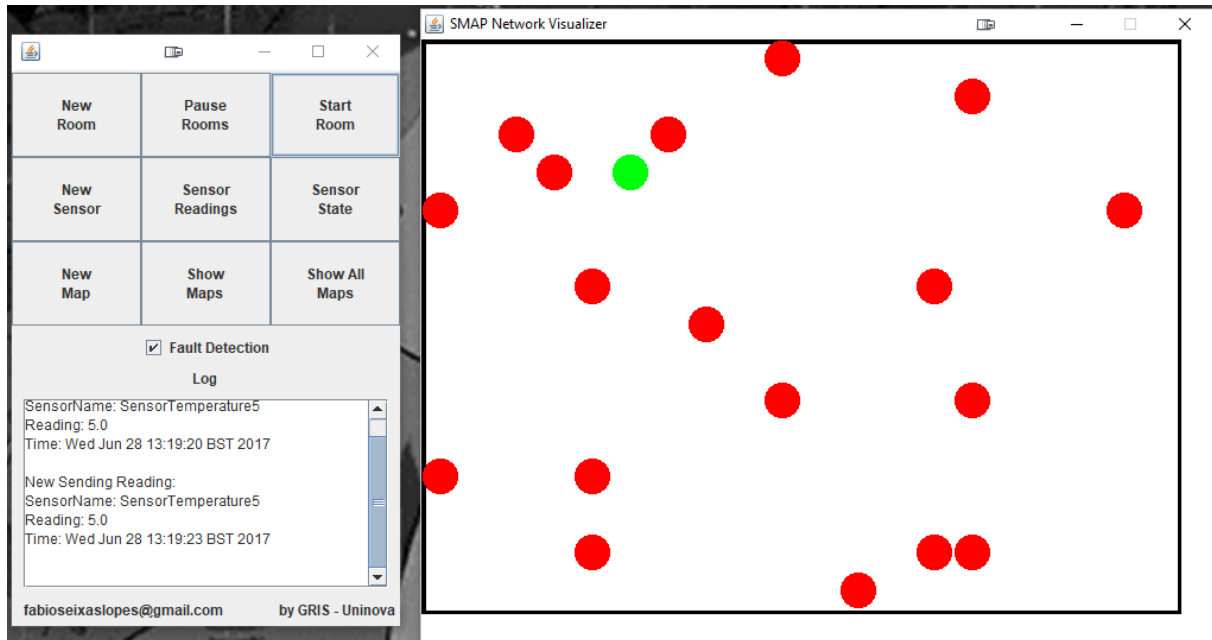


Figure 5-15 - Room 3: Initial Network State.

The active sensor has a semantic map that lists almost all other sensors as destination sensors, as can be seen in Figure 5-16. This situation shows the capability to use semantic maps from one to many sensors and it is intended to depict a situation of emergency, where all sensors are activated for the operator (or user) to get all the possible readings to figure out what is going on.

In this case, a failure in the origin sensor represents the reason for the situation of emergency. In practical terms, this scenario is also helpful to understand if the developed program and simulator are capable of functioning during a heavier process. After the use of the mentioned semantic map, another developed function is tested, the capability of pausing the room, putting all active sensors from all existent rooms on standby.

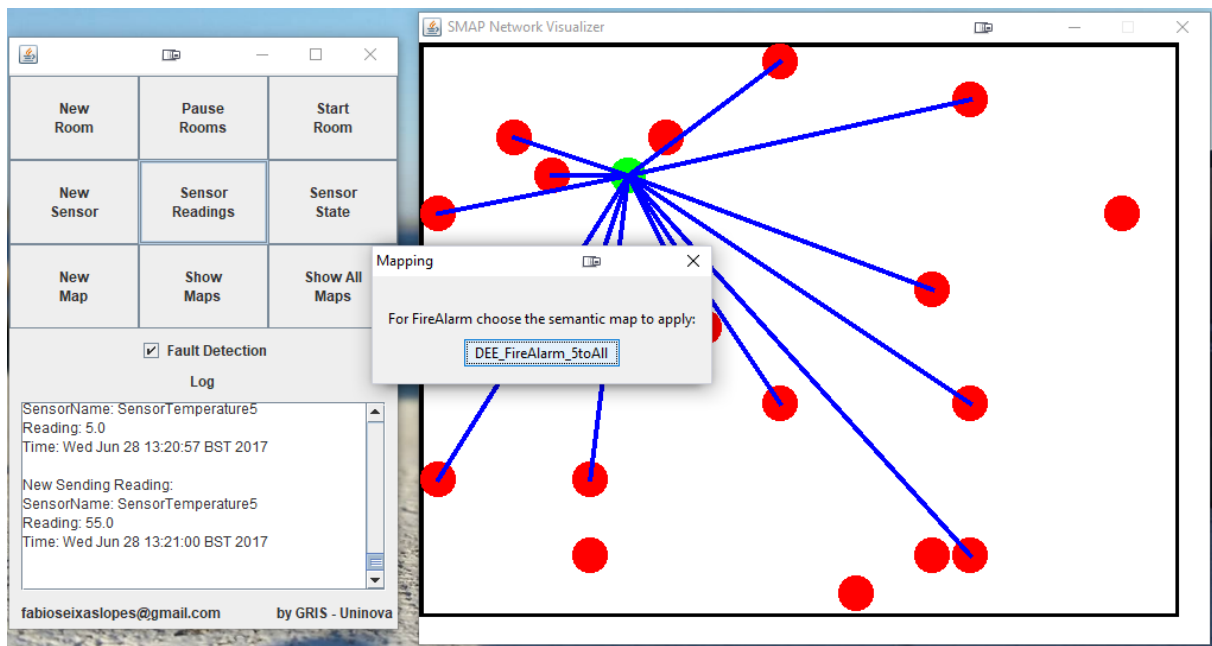


Figure 5-16 - Room 3: Semantic Map.

In Figure 5-16, in the SMAP Network Visualizer, is shown the semantic map from *SensorTemperature5* to thirteen other sensors, present in the room. In the Log there are two visible readings, in the first the reading is 5.0 units and in the second it is 55.0. This change was forced by the user to make the simulator acknowledge a failure and start the process for the use of the desired semantic map, called *DEE_FireAlarm_5toAll*. The semantic map is then chosen by the user and Figure 5-17 shows the resulting network state after this change.

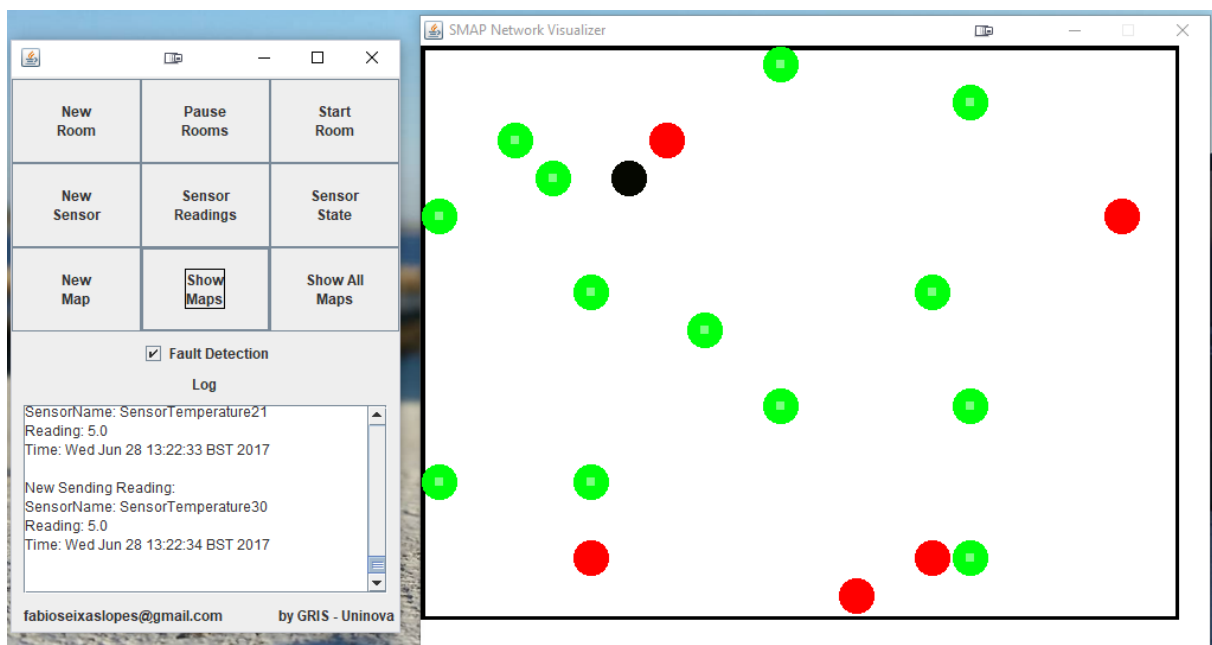


Figure 5-17 - Room 3: Network after the Semantic Map.

In the log, there are two readings, from *SensorTemperature21* and *SensorTemperature30* which are two of the thirteen sensors that are now active due to the

use of the selected semantic map. The new state of the origin sensor, *SensorTemperature5*, is *Error*. The rest of the sensors, not comprised by the semantic map, remained *Off*.

As mentioned previously, to finish the testing procedure of this scenario, the functionality of pausing all the active sensors is also tested and presented in Figure 5-18. To use this, it is only needed to press the “Pause Rooms” button and all sensors are put on standby, interrupting all readings. By clicking again in the button, all standby sensors become active again. None of the other possible states for sensors (*Off* and *Error*) are affected during this process.



Figure 5-18 - Room 3: Room Paused.

The correspondent table, Test Case 4, relative to the example scenario for Room 3, is presented in Table 5-4. By analysing the table of the test case and Figures from 5-15 to 5-18, we can conclude that the implementation of this scenario was a *Success* and the objective of this step was accomplished.

Table 5-4 - Test Case 4

Test Case 4		
Test Case:	Semantic Map from One to Many and Pausing Room	
Group:	Room 3 Tests	
Purpose:	Check if the program handles the use of a semantic map from one origin sensor to many destination sensors and if the pausing room function works properly.	
Behaviour	Constraints	Verdict
! Press Show Maps ? Semantic Maps Loaded ! Change Sensor Output ? Sensor Output Changed ? Failure Detected ! Choose Semantic Map ? Network Recovery ! Press Pause Rooms ? Sensors Are Standby ! Press Pause Rooms Again ? Sensors Are Active Again OTHERWISE OTHERWISE OTHERWISE OTHERWISE OTHERWISE OTHERWISE		Success Failure Failure Failure Failure Failure

5.3 Hypothesis Validation

In section 1.3, the hypothesis of this dissertation was defined and from there were drawn the main objectives to achieve during the development of this thesis. From the testing and validation presented in this dissertation, it can be concluded that the developed architecture successfully achieved the desired functionalities.

The developed architecture tackled the use of contextual meta-information from the network sensors with the creation of semantic maps to obtain a more reliable and intelligent network, capable of reacting to the surrounding environment constraints and variability. With

these semantic maps, the meta-information became readily available for the application of redundancy to faulty sensors, to maintain the functionalities of the network.

The resulting implementation was able to monitor the network and detect situations of interest that suggest the malfunction of devices, using the measurements that were controlled by the simulation environment. The simulator corresponded to the expectation, by maintaining a stable view of a functioning network, retrieving and showing data from the simulated active devices. This data was continuously used for the integration of architectural concepts that were meant to be tested during this dissertation and allowed their experimentation by their manipulation, considering a generic and typical real-world environment.

As it was explained previously, the core concepts of this architecture, such as the mapping equation and the structure of the knowledge bases, must be very similar in each implementation, to allow interoperability between systems and easy information exchange. By testing these concepts in a scenario, as it was done in this dissertation, it is possible to conclude the capability of functioning properly in that given scenario. But other concepts or elements of the proposed architecture can be subjective to the constraints of the environment, to allow better adaptability and reliability of the network. Examples of such elements are the types of mismatches and roles allowed. This means that the proposed architecture can be implemented to very different technological situations, due to the nature of the context in which was developed, resulting in different variations of the presented model. Which leads to conclude that the validation of the architecture can always be contested when the paradigm of the surrounding environment forces a very different variation of the implementation of the proposed model.

In terms of applicability, the proposed architecture was meant to be both suitable for integration, before and after the implementation of the network (or system). This was accomplished by creating the SMAP module, which functions independently and does not require any structural change to the typical IoT System.

5.4 Scientific Validation

A scientific paper was submitted to the IEEE International Conference on Systems, Man and Cybernetics (SMC 2017), to validate the proposed theoretical concepts of this dissertation. This paper (Lopes, Ferreira, Agostinho, & Jardim-Goncalves, 2017) was accepted and is scheduled for presentation in early October of 2017, in Banff, Canada. Below, the paper is formally mentioned:

- Lopes F., Ferreira J., Agostinho C., Jardim-Goncalves R., “Semantic Maps for IoT Network Reorganization”, Accepted in: IEEE International Conference on Systems, Man and Cybernetics (IEEE SMC 2017). October 5-8, Banff, Canada, 2017.

5.5 Industrial Validation

The industrial acceptance and validation is also an important aspect of developing scientific work, because if the industry does not approve the obtained results, this solution is most likely to never be used. By acknowledging this, the integration in practical and real-world scenarios has always been one of main focuses.

With the integration of this thesis in the C2NET project, it was possible to get into that paradigm and develop according to the specifications of an industrial research project with a real-world environment. As mentioned in the Testing Implementation, sub-chapter 5.2, the developed work had a direct impact during its development in the implementation of the C2NET project, namely in the “*António Abreu Metalomecânica, Lda*” factory.

5.5.1 SMAP in C2NET

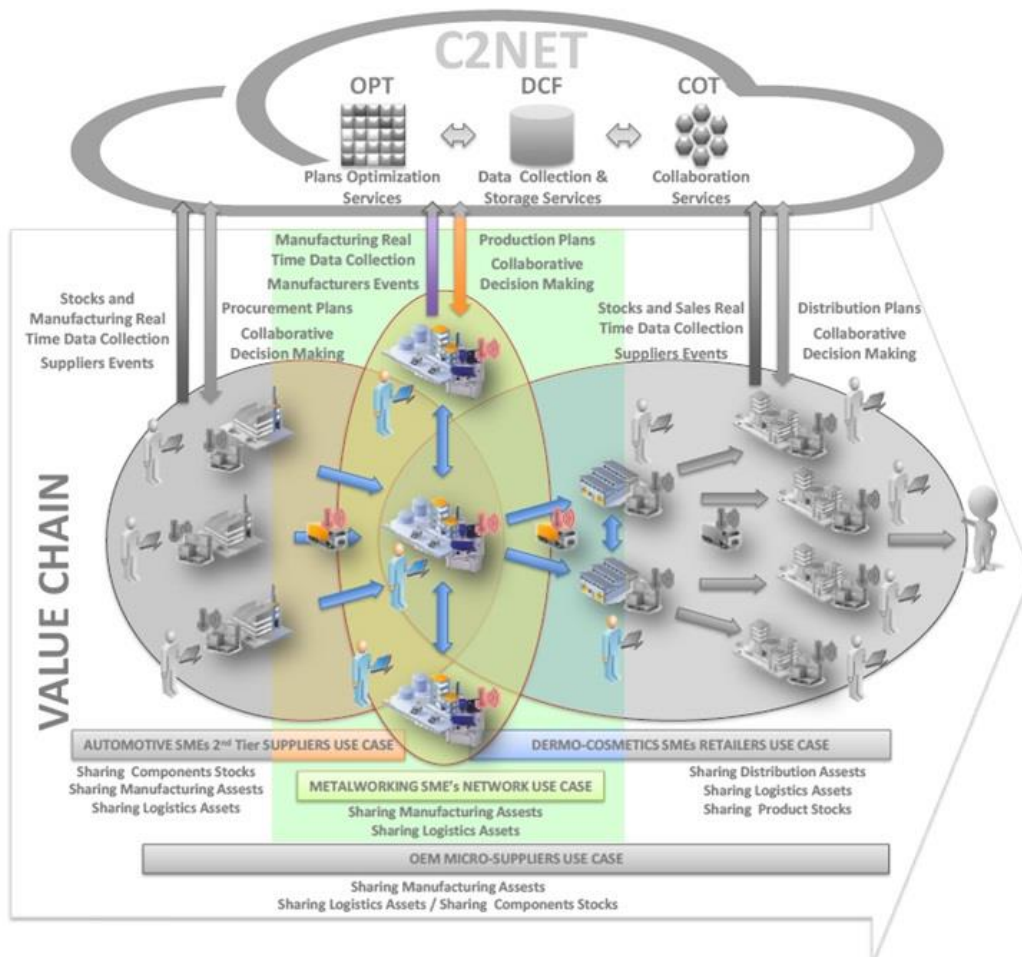


Figure 5-19: C2NET Project Overview.

This large research project involves 20 partners across the European Union and it was founded by the European Commission on the 1st of January of 2015, with a 3 years grant. The goal of the project is the creation of cloud-enabled tools for supporting the SME's supply

network optimization of manufacturing and logistic assets based on real time collaborative demand, production and delivery plans (C2NET, 2015), as can be seen in Figure 5-19.

As is expected from this type of projects, many discussions take place with the industrial partners, recurring to both physical and teleconference meetings, involving them in the definition of the concepts proposed. During this dissertation, the main focus has been to provide industrial pilots with data collection functionalities and virtualized resource management. The metalworking SME's case where "*António Abreu Metalomecânica, Lda*", highlighted in Figure 5-19, is integrated, is a real test environment for the developments described in the implementation and testing scenarios of this thesis, specifically the network monitoring stage, where events and redundancies are triggered, based on processed data from the sensors.

The SMAP module enables any company to implement IoT without any significant structural change and enables an additional set of IoT functions with the intend of improving its processes. These functions are related with the monitoring of data within spaces, or rooms, the detection of situations of interest and the application of similar network configurations, based on the semantic maps, to allow redundancy and recovery from device failures. The use of semantic maps enables more trustworthy systems and a landscape of automatic solutions to solve its most common problems autonomously. The creation of the architectural ontologies and the creation of a common tuple for the information present in each semantic map, creates a standard for its use and is prone to these automatic processes.

5.5.2 IoT Network Configuration in the Metalworking Case

As it was mentioned in the beginning of this sub-chapter, the implementation of the architecture and concepts proposed by this dissertation, needs to be complemented with work that is not directly related to what is the objective of this thesis, due to the nature of the project itself. Nevertheless, despite not taking part in the previous sections of implementation and testing, that work was necessary for real-world validation. Considering this, a brief explanation is presented here to allow a better understanding of how the concepts of this dissertation can be implemented in a IoT environment, without getting into too much detail.

This work consisted in developing communication patterns and the structure for messages to be exchanged within the network of the factory mentioned previously. The communication structure in this work is assured via CAN bus and/or Ethernet connection (using UDP). The participants in this process are the nodes (sensors and actuators) and the hub, developed and validated in the work of (Costa, 2016), responsible for the management of the network. Each node is constituted by an Arduino² device (with an Ethernet shield for Ethernet connection or CAN bus wiring for communication) and may have multiple sensors

² <https://www.arduino.cc/>

attached to it. The hub is constituted by a single Raspberry Pi³ device. A illustration of the type of hardware implemented in the AAMM factory is provided by Figure 5-20.



Figure 5-20 - Hardware for the C2NET implementation in AAMM.

The sensors send data over the network, from the node to the hub, where the data is collected for further processing. The hub is responsible for actuating on the devices, enabling or disabling them, and is responsible for the mechanisms to detect new devices, or nodes, automatically. One of the implementations can be seen in Figure 5-21, where the data cables used for communication are visible as well as a sticker on a cable denoting the connection to the hub (as “PI”).



Figure 5-21 - C2NET Implementation in the AAMM factory.

³ <https://www.raspberrypi.org/>

The structure of the messages is constituted by a maximum of 8 bytes, where each one of them contains specific data. The first byte is always used for the message type, which defines the use of the remaining bytes. Typically, the second byte denotes the node ID, the third defines the sensor ID (within each node), the fourth denotes the type of sensor (useful for messages from the node to the hub, where the hub acknowledges the existence of that sensor and what type of data it measures) and the remaining bytes are usually for values of measurements, thresholds and frequencies (for communication to happen). Those values may be very high and need to be represented by more than two bytes.

An example of this structure is presented in Figure 5-22. It is a message intended for the actuation of a warning light. The actuator is located in node 3 and has “10” as value for its ID (inside the respective node). To activate this warning light, the value 1 should reach the actuator. So, the first byte has the pre-defined message type code “0x17” that denotes the message for actuation, “0x03” is the node ID and “0x10” is the actuator ID, and occupy the bytes 2 and 4, respectively. This type of actuator has a pre-determined value for its designation, “0x30”, and that information is inserted into the third byte of the message. The value for actuation purposes occupies the fifth byte and to activate this warning light “0x01” is written. The three remaining bytes are not used for this type of message.

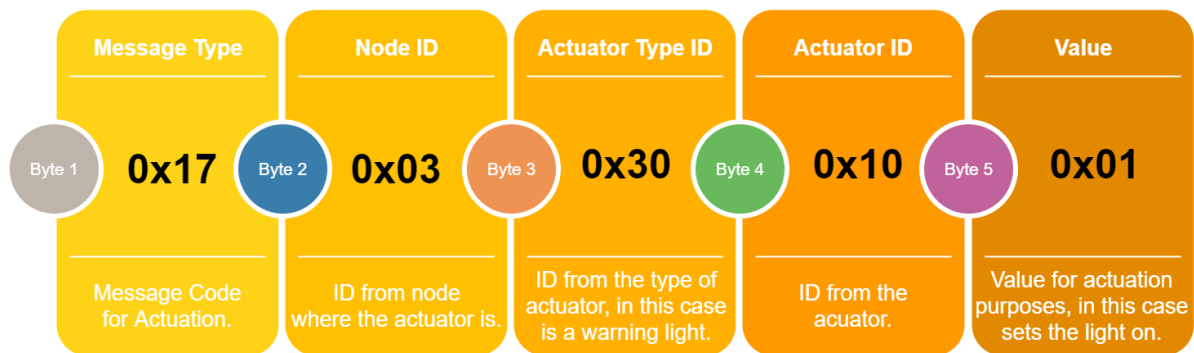


Figure 5-22 - C2NET AAMM's Communication Message Example.

This structure of communication allows the application of the IoT paradigm and the concepts proposed in this dissertation. With the structure of messages here described, the network management and processes related to the use of semantic maps, are achieved afterwards. This description was very brief but serves to elucidate the reader on how the basis of an IoT network can be build using simple programmable components as the Arduino and Raspberry Pi devices.

6 Final Considerations and Future Work

This section is the final appreciation of the author regarding the developed project and the writing of this dissertation. A comparison to the hypothesis and initial approach is here presented, to understand the effectiveness of this scientific work and to situate it in means of contemporary importance and possible further implementation. Some keynotes about possible future work to be made in the module of this project or even in the subject of this dissertation are provided, because “science is never finished” (Hermanns, 1983).

6.1 Main Results

The contribution of this thesis to the C2NET project is based on the demand of IoT solutions for autonomous adaptation and reliable functioning. These solutions, are moulded by the context of the implementation environment and are intended to be resistant to the surrounding constraints and variability. To function properly, under these circumstances, the IoT systems need to dynamically adapt and transform, using viable mechanisms. This is where the proposed semantic maps and the related concepts and processes come into play.

Focusing on the specific contribution of this thesis, to the mentioned project, the SME’s manufacturing networks visualization and processing, in terms of collecting important information, is a very important aspect that allows optimization, as intended by the project. In this case, the envisioned optimization is the application of redundancy, using the semantic maps, to ensure that the manufacturing processes are more resistant to failure in the sensing devices, responsible for triggering important manufacturing events. With this, situations like emergencies and machine failures, are easily detected and handled autonomously, when that is possible.

The original elements of this dissertation are:

- Semantic Maps Concept applied in IoT;
- Semantic Mapping Equation;
- SMAP Architecture and Software prototype;
- SMAP Ontology, Methodology and Processes;
- Fault Detection Methods, using the SMAP Ontology;
- Simulator and respective Graphical User Interface.

Additional information provided in sections 5.2 and 5.5, shows how this dissertation is used to contribute to the C2NET project activities and how it can be implemented in various IoT domains that share similar environments.

6.2 Conclusions

In this work, a suggested methodology for using semantic maps, recurring to network metadata and function-oriented recovery methodology, to accomplish autonomous error recovery and network reorganization, is presented. The objective of this solution is to improve IoT network sustainability, reliability and trustworthiness, regarding IoT devices, without compromising performance or forcing significant structural changes to systems already implemented. These were the main aspects considered during the design of the SMAP module and its interactions with the rest of the system.

To contextualize the environment of this work, the IoT paradigm, chapter 2 presents the researched information that served as basis for the development of the proposed architecture. Not only the environment was deemed as important, but also existing implemented models were studied and compared, allowing to understand what is considered relevant when creating an IoT network, or system, based on semantic models.

The studied models were the IoT-A, the W3C SSN Ontology and the IoT Lite. The IoT-A presents a very complete approach but tends to be difficult to apply due to existing various aspects and constraints that the developer must deal with, before even started implementing. Besides that, it can be seen as a model that describes the guidelines for more specific ontologies. The other two models have a more specific approach and are incisive on the problem they want to solve. This author believes that by developing ontologies for more specific solutions, but regarding models like IoT-A as basis to ensure proper interoperability within the IoT environment, creates a bigger opportunity for the scientific community to excel in the field they wish to improve, instead of dissolving into a generic solution. Nevertheless, as mentioned before, the semantic technologies are widely claimed to be a qualitatively stronger approach to interoperability than contemporary standards-based approaches (Lassila, 2005).

Following this thematic, in the same chapter, crucial aspects and concepts, related to making the network more dynamic, autonomous and intelligent were mentioned and explored, always having the semantic maps as end goal.

In the following chapter, the architecture for the semantic mapping module was presented and explained, considering each sub-module, element, specification and concept, that allowed its design and implementation. Here, a semantic mapping tuple, and the use of the resulting map, were described to show the viability of this solution and explain the processes that are associated with it. In an early phase of this design process, some event processing aspects were considered as part of the module (such as rule comparison and associated storage information), due to necessity of such aspects during the network

processing. But after reviewing and understanding all the elements of the typical IoT network and designing the module to be adaptive to an already implemented system, those aspects were dismissed, leaving them to certain network elements (e.g. CEP) in order to construct an independent solution and to accomplish the goals, of this work, in an objective manner.

In the Proof of Concept Implementation chapter, the requirements and functionalities of the general architecture were listed, along with the technological specifications and the implementation steps that allowed the validation explained in the following chapter. These steps were the culmination of all the research and model design of this thesis, and the end result was the graphical user interface, which allows the testing of the semantic maps.

Chapter 5 consisted in the presentation of the testing methodology and implementation. The implementation focused on C2NET project scenarios that were intended to mimic the IoT smart rooms from the contemporary industry and the performed tests were based on functional behaviors of the network and monitoring operator.

During the testing, it was possible to see the application of the fault detection mechanisms, developed in this dissertation. Comparing to the Esper software, initially responsible for this part of the semantic mapping process, it makes sense to use the developed fault detection ontologies when implementing within semantic models like the one of this project. Although the Esper software is proven to be a robust solution and the fact that, during the testing process, detected SOI more rapidly, it does not allow the direct correspondence to the sensors like the developed solution, which allows a much easier and automated manipulation of the conditions to detect SOI. The major disadvantage of using the created fault detection ontology is the inclusion in the architectural design, making it a heavier and more difficult configuration process.

An important implementation and testing aspect is the fact that the main tests were all carried out in a simulator, that emulated virtual sensors, instead of using real physical sensors. This occurred because the implementation of physical sensors was not the focus of this dissertation. The use of virtual sensors provided a better manipulation over them, allowing a better testing process. The errors and failures, induced into them, were based on real-world situations and the sensors represented individual and independent threads. With this in mind, the author believes that the application of the semantic mapping process to physical sensors does not bring any advantage over the developed solution because the module only focuses on SOI events. Nevertheless, as explained in section 5.5, the implementation on physical devices is an undergoing project, regarding the C2NET project activities.

In summary, the validation of the objectives of this work, supported by the implementation of the architecture, shows the capability of the network, when using the semantic mapping methodology, to detect common errors, trigger an error recovery process, analyzing the redundancies provided by the existing semantic maps, and to reorganize the network, to return it to a similar working state, as it was before the error occurred.

With all things considered, it is safe to assume that the results were positive, and the use of semantic maps is, indeed, useful to accomplish the mentioned objectives and allow the creation of better networks in the IoT environment.

6.3 Future Work

In future developments of this work, it would be interesting to improve the fault detection of sensor malfunctions, a process that by itself can lead to a significant degree of complexity, enhancing the possible uses of the SMAP module regarding the detection of SOI and providing more stability to the typical network. This degree of complexity would be based on pattern recognition of typical events, in a *big data* paradigm, and the use of mathematical models to extensively study each case.

Another aspect that deserves attention is the improvement of the process that leads to the autonomous creation of the semantic maps, creating possible redundancies with different types of sensors and measurements, to provide a better response to non-typical sensor malfunctions. Currently, this work presents a mostly human-based procedure, while creating the maps, to ensure proper development in an early stage. Nevertheless, the creation of core semantic maps, while implementing the process described in this work, is important to be kept manual and the autonomous procedures to be held during the runtime process.

- Agostinho, C., Malo, P., Jardim-Goncalves, R., & Steiger-Garcia, A. (2007). Harmonising Technologies in Conceptual Models Representation. *International Journal of Product Lifecycle Management (IJPLM)*, 2(2).
- Agostinho, C., Sarraipa, J., Goncalves, D., & Jardim-Goncalves, R. (2011). Tuple-Based Semantic and Structural Mapping for a Sustainable Interoperability. *Technological Innovation for Sustainability. DoCEIS 2011.*, (IFIP International Federation for Information Processing).
- Akbari, A., Dana, A., Khademzadeh, A., & Beikmahdavi, N. (2011). Fault Detection and Recovery in Wireless Sensor Network Using Clustering. *International Journal of Wireless & Mobile Networks (IJWMN)*, 3(1).
- Anderson, S. (2011). Structural Testing. The University of Edinburgh - School of Informatics.
- Athreya, A., DeBruhl, B., & Tague, P. (2013). Designing for Self-Configuration and Self-Adaptation in the Internet of Things. *Proceedings of the 9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*. <https://doi.org/10.4108/icst.collaboratecom.2013.254091>
- Bauer, M., Bui, N., Carrez, F., Giacomini, P., Haller, S., Ho, E., ... Meissner, S. (2012). Introduction to the Architectural Reference Model for the Internet of Things. Retrieved from http://www.iot-a.eu/public/public-documents/copy_of_d1.2/at_download/file
- Bermudez-edo, M., Barnaghi, P., & Elsaleh, T. (2015). IoT-Lite Ontology Working Draft. Retrieved June 10, 2016, from <http://iot.ee.surrey.ac.uk/fiware/ontologies/iot-lite>
- Bermudez-edo, M., Elsaleh, T., Barnaghi, P., & Taylor, K. (2015). IoT-Lite Ontology. Retrieved June 13, 2016, from <https://www.w3.org/Submission/iot-lite/>
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American Magazine*.
- C2NET. (2015). C2NET. Retrieved August 24, 2016, from <http://c2net-project.eu/>
- Camarinha-Matos, L. M. (2012a). Scientific Method.
- Camarinha-Matos, L. M. (2012b). Thesis Organization and Validation.
- Carr, J. J., & Brown, J. M. (2000). *Introduction to Biomedical Equipment Technology, Fourth Edition*. Pearson.
- Chakraborty, S., & Eberspacher, J. (2012). *Advances in Real-Time Systems*. Springer.
- Chatzigiannakis, I., Hasemann, H., Karnstedt, M., Kleine, O., Kröller, A., Leggieri, M., ... Truong, C. (2012). True self-configuration for the IoT. *Proceedings of 2012 International Conference on the Internet of Things, IOT 2012*.

<https://doi.org/10.1109/IOT.2012.6402298>

Chinneck, J. W. (1999). How to Organize your Thesis. Retrieved August 25, 2016, from <http://www.sce.carleton.ca/faculty/chinneck/thesis.html>

Cisco. (2008). Cisco Context Aware Mobility Solution.

Compton, M., Barnaghi, P., & Bermudez, L. (2011). The SSN Ontology of the Semantic Sensor Networks Incubator Group. *Journal of Web ...*, 1–6. Retrieved from http://www.w3.org/2005/Incubator/ssn/wiki/images/f/f3/SSN-XG_SensorOntology.pdf

Correia, F. (2010). Model Morphisms (MoMo) to Enable Language Independent Information Models and Interoperable Business Networks. *Technology*, (September). Retrieved from <http://dspace.fct.unl.pt/handle/10362/4782?mode=full>

Costa, P. (2016). *IoT for Efficient Data Collection from Real World*. FCT-UNL.

Ding, Y., & Fensel, D. (2001). Ontology Library Systems : The key to successful Ontology Re-use.

EsperTech. (2016). About Esper. Retrieved June 1, 2017, from <http://www.espertech.com/esper/>

Estes, T. H. (1999). Semantic Maps. Retrieved March 1, 2017, from http://www.readingquest.org/edis771/semantic_maps.html

Ferreira, J. (2012). Monitoring morphisms to support sustainable interoperability of enterprise systems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7046 LNCS. https://doi.org/10.1007/978-3-642-25126-9_15

Ferreira, J., Agostinho, C., Sarraipa, J., & Jardim-Goncalves, R. (2012). Monitoring morphisms to support sustainability of interoperability in the manufacturing domain. In *IFAC Proceedings Volumes (IFAC-PapersOnline)* (Vol. 14). <https://doi.org/10.3182/20120523-3-RO-2023.00221>

Garlan, D., Schmerl, B., & Cheng, S.-W. (2009). Software Architecture-Based Self-Adaptation. *Autonomic Computing and Networking*. <https://doi.org/10.1007/978-0-387-89828-5>

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubezy, M., Eriksson, H., ... Tu, S. W. (2003). The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human Computer Studies*, 58(1), 89–123. [https://doi.org/10.1016/S1071-5819\(02\)00127-1](https://doi.org/10.1016/S1071-5819(02)00127-1)

Guardalben, L., Villalba, L. J. G., Buiati, F., Sobral, J. B. M., & Camponogara, E. (2011). Self-configuration and self-optimization process in heterogeneous wireless networks. *Sensors*, 11(1). <https://doi.org/10.3390/s110100425>

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645–1660. <https://doi.org/10.1016/j.future.2013.01.010>

Heflin, J. (2006). Introduction to the OWL Web Ontology Language.

- Hermanns, W. (1983). *Einstein and the Poet*.
- Hoßbach, B., & Seeger, B. (2013). Anomaly management using complex event processing: extending data base technology. *Proceedings of the 16th International Conference on Extending Database Technology*. <https://doi.org/10.1145/2452376.2452395>
- Infocomm Development Authority. (2012). The Internet of Things (IOT), (July 2012). Retrieved from <https://www.ida.gov.sg>
- Intersog. (2016). Internet of Things. Retrieved August 20, 2016, from <http://iot.intersog.com/>
- Jacob Morgan. (2014). A Simple Explanation Of IoT. Retrieved June 1, 2016, from <http://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#4bf45b046828>
- Kastelein, R. (2012). The Internet of Things and Change – Will You Be Ready For The M2M World? Retrieved August 25, 2016, from <http://www.richardkastelein.com/the-internet-of-things-and-change-will-you-be-ready-for-the-m2m-world/>
- Katasonov, A., Kaykova, O., Khriyenko, O., Nikitin, S., & Terziyan, V. (2006). Smart Semantic Middleware for the Internet of Things.
- Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *Computer*, 36(1). <https://doi.org/10.1109/MC.2003.1160055>
- Kim, C. (2009). Scalable and Efficient Self-Configuring Networks.
- Knoblock, C. A., & Szekely, P. (2013). Semantics for big data integration and analysis. *2013 AAAI Fall Symposium, FS-13-04*. Retrieved from <http://www.scopus.com/inward/record.url?eid=2-s2.0-84898846943&partnerID=40&md5=121e9677da1feb31022a1fcf74fcb209>
- Knublauch, H., Fergerson, R. W., Noy, N. F., & Musen, M. a. (2004). The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. *Iswc*, (3298), 229–243. <https://doi.org/10.1.1.89.9268>
- Kolozali, S., Barnaghi, P., & Bermudez, M. (2016). No Title. Retrieved August 25, 2016, from <http://iot.ee.surrey.ac.uk/citypulse/ontologies/sao/sao#>
- Konstantinou, A. V, Florissi, D., & Yemini, Y. (2002). Towards Self-Configuring Networks.
- Lassila, O. (2005). Using the Semantic Web in Mobile and Ubiquitous Computing.
- Lopes, F., Ferreira, J., Agostinho, C., & Jardim-Goncalves, R. (2017). *Semantic Maps for IoT Network Reorganization* (IEEE International Conference on Systems, Man and Cybernetics (IEEE SMC 2017)). Banff, Canada.
- Margara, A., Cugola, G., & Tamburrelli, G. (2014). Learning from the past: automated rule generation for complex event processing. *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems - DEBS '14*, 47–58. <https://doi.org/10.1145/2611286.2611289>
- Merriam-Webster. (2017). Definition of Semantics. Retrieved March 1, 2017, from <https://www.merriam-webster.com/dictionary/semantics>

- Muller, G. (2008). A Reference Architecture Primer, (February).
- Munir, S., & Stankovic, J. A. (2015). FailureSense: Detecting sensor failure using electrical appliances in the home. *Proceedings - 11th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2014*, 73–81. <https://doi.org/10.1109/MASS.2014.16>
- Myers, G. J., Thomas, T. M., & Sandler, C. (2004). *The Art of Software Testing*. *Booksgooglecom* (Vol. 1). <https://doi.org/10.1002/stvr.321>
- Nordgren. (2004). The Scientific Method.
- Noy, N. F., Fergerson, R. W., Musen, M. a, & Informatics, S. M. (2000). The knowledge model of Protégé-2000: combining interoperability and flexibility. *Proceedings of the 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00)*, (1), 17–32. https://doi.org/10.1007/3-540-39967-4_2
- Oberoi, S. (2011). Esper Complex Event Processing Engine. Retrieved from <http://esper.codehaus.org/>
- Paiva, L. M. S. S. (2015). *Semantic relations extraction from unstructured information for domain ontologies enrichment*. FCT-UNL. Retrieved from <http://run.unl.pt/handle/10362/16550>
- Pathan, M., Taylor, K., & Compton, M. (2010). Semantics-based Plug-and-Play Configuration of Sensor Network Services.
- Perera, C., Jayaraman, P. P., & Christen, P. (2013). Sensor Discovery and Configuration Framework for The Internet of Things Paradigm.
- Perera, C., Zaslavsky, A., Compton, M., Christen, P., & Georgakopoulos, D. (2013a). Context aware sensor configuration model for internet of things. *CEUR Workshop Proceedings*, 1035.
- Perera, C., Zaslavsky, A., Compton, M., Christen, P., & Georgakopoulos, D. (2013b). Semantic-driven configuration of internet of things middleware. *Proceedings - 2013 9th International Conference on Semantics, Knowledge and Grids, SKG 2013*. <https://doi.org/10.1109/SKG.2013.9>
- Prud'hommeaux, E., & Seaborne, A. (2008). SPARQL Query Language for RDF. Retrieved June 1, 2017, from <https://www.w3.org/TR/rdf-sparql-query/>
- Rannenber, K., Royer, D., & Deuker, A. (2009). *The Future of Identity in the Information Society - Challenges and Opportunities*.
- Rivera, J., & Van der Muelen, R. (2013). Gartner Says the Internet of Things Installed Base Will Grow to 26 Billion Units By 2020. *Gartner*, 26–28. Retrieved from <http://www.gartner.com/newsroom/id/2636073>
- Rose Jaren, Eldridge Scott, C. L. (2015). The internet of things: an overview - Understanding the issues and challenges of a more connected world. *The Internet Society (ISOC)*, (October).
- Seeger, B. (2012). Dynamic Complex Event Processing. Retrieved from https://db.in.tum.de/hosted/scalableanalytics/presentations/CEP_Dynamic.pdf

- Segaran, T., Evans, C., & Taylor, J. (2009). *Programming the Semantic Web*.
- Sharma, A. B., Golubchik, L., & Govindan, R. (2010). Sensor Faults : Detection Methods and Prevalence in Real-World Datasets. *ACM Transactions on Sensor Networks*, 6(3).
<https://doi.org/10.1145/1754414.1754419>
- Sniderman, B., Monika, M., & Cotteleer, M. J. (2016). Industry 4.0 and manufacturing ecosystems. *Deloitte University Press*.
- Stevens, R. (2001). What is an Ontology? Retrieved from
<http://www.cs.man.ac.uk/~stevensr/onto/node3.html>
- Theunis, J., Stevens, M., & Botteldooren, D. (2017). Participatory Sensing, Opinions and Collective Awareness. *Participatory Sensing, Opinions and Collective Awareness*, 21–46.
<https://doi.org/10.1007/978-3-319-25658-0>
- Tretmans, J. (1999). Testing Concurrent Systems: A Formal Approach. In *Proceedings of the 10th International Conference on Concurrency Theory (CONCUR)* (Vol. 1664).
https://doi.org/10.1007/3-540-48320-9_6
- Tretmans, J. (2001). An Overview of OSI Conformance Testing. *Methods*, 1–14.
- TTCN-3. (2013). TTCN Free Tutorials. Retrieved June 1, 2017, from <http://www.ttcn-3.org/index.php/learn/tutorials>
- Unis, M., Nettsträter, A., Iml, F., Stefa, J., Suni, C. S. D., Salinas, A., & Sapienza, U. (2013). Internet of Things – Architecture IoT - A Final architectural reference model for the IoT v3, (257521).
- Vermesan, O., Friess, P., Guillemin, P., Gusmeroli, S., Sundmaeker, H., Bassi, A., ... Pat, D. (2009). Internet of Things Strategic Research Roadmap. *Internet of Things Strategic Research Roadmap*, 9–52. https://doi.org/http://internet-of-things-research.eu/pdf/IoT_Cluster_Strategic_Research_Agenda_2011.pdf
- W3C. (2012). SSN Applications. Retrieved September 1, 2016, from
https://www.w3.org/community/ssn-cg/wiki/SSN_Applications
- Weiss, G., Zeller, M., & Eilers, D. (2010). Towards Automotive Embedded Systems with Self-X Properties.
- White, L. J. (1987). Software Testing and Verification. *Advances in Computers*, 26(C), 335–391.
[https://doi.org/10.1016/S0065-2458\(08\)60010-8](https://doi.org/10.1016/S0065-2458(08)60010-8)
- Woods, L., Teubner, J., & Alonso, G. (2010). Complex Event Detection at wire speed with FPGA's.
- Yee, M. J., Philips, S., Condon, G. R., Jones, P. B., Kao, E. K., Smith, S. T., ... Waugh, F. R. (2013). Network Discovery with Multi-intelligence Sources, 20(1).
- Zeller, A. (2017). Functional Testing. Saarland University.