SEMIoTICS

SEmantic **M**odel-driven development for
IoT Interoperability of emergen**Cy** **S**ervices

Improving the Semantic Interoperability of
IoT Early Warning Systems

João Luiz Rebelo Moreira

SEmantic **M**odel-driven development for **IoT**
Interoperability of emergen**Cy** **S**ervices

Improving the Semantic Interoperability of
IoT Early Warning Systems

João Luiz Rebelo Moreira

SEMIOTICS

SEMANTIC MODEL-DRIVEN DEVELOPMENT
FOR IOT INTEROPERABILITY OF
EMERGENCY SERVICES

IMPROVING THE SEMANTIC
INTEROPERABILITY OF IOT EARLY
WARNING SYSTEMS

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof.dr. T.T.M. Palstra,
on account of the decision of the Doctorate Board,
to be publicly defended
on Wednesday the 3rd of July, 2019 at 12.45.

by

João Luiz Rebelo Moreira

born on the 9th of July, 1980
in Rio de Janeiro, Brazil

This dissertation has been approved by:

Supervisors:

Dr. L. Ferreira Pires

Dr. Ir. M. J. van Sinderen

Cover design: João Luiz Rebelo Moreira and Tiago Mafra Guidicini

Printed by Ipskamp, Enschede, the Netherlands

ISBN: 978-94-028-1587-0

DOI: 10.3990/1.9789402815870

© 2019: João Luiz Rebelo Moreira, the Netherlands

All rights reserved. No parts of this thesis may be reproduced, stored in a retrieval system or transmitted in any form or by any means without permission of the author.

Graduation Committee

Chairman: Prof. Dr. J.N. Kok
(Represented by Prof. Dr. A. Rensink)

Promotors: Dr. L. Ferreira Pires
Dr. Ir. M.J. van Sinderen

Members: Prof. Dr. F. Benaben (IMT Mines Albi)
Prof. Dr. M.L.M. Campos (Universidade Federal do Rio
de Janeiro)
Prof. Dr. Ir. B. Tekinerdogan (Wageningen University)
Prof. Dr. Ir. M. R. van Steen (University of Twente)
Prof. Dr. R. J. Wieringa (Univeristy of Twente)

SEMIOTICS
SEMANTIC MODEL-DRIVEN DEVELOPMENT FOR IOT
INTEROPERABILITY OF EMERGENCY SERVICES

IMPROVING THE SEMANTIC INTEROPERABILITY OF IOT EARLY
WARNING SYSTEMS

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de Universiteit Twente, op gezag van
de rector magnificus, prof.dr. T.T.M. Palstra, volgens besluit van het College
voor Promoties in het openbaar te verdedigen op woensdag 3 Juli 2019 te 12.45
uur.

door

João Luiz Rebelo Moreira
geboren op 9 Jul 1980
te Rio de Janeiro, Rio de Janeiro, Brazilië

Dit proefschrift is goedgekeurd door:
Dr. Luís Ferreira Pires (promotor)
Dr. Ir. Marten J. van Sinderen (promotor)

Abstract

Disaster Risk Reduction (DRR) is a systematic approach to analyze potential disasters and reduce their occurrence rate and possible impact. The main DRR component is an Early Warning System (EWS), which is a distributed information system that is able to monitor the physical world and issue warnings if abnormal situations occur. EWSs that use Internet-of-Things (IoT) technologies, so called IoT EWS, are suitable to realize (near) real-time data acquisition, risk detection and message brokering between data sources and information receivers, comprising both humans (e.g., emergency managers) and machines (e.g., sirens).

Over the last years, numerous IoT EWSs were developed to monitor different types of hazards. *Multi-hazard EWSs* are a special class of EWS that can detect different types of situations and, if necessary, react to them. Multi-hazard EWSs require integration of existing EWSs and seamless integration with new EWSs. Interoperability of EWS components is necessary for effective integration, e.g., so that sensors, devices and platforms work with each other and with other EWSs.

Although IoT technologies offer possibilities to improve the EWS efficiency and effectiveness, this potential can only be exploited if interoperability challenges are addressed at all levels. In this thesis, we focus on how to improve the semantic interoperability of IoT EWSs. Semantic interoperability refers to the ability of two or more EWSs (or EWS components) to share data elements in a prescribed format (syntax) and precise unambiguous meaning (semantics). From a literature review on semantic IoT EWS approaches, we selected the three major challenges that need to be addressed together:

- 1) semantic integration of a variety of data sources that make use of different standards, ontologies and data models;
- 2) near-real-time processing in time- and safety-critical applications; and
- 3) data analysis for effective situation awareness and decision support.

This thesis introduces the “SEmantic Model-driven development for IoT Interoperability of emergenCy serviceS” (SEMIoTICS) framework, which is a

holistic approach for semantic IoT EWS. SEMIoTICS consists of a semantic model-driven architecture that facilitates the application of data representations, model transformations and distributed software components.

SEMIoTICS is a framework that can be used to develop interoperable IoT EWSs for different domains, enabling an IoT EWS to act as a cloud-based semantic broker for situation-aware decision support. SEMIoTICS is leveraged by the adoption of ontology-driven conceptual modelling for situation-aware applications, covering both EWS design-time (specification and implementation) and runtime. Furthermore, the SEMIoTICS guides the application of the *Findable, Accessible, Interoperable and Reusable* (FAIR) data principles, in which the role of standardization is emphasized.

SEMIoTICS was validated in the context of the H2020 INTER-IoT project, in which a semantic interoperable IoT EWS was developed to detect accident risks with trucks that deliver goods at the Valencia port area. The research in this case study addresses the semantic integration of a variety of data sources with processing in safety-critical applications for effective emergency response. The solution considers existing domain-specific ontologies and standards, along with their serialization formats. In this case study, accident risks are assessed by monitoring two types of data, namely (1) the drivers' vital signs with electrocardiogram (ECG), and (2) the trucks' position, speed and acceleration. The case study includes the detection of health issues with drivers and collisions with vehicles with dangerous goods. A special result of this research for this case study is SAREF4health, which extends the European semantic standard for IoT (Smart Appliances REference, SAREF) with the representation of ECG data.

The framework has been validated in three ways with respect to non-functional aspects: (1) an analysis of the accuracy and efficiency of the semantic translations, (2) an analysis of the communication efficiency of JSON for Linked Data (JSON-LD) for IoT scenarios, and (3) an analysis of the scalability of the semantic brokering between data sources and information receivers.

The most important contributions of this thesis are:

- Improved IoT Semantic Interoperability: (1) semantic translations between IoT standards (W3C SSN/SOSA and ETSI SAREF) for semantic integration; and (2) SAREF4health, as the first extension of SAREF for the healthcare domain;
- Improved Situation Identification for IoT EWS: higher semantic expressiveness with a new version of the Situation Modelling Language (SML) and Complex Event Processing (CEP) technology;
- Interoperability reference for disaster services: improved reference architecture validated with an open source cloud-based IoT EWS for ECG monitoring.

Samenvatting

Vermindering van rampenrisico's (*Disaster Risk Reduction* - DRR) is een systematische aanpak voor het analyseren van mogelijke rampen teneinde de kans op het daadwerkelijk optreden van de rampen en de omvang van de schade te verminderen. De belangrijkste DRR-component is een *Early Warning System* (EWS), een gedistribueerd informatiesysteem dat in staat is om de toestand en veranderingen in de fysieke wereld te meten en waarschuwingen te geven als zich abnormale situaties voordoen. EWS's die gebruik maken van Internet-of-Things (IoT) technologie, IoT EWS's, zijn geschikt om (bijna) real-time data-acquisitie, risicodetectie en communicatie tussen gegevensbronnen en informatie-ontvangers te realiseren. Informatie-ontvangers in deze context kunnen zowel mensen (bijvoorbeeld voor het managen van noodgevallen) als machines (bijvoorbeeld voor het geven van een alarmsignaal) zijn.

In de afgelopen jaren zijn er talloze IoT EWS's ontwikkeld, voor verschillende soorten gevaren. Een speciale categorie vormen de *multi-hazard* EWS's, die verschillende soorten gevaarsituaties kunnen detecteren en zo nodig op deze situaties kunnen reageren. Multi-hazard EWS's vereisen integratie van bestaande EWS's en naadloze integratie met nieuwe EWS's. Voor effectieve integratie is interoperabiliteit van EWS-componenten noodzakelijk, zodat bijvoorbeeld sensoren, apparaten en platforms onderling en met andere EWS's kunnen samenwerken.

Hoewel IoT-technologie mogelijkheden biedt om de efficiëntie en effectiviteit van EWS's te verbeteren, kan dit potentieel alleen worden benut als interoperabiliteitsuitdagingen op alle niveaus worden aangepakt. In dit proefschrift richten we ons op het verbeteren van de semantische interoperabiliteit van IoT EWS's. Semantische interoperabiliteit duidt op het vermogen van twee of meer EWS's (of EWS-componenten) om gegevens-elementen te delen in een voorgeschreven formaat (syntaxis) en met ondubbelzinnige betekenis (semantiek).

Uit een literatuuronderzoek naar semantische IoT EWS benaderingen hebben we de drie belangrijkste uitdagingen geselecteerd die samen moeten worden aangepakt:

(1) semantische integratie van een verscheidenheid aan gegevensbronnen die gebruik maken van verschillende standaarden, ontologieën en datamodellen;

(2) tijdige verwerking in applicaties met strenge eisen aan reactiesnelheid en veiligheid; en

(3) data-analyse voor het herkennen van gevaarsituaties en de ondersteuning van reactiebeslissingen.

Dit proefschrift introduceert het *SEmantic Model-driven development for IoT Interoperability of emergenCy services* (SEMIoTICS) raamwerk, een holistische benadering is voor semantische IoT EWS's. SEMIoTICS omvat een semantische model-gedreven architectuur die de toepassing van gegevensrepresentaties, modeltransformaties en gedistribueerde software componenten vergemakkelijkt.

SEMIoTICS is een raamwerk dat kan worden gebruikt om interoperabele IoT EWS's voor verschillende domeinen te ontwikkelen, waardoor een IoT EWS kan optreden als een cloudgebaseerde semantische makelaar voor situatiebewuste besluitvormingsondersteuning. SEMIoTICS maakt gebruik van een ontologiegedreven conceptuele modellering voor situatiebewuste toepassingen, zowel tijdens de ontwerpfase (specificatie en implementatie) van een EWS als tijdens de operationele fase. Bovendien begeleidt SEMIoTICS de toepassing van de *Findable, Accessible, Interoperable and Reusable* (FAIR) principes voor gegevensmanagement, waarbij de nadruk wordt gelegd op de rol van standaardisatie.

SEMIoTICS werd gevalideerd in de context van het H2020 *INTER-IoT*-project. In dit project werd een semantische, interoperabele IoT EWS ontwikkeld voor het herkennen van risico's op ongelukken met vrachtwagens die goederen leveren in het havengebied van Valencia. Het onderzoek voor deze casus richt zich op de semantische integratie van verschillende gegevensbronnen in veiligheidskritieke applicaties voor een effectieve reactie op noodsituaties. De oplossing houdt rekening met bestaande domeinspecifieke ontologieën en standaarden, samen met hun serialisatieformaten. In deze studie worden de risico's beoordeeld door analyse van twee soorten gegevens, namelijk (1) het elektrocardiogram (ECG) van de bestuurders en (2) de positie, snelheid en versnelling van de trucks. De studie omvat de detectie van gezondheidsproblemen bij bestuurders en van botsingen van voertuigen met gevaarlijke goederen. Een bijzonder resultaat van het onderzoek voor deze casus is *SAREF4health*, een uitbreiding van *Smart Appliances REference*

(SAREF), een Europese semantische standaard voor IoT, met de weergave van ECG-gegevens.

Het raamwerk is op drie manieren gevalideerd met betrekking tot niet-functionele aspecten: (1) een analyse van de nauwkeurigheid en efficiëntie van de semantische vertalingen, (2) een analyse van de communicatie- efficiëntie van *JSON for Linked Data* (JSON-LD) voor IoT-scenario's, (3) en een analyse van de schaalbaarheid van de communicatie tussen gegevensbronnen en informatie-ontvangers.

De belangrijkste bijdragen van dit proefschrift zijn:

- Verbeterde IoT semantische interoperabiliteit: (1) semantische vertalingen tussen IoT-standaarden (W3C SSN / SOSA en ETSI SAREF) voor semantische integratie; en (2) SAREF4health als de eerste uitbreiding van SAREF voor het domein gezondheidszorg;
- Verbeterde situatie-identificatie voor IoT EWS: hogere semantische expressiviteit met een aangepaste versie van de *Situation Modeling Language* (SML) en *Complex Event Processing* (CEP) technologie;
- Interoperabiliteitsreferentie voor nooddiensten: verbeterde referentiearchitectuur gevalideerd door een *open source* cloud-gebaseerde IoT EWS voor ECG monitoring.

Acknowledgements

“Gratitude can transform common days into thanksgivings, turn routine jobs into joy, and change ordinary opportunities into blessings”

William Arthur Ward

Completing a PhD is challenging, it requires passion for the research, discipline, persistence and emotional endurance. In total, there were 17 conference/workshop participations; a complete research collaboration (INTER-IoT), from the proposal to results; a high rated proposal for a H2020 call, which included 13 institutions; several publications, presentations and lectures. Personal issues, concerns and some losses in the family; and, last but not least, having a child. Everything within 4 years. The saying “behind every successful man there is a strong woman” never fit so well as for this situation. I wouldn’t be able to handle all of this without the full support of my wife, whom I will forever be grateful for. Therefore, a special thanks to my dear wife Bel, for all friendship and comprehension in those tough years. I love you.

I would like to express my gratitude to my supervisors Luís and Marten. I appreciate all the guidance, patience and motivation throughout this journey, including the unconditional support and freedom to explore! You have been of great importance to my academic knowledge and career. Thanks for the friendship and assistance for all personal issues. It’s amazing how Luís and Marten are complementary in several aspects and, in some extent, exchanged some “cultural behaviors” during this long period that they are together: while Luís has a more “Dutch-methodic” profile, Marten looks like the typical “carioca”: relaxed and easygoing. A special thanks for Luís, I will never forget that day we met in 2013 during a studying abroad fair in UFRJ, when coincidentally I went to him to ask if he was aware of Giancarlo’s thesis. I wouldn’t be here in Twente if it wasn’t you.

I would like to thank the members of my defense committee: Prof. Dr. Frederick Benaben, Prof. Dr. Ir. Bedir Tekinerdogan, Prof. Dr. Ir. Maarten van Steen, Prof. Dr. Roel Wieringa and Prof. Dr. Maria Luiza M. Campos. It is an honor to have you in this committee. In particular Roel, for following closely my

PhD trajectory and for his precious feedbacks; and Maria Luiza, my "academic mother" and guru, for all the support that she has been giving to me in these more than ten years together.

I would like to thank my colleagues at the Services and CyberSecurity (SCS) group for the fruitful cooperation and for welcoming me as part of the team. In particular, I would like to express a special gratitude to Suse, who acted as a mother to me, helping us with any issues. She has become a close friend to my family (another "oma" of Teo), to whom we have much affection and respect. Thanks also to Geert Jan for the full support on any laboratory needs, and for Bertine, who also helped whenever I needed.

Thanks to the NEMO (João Paulo, Giancarlo, Renata, Veruska, John, Freddy, Bernardo, among others) and LPRM (Patricia, Zé, among others) groups in UFES, and the GRECO group (Marcos Borges, Jonice, among others). A special thanks to Patricia Dockhorn for being a kind of "long-distance supervisor", helping me since the PhD proposal to the result analysis of this research. Thanks also to colleagues from other institutions that we collaborated during this period, such as Tiago, Roberta and Nicola (LOA); Katarzyna and Pawel (SRI-PAS); Pablo and Miguel (Valencia port); Carlos (UPV); Laura (TNO); Norman (ITC) and many others that interacted with me during conferences and project proposals. I thank the Brazilian national agency CAPES for financial support (BEX 1046-14-4) and the European Commission for the collaboration within the INTER-IoT project.

Thanks to all "family in Enschede", such as the "Brazilian office" in 2015 (Zé Gonçalves, Robson, Carlos, Glaucia and Rodolfo); the "Enschede citizens", including (but not limited to) Luiz Olavo, Luciana, Jair, Priscila, Ivan, Bia, Henrique, Eveline, Lidiane, Peter, Diego, Ju, Filipe, Steph, Anderson, Milena, André, Natália, Maurício, Thais, Felipe, Aline, Liniker, Suelen, Philip, Johanna, Eduardo, Nayeli, among many other friends. You have spared no efforts to make life so "gezellig" here! A special thanks for Luiz Olavo for all matters involving the GO-FAIR initiative and related research; for "oma" Patricia and "opa" Jaury for all assistance with both academic and personal life, acting as a psychologist for the difficult moments. Thanks for being such great persons in which we can place all our confidence.

The support from my family (including Kitty, Luli and relatives) and dear friends in Brazil was also fundamental for this achievement, especially from my parents and my dear "tia", who always were there for me; as well as friends from ONS who encouraged me for this adventure. Special thanks to my parents José and Lucia, for providing me all the necessary education to achieve this title and the unconditional love even when I was absent.

This book is in memory of my "tia" and dedicated to my parents José and Lucia, to my wife Bel and my son Teo. None of this could be done without you.

Agradecimentos

“A gratidão pode transformar dias comuns em ações de graças, transformar tarefas rotineiras em alegria e transformar oportunidades comuns em bênçãos”

William Arthur Ward

Completar um PhD é desafiador, requer paixão pela pesquisa, disciplina, persistência e resistência emocional. No total, foram 17 participações em conferências / workshops; uma colaboração de pesquisa completa (INTER-IoT), da escrita da proposta aos resultados finais; uma proposta bem classificada em uma chamada do H2020, que incluía 13 instituições; várias publicações, apresentações e palestras. Questões pessoais, preocupações e algumas perdas na família; e, por último mas não menos importante, ter um filho. Tudo isso em 4 anos. O ditado “por trás de todo homem de sucesso existe uma forte mulher” nunca se encaixou tão bem quanto nessa situação. Eu não conseguiria lidar com tudo isso sem o apoio total da minha esposa, a qual ficarei grato para sempre. Portanto, um agradecimento especial à minha querida esposa Bel, por toda amizade e compreensão nesses anos difíceis. Eu te amo.

Gostaria de expressar minha gratidão aos meus orientadores Luís e Marten. Obrigado por toda orientação, paciência e motivação ao longo desta jornada, incluindo o apoio incondicional e a liberdade para explorar! Vocês têm sido de grande importância para o meu conhecimento acadêmico e para minha carreira. Obrigado pela amizade e assistência para todos os problemas pessoais. É incrível como Luís e Marten se complementam em vários aspectos e, em certa medida, trocaram alguns “comportamentos culturais” durante esse longo período em que estão juntos: enquanto Luís tem um perfil mais “holandês-metódico”, Marten parece o típico “carioca”: descontraindo e relaxado. Um agradecimento especial ao Luís, nunca esquecerei aquele dia em que nos conhecemos durante uma feira internacional na UFRJ em 2013, quando coincidentemente fui até ele para perguntar se ele conhecia a tese do Giancarlo. Eu não estaria aqui em Twente se não fosse você.

Gostaria de agradecer aos membros do meu comitê de defesa: Prof. Dr. Frederick Benaben, Prof. Dr. Ir. Bedir Tekinerdogan, Prof. Dr. Ir. Maarten van Steen, Prof. Dr. Roel Wieringa e Profa. Dra. Maria Luiza M. Campos. É uma

honra tê-los neste comitê. Em particular, Roel, por acompanhar de perto minha trajetória de doutorado e por seus preciosos feedbacks; e Maria Luiza, minha "mãe acadêmica" e guru, por todo o apoio que ela tem me dado nesses mais de dez anos juntos.

Gostaria de agradecer aos meus colegas do grupo Services and CyberSecurity (SCS) pela cooperação e por me acolher na equipe. Em particular, gostaria de expressar uma gratidão especial a Suse, que agiu como mãe para mim, ajudando-nos com quaisquer problemas. Ela se tornou uma amiga próxima da minha família (outro "oma" do Teo), a quem temos muito carinho e respeito. Agradeço também a Geert Jan pelo apoio total à qualquer necessidade de laboratório e à Bertine, que também ajudou sempre que eu precisava.

Obrigado aos grupos NEMO (João Paulo, Giancarlo, Renata, Veruska, John, Freddy, Bernardo, entre outros) e LPRM (Patricia, Zé, entre outros) da UFES, e ao grupo GRECO (Marcos Borges, Jonice, entre outros). Um agradecimento especial à Patricia Dockhorn por ser uma espécie de "orientador de longa distância", me ajudando desde a proposta do doutorado até a análise dos resultados desta pesquisa. Obrigado também aos colegas de outras instituições, como Tiago, Roberta e Nicola (LOA); Katarzyna e Pawel (SRI-PAS); Pablo e Miguel (porto de Valência); Carlos (UPV); Laura (TNO); Norman (ITC) e muitos outros que interagiram comigo durante conferências e propostas de projetos. Agradeço à agência nacional brasileira CAPES pelo apoio financeiro (BEX 1046-14-4) e à Comissão Europeia pela colaboração no âmbito do projeto INTER-IoT.

Obrigado a todos da "família Enschede", como o "escritório brasileiro" em 2015 (Zé Gonçalves, Robson, Carlos, Glaucia e Rodolfo); os "cidadãos de Enschede", incluindo (mas não se limitado à) Luiz Olavo, Luciana, Jair, Priscila, Ivan, Bia, Henrique, Eveline, Lidiane, Pedro, Diego, Ju, Filipe, Steph, Anderson, Milena, André, Natália, Maurício, Thais, Felipe, Aline, Liniker, Suelen, Philip, Johanna, Eduardo, Nayeli, entre muitos outros amigos. Vocês não pouparam esforços para tornar a vida tão "gezellig" aqui! Um agradecimento especial à Luiz Olavo por todas as questões envolvendo a iniciativa GO-FAIR e pesquisas relacionadas; para "oma" Patricia e "opa" Jaury por toda a assistência com a vida acadêmica e pessoal, atuando como psicóloga nos momentos difíceis. Obrigado por serem pessoas tão boas na qual podemos depositar toda a nossa confiança.

O apoio da minha família (incluindo Kitty, Luli e parentes) e queridos amigos no Brasil também foi fundamental para essa conquista, principalmente dos meus pais e da minha querida tia, que sempre estiveram lá para mim; bem como amigos do ONS que me incentivaram para essa aventura. Um agradecimento especial aos meus pais José e Lucia, por me fornecerem toda a educação necessária para alcançar este título e o amor incondicional, mesmo quando eu estava ausente.

Este livro é em memória à "tia" e dedicado aos meus pais José e Lucia, à minha esposa Bel e ao meu filho Teo. Nada disso seria possível sem vocês.

Contents

1. Introduction	17
1.1 Disaster Risk Reduction	17
1.2 Interoperability of IoT Early Warning Systems	19
1.3 Research Challenges	25
1.4 Research Objectives and Scope	27
1.5 Research Methodology	28
1.6 Thesis Structure	30
2. Conceptual Framework	33
2.1 Disaster Risk Reduction	33
2.2 Interoperability	40
2.3 Semantic Technologies	51
2.4 Context- and Situation-Aware Applications	63
3. Internet-of-Things Platforms	71
3.1 Enabling Technologies	71
3.2 Microsoft Azure IoT Platform	82
3.3 Semantic Interoperability	87
3.4 Concluding Remarks	100
4. Review on EWS interoperability	103
4.1 Interoperability of Early Warning Systems	103
4.2 Result Analysis	109
4.3 Interoperable Standard-Based EWS	114
4.4 Emergency Data Representation	118
4.5 Challenges and Research Questions Revisited	124
4.6 Concluding Remarks	127
5. The SEMIoTICS Framework	129
5.1 Reference Architecture	129
5.2 Semantic Model-Driven Methodology	138
5.3 Standard-based FAIR Data for IoT EWS	144
5.4 Concluding Remarks	145
6. Improving the SEMIoTICS Foundations	149
6.1 Sematic Translations for the IoT	149
6.2 Situation Foundations for Context Modelling	159
6.3 Adapting SML for IoT EWS	167

6.4 Concluding Remarks	174
7. Case Study: The Valencia Port IoT EWS	177
7.1 INTER-IoT: Detecting Accidents with Trucks	177
7.2 INTER-IoT-EWS Solution	186
7.3 INTER-IoT-EWS Functional Validation	198
7.4 Chapter Conclusions	203
8. SAREF4health: IoT e-Health Ontology	205
8.1 Semantic Interoperability for Healthcare	205
8.2 Limitations of ECG Representations for the IoT	209
8.3 Ontology Development	212
8.4 Ontology Validation	218
8.5 Chapter Conclusions	221
9. Performance Validation	223
9.1 Comparison of JSON versus JSON-LD for IoT	223
9.2 Semantic Translation Analysis	228
9.3 Performance Analysis	237
9.4 Cost Analysis	240
9.5 Chapter Conclusions	241
10. Conclusions	245
10.1 Challenges and Lessons Learned	245
10.2 Main Contributions	251
10.3 Limitations and Future Work	253
Author Publications	257
Stakeholders in EWS interoperability	259
Translations of SSN/SOSA and SAREF	263
Meta-model of SML 2.0	267
FAIR Research Data Management	277
References	287

Introduction

This thesis presents a framework that aims at improving the semantic interoperability of early warning systems (EWSs) supported by IoT technologies. In this chapter, we describe the research motivation, presenting the thesis context and relevance. Section 1.1 describes the problem context by presenting the disaster risk reduction (DRR) process, emphasizing the role of early warning system (EWS) and IoT technologies in DRR. We discuss the semantic interoperability problem for the integration of multi-hazard EWSs in Section 1.2, considering the most advanced initiatives in the disaster and emergency management field. Section 1.3 describes the three main challenges that need to be addressed in combination to improve the semantic interoperability of IoT EWSs: semantic integration, performance and situation-awareness. Section 1.4 presents the main research objectives and delimit the scope of this thesis. Section 1.5 defines the research methodology used in this thesis. Section 1.6 outlines the thesis structure and describes how each chapter is related to our publications.

1.1 Disaster Risk Reduction

According to the United Nation International Strategy for Disaster Reduction (UNISDR), coordinated by the United Nations Office for Disaster Risk Reduction, a disaster is “a situation where serious disruption of the functioning of a community or a society occurs, involving widespread human, material, economic or environmental losses and impacts, which exceeds the ability of the affected community or society to cope using its own resources” [1].

Disaster situations are caused by disastrous (hazardous) events [2] and are, by nature, complex and dynamic [3]. A disaster situation is triggered by these hazardous events, which cause the disruption of the functioning of a community, i.e., the effect of these events exceeds the capacity of a community to cope with the situation by using its own resources, requiring assistance from external sources. The effect is often widespread and lasts for a long period, involving human losses and/or injuries, as well as damages to properties and/or to the environment.

Over the past 30 years, there has been a substantial increase of all types of hazardous events leading to disaster situations. It is estimated that more than US\$ 1.7 trillion has been spent on recovery from damages of disasters, more than 2.9 billion people have been directly affected and more than 1.2 million people have died because of disasters [4]. Disaster Management (DM) (or emergency or crisis management), addresses urgent social needs to prevent (mitigate), prepare, respond and recover from disaster situations, involving a number of different roles, e.g., emergency managers, first responders and victims [5]. The prevention of disasters through risk reduction and planning activities are permanent and, over the past fifteen years, a new paradigm for DM has emerged to cope with this complexity, namely Disaster Risk Reduction (DRR).

DRR is the modern and holistic paradigm of DM, which prescribes systematic actions to analyze and reduce the causal factors of disasters, such as reducing exposure to hazards, lessening vulnerability of people and property, improving environmental management, and enhancing preparedness through early warning of adverse events [6]. The main institutional stakeholder in DRR is the UNISDR.

1.1.1 Early Warning Systems and the Internet-of-Things

One of the main DRR components is an Early Warning System (EWS), which is an integrated system that supports disaster prevention, when the impact is imminent, and disaster recovery, when the disaster has already happened. An EWS monitors physical entities, detecting situations of interest and warning relevant parties. An EWS needs to provide timely and effective information that allows individuals exposed to take actions to reduce their risks and prepare to respond [7].

The UNISDR framework for the development of an effective people-centered EWS describes the four main elements of an EWS: risk knowledge, monitoring and warning service, dissemination and communication, and response capability [8]. In this thesis we give emphasis to the ICT viewpoint of EWSs, i.e., the technological aspects. The concept of EWS as a system-of-systems is exploited in many initiatives [7-17]. An EWS is a combination of information systems that are integrated through a workflow that comprises sense, detect, decide, broker and respond tasks, in the given order [17]. An EWS is developed as a distributed

information system [7] and, usually, its design is guided by the requirements of a response system [18].

EWSs can benefit from the Internet-of-Things (IoT) technologies, such as IoT platforms, to realize (near) real-time data acquisition, risk detection and message brokering between data sources and warnings' destinations [12]. The evolution and widespread use of smartphones enable numerous services based on multiple sensors attached to these devices.

The concept of "IoT EWS" is often cited in the literature [14, 19-21] to denote EWSs supported by IoT technologies. If on the one hand IoT technologies improve EWSs with new and more accurate features, on the other hand the increasing amount of devices leads to processing and scalability problems, such as the impact of big data traffic on wireless networks, performance and, the variety of data models to encode the messages.

1.2 Interoperability of IoT Early Warning Systems

Seamless integration is the process of smoothly adding a new feature or program, e.g., routine, application or device, in a way that the existing system keeps operating without introducing defects or failures [22-24]. The information exchange is a fundamental capability to allow seamless integration of EWSs, which can leverage the efficiency of the EWS workflow tasks.

The ability to exchange information among two or more systems (or components), and use this information meaningfully, is defined as interoperability [25]. Interoperability is an important aspect of effective IoT EWSs, which enables (1) the integration of EWS components, such as sensors, devices and platforms; and (2) interworking with other EWSs. The level of interoperability depends on the standardization of interfaces, data exchange formats and protocols [16].

The new *European Interoperability Framework* (EIF) [26] describes four interoperability levels: technical, semantic, organizational and legal interoperability. Technical interoperability refers to linking ICT solutions, such as information systems and services, which includes aspects related to the, e.g., interface specifications, service protocols and data integration services. Semantic interoperability refers to the ability of two or more systems to share precise format (syntax) and meaning (semantic) of the data elements, where semantic technologies, e.g., linked data, play a determinant role. Organizational interoperability refers to the alignment of business processes and business roles among collaborating organizations. Legal interoperability refers to the ability of different organizations to work together even if they are under different legal frameworks.

In software engineering, data coding and formatting, i.e., the binary encoding and the packaging of messages/streams that carry data, is often called syntactic interoperability [27]. In EIF, this aspect is approached at both technological (within ICT domain) and semantic levels (within the context domain). In addition, in software engineering, the dialogue aspect refers to process synchronization for the exchange of messages, i.e., process interoperability [28], which directly maps to the EIF organizational layer.

If on the one hand IoT technologies offer potential to improve the EWS efficiency and effectiveness, on the other hand this potential can only be exploited if interoperability challenges are addressed at all interoperability levels. The importance of EWS interoperability is reinforced by the UN agenda for DRR, i.e., the Sendai framework [29], which defines a set of global targets to be achieved by 2030. Global target G refers to “substantially increase the availability of and access to multi-hazard early warning systems and disaster risk information and assessments to the people by 2030”.

UNISDR emphasizes that achieving this goal depends on improving risk identification through a holistic approach for EWS integration [30]. The Sendai framework emphasizes the requirement to share disaster risk information and to increase data availability towards a multi-hazard integrated global warning and response system, which must be tailored to user needs across distinct and overlapping domains [29]. At the EU level, the low level of information sharing within and across organizations is the major problem to integrate EWSs, which led to the EU initiative to integrate all official EWSs at the global level [31].

The ability to seamlessly integrate IoT EWSs with other (existing and new) systems is a fundamental requirement for “processing and management of multisource data from multi-sensors” [32], which relies on the interoperability level of the involved components [33]. Therefore, interoperability means that all the available data can be used in the best way possible for better decision-making and more effective prevention. Furthermore, “one of the main challenges in developing such a system (IoT EWS) involves the seamless integration of physical things and human behaviors on a changing site” [34]. Seamless integration of distinct IoT EWSs to achieve an interoperable *Multi-Hazard EWS* (MHEWS) was discussed in [35], in the context of the *Integrated Public Warning System* (IPAWS), which strengthens the requirement on adopting and evolving interoperability standards.

Interoperability standards for IoT have been defined to improve the syntactic interoperability of EWSs in multi-agency sensor information integration [13, 36, 37], such as the OGC’s Sensor Web Enablement (SWE)¹, the OASIS Emergency

¹ <http://www.opengeospatial.org/ogc/markets-technologies/swe>

Data Exchange Language (EDXL)² and Health Level Seven (HL7)³ standards. For example, the IPAWS and the German Indonesian Tsunami Early Warning System (GITEWS)⁴, the precursor of the PTWS, implement EDXL-CAP, which is a common alert data format protocol [38].

1.2.1 Interoperable EWS Conceptual Architecture

A conceptual architecture of an interoperable *Standard-Based* EWS was introduced in [16] and is illustrated in Figure 1.1, which is aligned with the EWS workflow tasks. In this architecture, data are acquired from sensor systems (sense), performed by a component often called *Upstream Data Acquisition*, and are represented according to interoperability standards, e.g., OGC SWE. The *Upstream Data Acquisition* component is responsible for pre-processing and storing data in an internal *Context Database*.

Figure 1.1. Typical standard-based EWS conceptual architecture [16]

The *Models* component represents the perception of the contextual elements as a whole, i.e., the situation identification, responsible for emergency situation detection (detect), e.g., rule-based or learning-based approaches [39]. The *Decision and Action* component is executed by applying the intrinsic rules of the models over the context data as a first step.

The *Decision and Action* component supports decision making (decide) by allowing the end user (EWS administrator) to configure actions according to the situation identified, usually with support of a workflow management system. Therefore, this component allows the representation of the projection of the possible future status, which is used by the system in the process of decision

² https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=emergency

³ <http://www.hl7.org/>

⁴ <http://www.gitews.org/en/homepage/>

making. According to pre-configured actions, the EWS sends messages towards different target groups through the *Downstream Information Dissemination* component, responsible for message brokering (broker). Usually each target group has specific information requirements for the message content, thus, the EWS must include the adequate contents according to these requirements. The target groups can be either humans, which can receive alerts through different technologies, as mobile applications and low-frequency radio, or actuators for automatic emergency response (respond).

Figure 1.2. MH-EWS reference architecture as a systems of standard-based EWSs

Often in the literature OASIS EDXL is mentioned as the common standard for downstream data representation for interoperable EWSs [16, 38, 40-46]. The interoperable EWS conceptual architecture allows an EWS to be able to integrate with other EWSs through event-based message-oriented middleware (MOM), which can play the role of either a context broker or a service broker. Figure 1.2 illustrates the concept of MHEWS as a system of distinct EWSs that interoperate through a MOM. Although semantic interoperability is absolutely necessary, the main issue of this standard-based lexicon approach is that it ignores semantic interoperability among the EWS components. Therefore, a similar approach with semantic standards is also necessary, which brings additional challenges.

1.2.2 Semantic Technologies for IoT EWS

The focus of this thesis is the semantic interoperability level of EWSs, i.e., data interpretation or the assignment of meaning to data. Some researchers claim that seamless integration depends on the improvement of semantic interoperability, especially when IoT solutions are involved [47]. Classical EWSs tend to use semi-formal or informal models, while only a few use semantic models, either lightweight, i.e., a semantic model of classes and their relations, or heavyweight,

i.e., a semantic model enriched with axioms that narrow the semantic interpretation [14].

Most of the semantic EWS approaches [14, 20, 21, 48-51] extend the *Standard-Based* architecture by using semantic technologies to represent data, both upstream and downstream, taking advantage of inference capabilities. Usually, the concept of a common information space is exploited in this type of approach, linking semantic technologies to SOA [21]. Performance tends to be a drawback, which makes scalable time-sensitive data exchange and processing from heterogeneous data sources quite challenging. The *Semantic IoT EWS* approach [14] partially addresses this challenge. This approach emerged from the consecutive initiatives of tsunami EWSs, e.g., the FP6 projects Distant Early Warning System (DEWS) and the German Indonesian Tsunami Early-Warning System (GITEWS), and was developed within the Collaborative, Complex, and Critical Decision Processes in Evolving Crises (TRIDEC) project [52] as evolution of the *Standard-Based* approach.

The *Semantic IoT EWS* approach deals with the challenge mentioned above, i.e., it improves semantic interoperability of an EWS with a scalable solution for data exchange and processing, by providing a balanced way to use lightweight and heavyweight ontologies. Lightweight ontologies are used in *Upstream Data Acquisition* and *Downstream Information Dissemination* components to reduce the messages payload during data exchange and processing. Heavyweight ontologies are used in decision support to add additional semantics to the data, enriching the fusion of new and historical data by enabling the discovery (inference) of new knowledge. This new information is serialized as lightweight semantics to be included in the warning messages that are brokered through the downstream information dissemination component. Only few recent initiatives use heavyweight semantics in EWSs, such as [53].

The Semantic IoT EWS approach introduced the Decision Support Ontology (DSO), which is a heavyweight ontology that is aligned to the W3C Semantic Sensor Network (SSN). W3C SSN is the most popular lightweight ontology in the IoT domain, founded on the OGC SWE standards. DSO also incorporates terms from the OGC Observations and Measurements (O&M) standard. Upstream sensor data are annotated only with the SSN predicates, excluding the few axioms included in SSN. Annotation is applied at the broker (gateway) level, since sensors only need to interoperate with a control center via a sensor's access node. This type of *Semantic Gateway* is a relatively new concept in the IoT domain [54, 55] and leverages on the evolution of the low-cost gateway devices, including smartphones and other IoT devices.

The semantic gateway publishes the semantically enriched data in a publish/subscribe MOM, enabling the EWS to acquire sensor data and store them in a knowledge base, i.e., the context database as an ontology management service that can be accessed through a semantic registry UI. The ontology management

component is responsible for adding heavyweight semantics to the data. A rule-based approach is adopted for risk identification within the decision and action component, which is implemented with a rule management service that takes advantage of the heavyweight semantics for inference. The patterns that represent a risk are pre-configured with this service through a decision table UI and each risk type is linked to a set of workflows that represent the corresponding response actions. The workflows are implemented with a decision support workflow service UI, which can trigger workflows when risks are identified. Risk assessment is performed by experts in an interactive way according to the workflows by using that UI to make decisions, which are based on data visualization, simulation results and analytic reports that are generated by the processing services.

EWS semantic interoperability has also been tackled with other approaches that apply domain-specific ontologies to support meaningful data integration [46, 56], and several other ontologies for DM were proposed. The conceptualization regarding events and situation-awareness is widely exploited, probably because it is often necessary to represent dynamic temporal aspects of emergencies. For example, the Event-Model-F well-founded ontology [57] was developed to address the requirements of sharing event interpretations in emergency response, exploiting the role of causality of events and situations.

From studying these semantic EWS approaches, we observed that it is necessary to match equivalent conceptualizations and combine multiple domain ontologies, because of the cross-disciplinary nature of the emergency field, in order to provide a higher level of interoperability for EWSs that acquire data fusion from multiple sources. The concept of *Semantic Gateway* can address this issue by enabling the execution of semantic translations that can tackle specific semantic integration problems, such as semantic overload, ambiguity and information distortion [58].

The development of semantic translations depends on mappings between two semantic models, which guides how the input data represented with the source ontology can be transformed in data represented with the target ontology, equivalent to model transformations in Model-Driven Engineering (MDE) [59]. For example, the specification of bi-directional transformations between EDXL for Tracking Emergency Patients (EDXL-TEP) and HL7 v2⁵ provides a set of mappings regarding the patient concept, albeit EDXL-TEP and HL7 v2 are not semantic models. W3C SSN ontology and several other ontologies in the context of IoT, e.g., ETSI SAREF, provide mappings to OGC SWE, especially SensorML and O&M, enabling higher interoperability among these standards.

To the best of our knowledge, the use of *Semantic Gateway* approaches that support the whole lifecycle of semantic translations' development, i.e.,

⁵ <http://docs.oasis-open.org/emergency/TEP-HL7v2-transforms/v1.0/TEP-HL7v2-transforms-v1.0.html>

specification and implementation, considering their configuration at runtime, has not been exploited in any EWS initiative. In particular, the *Semantic IoT EWS* approach concludes that this type of mechanism is as trend “in order to more effectively apply the use of semantic computing models for use with EWS” [14], but it is still lacking in the literature.

1.3 Research Challenges

The increasing number of initiatives claiming their smartness based on automated data collection and analysis have only worsened the integration problems: conventional and new sensor network technologies, distinct communication system running proprietary incompatible protocols on legacy DM systems, data in formats that are not machine readable stored in several databases, multiple standardization initiatives of the same domain, as well as crowd sensed data are increasing the volume and heterogeneity of the data meant to be processed by an EWS. Therefore, the poor interoperability of such technologies is a big barrier for enabling smart behavior of emergency services based on IoT EWSs. In this section we present main general challenges and the derived research questions to address them.

As a starting point, the high-level research question is stated as:

How to improve the semantic interoperability of IoT EWSs?

In the past decade, we have experienced a wave of new interoperability standards for sensor systems, including lexicon and semantic models, as well as proprietary data models, resulting in a number of different data representation assets. Some projects incorporated these standardized ontologies, but few of them gave emphasis to their alignment to support sensor data integration.

Furthermore, the emergency domain is characterized by a variety of heterogeneous sources and information targets, and information itself is represented and communicated using different vocabularies that rely on different standards. All these vocabularies are means to express different points of view about the domain, and are used by different stakeholders participating in the DM process, which can adopt different terms to express the same concept and the same term to express different concepts. A review on challenges of communication systems for disaster response concludes that “future disaster communication system should include data that reflect the knowledge transferred to the community together with level of community awareness” [60].

Semantic technologies, e.g., ontologies, are the tools to define explicitly the connection between terms and concepts, but each ontology represents these connections with predicates based on the designers' vocabulary. In order to reach semantic interoperability among stakeholders, the emergency domain requires ontology alignments based on a deep ontological analysis. Our ambition is to enable EWS semantic interoperability for DRR by considering not only IoT ontologies, but also addressing cross-domain relationships through the alignment of multiple domain ontologies. A particular trend in the EWS evolution is to be able to interoperate with health monitoring systems that support medical emergency services [61], thus, considering *e-Health* standards. For example, EDXL-TEP addresses the monitoring of patients among health units, such as hospitals, clinics and ambulances. Therefore, logistics is also an important domain to be included.

The *Semantic IoT EWS* approach [14] addresses the challenges of scalable time-sensitive data handling from heterogeneous sources, enabling effective responses. As mentioned in Section 1.2.2, this approach also introduces the DSO ontology “to aggregate and align multiple ontologies to support compound EWS semantics and ontology commitments”. However, this approach lacks support to the development of alignments of different domain ontologies that can be configured at runtime, i.e., it does not provide a mechanism for describing and executing semantic translations. Furthermore, efficiently publishing large volumes of semantically rich sensor data is a major architectural challenge due to inefficient data exchange throughput caused by the representation overhead imposed by the encoding schemas of the current semantic models, e.g., OWL or RDF serialized as XML, requiring further research especially for EWSs [10].

The problem investigated in this research was leveraged by a systematic literature review on EWS interoperability. Our knowledge questions targeted the existing interoperable EWSs and their architecture and components' functions, emphasizing the role of data representation mechanisms, e.g., data/message models, protocols and ontologies. Since semantic models rely on syntactic models, the review also included these data representation approaches, giving emphasis to the IoT and healthcare domains as input (data sources) and emergency services domain as output (warnings). In particular, the concept of MHEWS is exploited here, since this type of system can benefit from having a higher interoperability to be able to seamlessly integrate multiple EWSs with other monitoring systems, which is an explicit UN requirement [29]. We identified the main issues concerning the improvement of MHEWS interoperability for smart situation-aware emergency services. To achieve a higher level of semantic interoperability, three main challenges were identified, which are interrelated, i.e., they need to be tackled together because achieving one impacts the others. These challenges are translated to the following research questions:

(RQ1) *Semantic integration of a variety of data sources:*

How to make systems to understand each other, i.e., avoid semantic errors and distortions, when multiple ontologies, standards and data models from different and overlapping domains are involved, considering their syntactic and semantic alignments?

(RQ2) *Processing in time- and safety-critical applications:*

How to achieve the appropriate performance and scalability for real-time upstream data acquisition, emergency risk detection and message brokering, in terms of total transaction time and quality of service?

(RQ3) *Data analysis for effective responses:*

How to provide high quality situation awareness, i.e., perception, comprehension and projection, to improve emergency decision support?

1.4 Research Objectives and Scope

The general objective of this thesis is *to improve the semantic interoperability of IoT EWSs for smart emergency services, enabling seamless integration of multiple data sources and targets, including other EWSs and large-scale IoT systems, by addressing heterogeneity problems at both syntactic and semantic levels.*

This thesis aims at enabling the integration of multiple data sources for integrated sensing, to multiple targets for integrated actuation, addressing semantic and syntactical interoperability to achieve smart behavior. Therefore, this research addresses heterogeneity in the integration of IoT solutions, by extending web technologies with semantic technologies (e.g., ontologies) and applying syntactical interoperability standards to improve data representation of IoT devices, as well as communication protocols used by different sensors and actuators.

More precisely, the goal of this thesis is to improve the semantic integration capabilities of IoT EWS components to enable situation identification from the integration of different data sources; and issue early emergency notifications to different targets. To achieve this goal, the artefact designed in this thesis is a meta-system, since it is a general framework to build semantic interoperable EWS. We call this framework “SEmantic Model-driven engineering for IoT Interoperability of emergenCy serviceS” (SEMloTICS). The main stakeholders (end users) of SEMloTICS are software developers, including architects, designers and programmers, while functional beneficiaries include a wide range of roles involved in DM, such as the EWS administrator, which can be an emergency manager, as well as first responders and victims.

Specific targets to achieve the SEMIoTICS goal is the definition of guidelines to adopt and, if required, to adapt different IoT solutions, taking advantage of IoT platforms' capabilities as part of the implementation strategy of the SEMIoTICS framework. Moreover, the SEMIoTICS framework aims at providing integration capabilities towards the MHEWS concept, i.e., enabling the integration of different EWSs. Therefore, the implementation strategy of the SEMIoTICS framework also considers integrating not only IoT devices, but also IoT devices with other systems. In order to improve data findability, accessibility, interoperability and reusability, the SEMIoTICS guidelines should be compliant to the FAIR Data principles [62], enabling an IoT EWSs to produce FAIR data.

Since the SEMIoTICS framework is designed to be domain-agnostic, the scope of this thesis include the illustration of developing different types of EWSs, such as for epidemiological surveillance (Tuberculosis and Zika), vehicle collisions and cardiac arrhythmia. However, it is out of the scope of our work to develop fully-functional IoT EWSs for all these use cases. Rather, we focused on fully applying SEMIoTICS to the development of an EWS for detecting traffic accidents that requires the integration of different domains. General properties that are inherent of medical emergency services are exploited, such as sensors that monitor people's vital signs, which could be related to any disaster type.

1.5 Research Methodology

This thesis followed the Design Science Methodology (DSM) [50] with short interactions between treatment design and validation activities. According to the DSM template for problem design, the problem addressed by this thesis is:

- *Improve the problem context: the semantic interoperability of IoT EWSs.*
- *By designing a treatment: SEMIoTICS framework.*
- *That addresses requirements: semantic integration of a variety of data sources (RQ1), processing in time- and safety-critical applications (RQ2), and high quality situation awareness for decision support (RQ3).*
- *In order to efficiently and effectively reduce the risks of disasters.*

DSM describes the engineering cycle as a process of four phases (Figure.2):

- (1) Problem investigation and/or implementation evaluation.
- (2) Treatment design.
- (3) Treatment validation.
- (4) Treatment implementation and implementation validation.

For problem investigation, we conducted a systematic literature review (SLR) to investigate the current open problems regarding semantic interoperability of EWSs, following the methodology for SLR in software engineering [63]. Knowledge goals covered the common architectures for interoperable EWSs and the data representation mechanisms of emergency services that may interact with EWSs. This state-of-the-art analysis considered the main initiatives (e.g., projects) on emergency interoperability and available surveys and reviews on EWS. Furthermore, the SLR search criteria benefited from the background on DM terminology, semantic interoperability research and IoT technologies.

Although we have validated the treatment developed in this research, a formal implementation evaluation (according to DSM) was not conducted in this research. This thesis used the outcomes (e.g., publications) reporting the open issues of semantic interoperability of EWSs. Therefore, our problem investigation consists of a sample-based research where cases of interoperable EWSs were studied, resulting in the identification of open issues from the literature. We have defined a semantic interoperable situation-aware EWS architecture from this investigation. This reference architecture organizes the main functions that an interoperable EWS needs and their interactions to fulfill the requirements, illustrating how the research challenges can be addressed.

Figure 1.3. The DSM engineering cycle [64]

Empirical cycles were applied to scrutinize the reference architecture, studying possible solutions to build each reference architecture component as well as the integration methodology to link these components as an IoT EWS. Therefore, the treatment design developed in this research is a framework, coined as SEMIoTICS, which is based on this reference architecture and guides the use of technologies to address the challenges.

The SEMIoTICS framework was validated through different validation approaches. For example, while the improvements on the Situation Modeling Language (SML) were validated through a single-case mechanism experiment, the performance analysis study comparing JSON to JSON for Linked Data (JSON-LD)

was validated through a statistical difference-making experiment following the Experimentation in Software Engineering methodology [65].

The functional validation of SEMIoTICS as a whole was performed in collaboration with the H2020 INTER-IoT project, employing a test-driven development methodology. This validation started by demonstrating how the IoT EWS for INTER-IoT, coined INTER-IoT-EWS, was developed with the SEMIoTICS framework to cover the emergency scenario of the project. This validation tested the suitability of SEMIoTICS from a functional point-of-view, i.e., focusing on the risk reduction of fatal accidents at the port of Valencia by improving health prevention and quick emergency response [66]; and from a non-functional point of view, i.e., semantic integration of different IoT application domains. INTER-IoT-EWS was validated both in the laboratory (University of Twente) and in the site (port of Valencia).

1.6 Thesis Structure

This thesis is structured according to the phases of the DSM engineering cycle. Part I describes the problem investigation by first presenting the conceptual framework and discussing IoT platform technologies, and then, presenting the SLR, which allowed us to detail the research challenges and elicit requirements for the framework. Part II describes the treatment design. Part III describes the treatment validation, which demonstrates how the framework was validated and the analysis of the results. Figure 1.3 depicts the thesis structure.

Figure 1.4. Structure of this thesis

Problem investigation	Chapter 2 Conceptual Framework (background)
	Chapter 3 Internet-of-Things Platform Technologies
	Chapter 4 Literature Review: Interoperability of EWSs
Treatment design	Chapter 5 The SEMIoTICS Framework
	Chapter 6 Improving the SEMIoTICS foundations
Treatment validation	Chapter 7 Case Study: The Valencia Port IoT EWS
	Chapter 8 Non-Functional Validation
	Chapter 9 Non-Functional Validation

The remaining of this thesis is organized as follows:

Part I: Problem investigation

- **Chapter 2:** Introduces the conceptual framework describing the main theoretical background used in thesis, comprising disaster management terminology, interoperability, semantic technologies, model-driven engineering and situation-aware applications.
- **Chapter 3:** Introduces the Internet-of-Things Platform concept, its main enabling technologies, the interoperability aspects and the ontology engineering best practices for the IoT domain.
- **Chapter 4:** Describes the SLR that was carried out in order to investigate interoperable architectures for EWSs and data representations for disaster management, highlighting the open issues and detailing the research questions. This chapter is related to this publication from the author: [67].

Part II: Treatment design

- **Chapter 5:** Introduces the SEMIoTICS framework, listing the main requirements derived from the research questions and how the framework components address them. This chapter also describes the semantic model-driven methodology adopted by the framework, which is enriched with the standard-based Findable, Accessible, Interoperable and Reusable (FAIR) data principles. This chapter is related to these publications from the author: [68] [69] [70]
- **Chapter 6:** Presents the main theoretical advances performed during this research towards the improvement of EWS semantic interoperability: (1) the semantic translations between W3C SSN/SOSA and ETSI SAREF for the SEMIoTICS input handler, enabling IoT standardized semantic integration; (2) the abstraction mechanism based on a foundational ontology, which supported the development of the SEMIoTICS context model as an emergency core ontology with multiple alignments; (3) the Situation Modelling Language (SML) redesign, adhering SML with the abstraction mechanism and providing new features to SML; and (4) the model-driven transformation approach to generate software components based on the IoT platform technologies, such as CEP engine code. This chapter is related to these publications from the author: [71] [72] [73] [74]

Part III: Treatment validation

- **Chapter 7:** Describes the single-case experiment of SEMIoTICS in the port of Valencia use cases within INTER-IoT project. It comprises the development

of a cardiac IoT EWS to monitor truck drivers delivering goods in the port; the system is able to detect accidents and assess accident risks by integrating health and logistics data. Upon situation-awareness, the system is capable of warning multiple targets about risks and accidents. This chapter also describes the functional validation of this IoT EWS, including emergency simulations. This chapter is related to this publication from the author: [75]

- **Chapter 8:** Describes the non-functional validation of SEMIoTICS in terms of how semantic interoperability was addressed in INTER-IoT-EWS through the extension of a standardized ontology, coined SAREF4health ontology. This ontology was developed according to the SEMIoTICS methodology to address the INTER-IoT project requirements, focusing on the representation of time series, such as ECG data for real-time health monitoring of the drivers delivering goods at the port of Valencia. The validation of SAREF4health is also presented. This chapter is related to this publication from the author: [76]
- **Chapter 9:** Describes the non-functional validation of SEMIoTICS in terms of the effects of semantic interoperability for the system performance, such as the statistical difference-making experiment comparing JSON-LD to JSON within common IoT scenarios using SAREF standard. The semantic translations between SSN/SOSA and SAREF were validated in terms of accuracy and efficiency. This chapter also discusses the semantic broker performance with cloud infrastructures, comparing to the *Semantic IoT EWS* performance (up to 700msg/sec). Finally, this chapter describes how the research data were managed according to the FAIR data principles, being validated through the current FAIR metrics. This chapter is related to this publication from the author: [77]
- **Chapter 10:** Concludes the thesis by discussing the lessons learned, our main contributions and limitations. It also proposes topics for future work.

Appendix

- **Appendix A:** Author publications during the development of this thesis.
- **Appendix B:** List of main stakeholders related to EWS interoperability.
- **Appendix C:** List of semantic translations between ETSI SAREF and W3C SSN/SOSA using the IPSM language.
- **Appendix D:** SML 2.0 metamodel.
- **Appendix E:** FAIR research data management with an existing data repository (EUDAT B2share).

Conceptual Framework

This chapter presents the background of this research, describing the main constructs of the scientific theories used in this thesis. Therefore, this chapter outlines the problem context, providing an overview of Disaster Risk Reduction (DRR) and interoperability, along with some of the current solutions based on semantic technologies and situation-aware applications. Section 2.1 describes DRR and the practice of Disaster Management (DM), emphasizing *Endsley's Situation-Aware* (SA) theory and the role of Early Warning System (EWS) for DRR. Section 2.2 discusses interoperability classifications and standards, such as the *European Interoperability Framework* (EIF), the *Findable, Accessible, Interoperable and Reusable* (FAIR) data principles and the role of *Model-Driven Engineering* (MDE) to improve software interoperability. Section 2.3 provides an overview of *semantic technologies*, covering the role of formal ontologies to address semantic interoperability, discussing ontology-driven conceptual modelling and the importance of ontology alignments for semantic translations. Finally, Section 2.4 describes the relevant research on situation identification techniques for the development of context- and situation-aware applications. It emphasizes on the *Situation Modelling Language* (SML) and its foundations to develop distributed applications with *Complex Event Processing* (CEP).

2.1 Disaster Risk Reduction

Disaster Management (DM), often referred to emergency management or crisis management, addresses the urgent social needs before, during and after a disaster situation happens. DM involves interactions among a number of parties playing different roles, e.g., emergency managers, first responders and victims [5], in each of the four DM phases: mitigation (or prevention), preparation (or preparedness), response and recovery. For a practical reason, here we consider disaster management, emergency management and crisis management as synonymous,

albeit the DRR community refers to disaster as a greater scale event, while an emergency has a smaller scale [78], which is a subjective classification. Crisis is often denoted as a situation out of normal operations that is caused by a disaster or an emergency.

The cyclical process of DM, depicted in Figure 2.1, shows the four DM phases and is well-known in the emergency field for decades [79]. The mitigation phase aims at preventing, reducing or eliminating human and material hazards through risk assessment and resources planning. The preparedness phase aims at prepositioning resources before the occurrence of a hazardous event, including long-term mitigation strategies. The response phase aims at responding to the hazardous event through rescue, relief, salvage and immediate damage assessment. Finally, the recovery phase aims at returning the affected area and victims' lives back to normality by performing damage assessments, restoration, re-habitation and repair.

Figure 2.1. Disaster Management cycle [80]

In practice, the mitigation of disasters through risk reduction is permanent and usually not triggered by a particular event, while preparation phase is triggered by events that can lead to a disaster situation, the responding phase is triggered by the hazardous event(s) and recovering is triggered by restoration events. Therefore, those four phases can overlap and are performed at some level before, during, and after the hazardous events occur. Furthermore, in reality, sometimes the response actions can even begin before the disaster actually happens, triggered by the almost certain expectation that hazardous events will trigger a disaster.

Over the past fifteen years, a new paradigm for DM has emerged to cope with this complexity, namely Disaster Risk Reduction (DRR). DRR is the modern and holistic paradigm of DM, which is the systematic practice of reducing the causal factors of disasters through the improvement of several activities, such as early warning of adverse events [6]. According to UNISDR, DRR focuses on strengthening economic, social, health and environmental resilience to achieve sustainable development, contrasting to DM, which aims at creating and

implementing plans to decrease the impact of disasters, where failures in plans can lead to damages.

The concept of disaster is classified as event [81, 82] or as situation [83]. This overloaded classification is found in glossaries of relevant stakeholders of DRR, such as UNISDR⁶, WHO⁷ and the International Disaster Database (EM-DAT)⁸. Inspired by these references, this defines disaster as: a complex situation that is triggered by one or more hazardous events interacting with conditions of exposure, vulnerabilities and capacity limitations. A disaster causes a crisis, which is a situation characterized by the disruption of the normal routines of communities, exceeding their capacity to tackle the situation with their own resources. A disaster leads (directly or indirectly) to one or more human, material, economic or environmental losses. Usually, the disaster concept is related to larger scale effects and/or longer period of existence when compared to the emergency concept (smaller scale). Furthermore, an emergency is often classified as a pre-disaster situation, in which daily procedures are suspended and actions are taken to avert a disaster [84]. A relevant taxonomy of disaster types is given by EM-DAT, relating a disaster type to hazard event types. A disaster can be natural or technological. Natural disasters include geophysical (e.g., earthquake), meteorological (e.g., storm), hydrological (e.g., flood), climatological (e.g., wildfire), biological (e.g., epidemic) and extraterrestrial (e.g., space weather). Technological disaster includes industrial accident (e.g., chemical spill), transport accident (e.g., road) and miscellaneous accident (e.g., explosion).

Over the last years, a large number of information systems, such as Decision Support Systems (DSS), have been developed to support all phases of DM [82]. Examples of these systems include distributed information systems (in terms of software and hardware) interacting to processes that involve people. Examples of these information systems are the Incident Management System (IMS), Incident Command System (ICS), Emergency Management System (EMS) and Early Warning System (EWS). In general, IMS, EMS and ICS are systems designed to store and share information about incidents in different levels [41]. An ICS can be defined as an EMS and supports emergency response by allowing the management of rescue operations using information from the incidents. IMS, EMS and ICS are usually integrated systems used by a Unified Operative Centre (UOC), which is an organization site responsible for DM [85]. Warning messages generated by an EWS are often consumed by EMS or IMS or ICS [86].

⁶ <https://www.unisdr.org/we/inform/terminology#letter-d>

⁷ <http://www.who.int/hac/about/definitions/en/>

⁸ <https://www.emdat.be/Glossary>

2.1.1 The Situation Awareness Theory

The notion of situation awareness is commonly used in the DRR field and some information systems for DRR are based on *Endsley's Situation Awareness (SA)* theory [87, 88] for dynamic decision making, especially adopted by the information fusion community in the 2000s [89]. The SA theory was developed in the 90s by the human factors community for dynamic decision making in human cognition, which is grounded on *Barwise's Situation* theory [90, 91].

Situation is a central concept in DRR, which has been discussed by many years in Philosophy, Linguistics, Semiotics and Computer Science. According to *Barwise*, a situation is characterized by the limited parts of the perceived reality that can be comprehended as a whole, akin the state of affairs [92]. A situation type is the set of patterns that composes a situation. One example is the situation of “John being infected with tuberculosis on July 9th 2015”, which is a unique instance of an abstraction of (structural and behavioral) patterns that forms a whole (i.e., an instance of a situation type). SA refers to the ability to recognize a situation.

Figure 2.2. The
Endsley Situation
Awareness Theory [87]

The SA conceptual framework is illustrated in Figure 2.2, which describes the situation awareness concept as a cognitive process of three levels (perception, comprehension and projection of future status) within the system factors of decision making. The first level refers to the perception of elements in the current situation by monitoring the state of the environment, thus, representing the collection of observed data as contextual elements. The second level refers to the comprehension of the current situation as a whole, i.e., the comprehension of current situation as a set of connections among the contextual elements. The third level refers to the action(s) that should be triggered to change or maintain the situation, i.e., the projection of future status.

According to the action(s) inferred by the SA, a decision is made, followed by the evaluation of the action(s) taken, i.e., the performance of actions. The workflow of the consecutive activities of SA, decision and evaluation is cyclical and provides feedback to the state of the environment. Each of these phases receives input from task/system factors, as system capability, stress and workload, and from individual factors, as personal goals and expectations, which rely on long term

memory and information processing mechanisms. This whole decision making process can be trained and is enhanced by experience [87, 88].

Some empiric research tests the impact of the information volume (“richness”) and information centralization on SA of emergency managers and first responders that interact with information systems (e.g., EMS and IMS) [93]. The main outcomes showed that if information is centralized in one coordinator then the information should be less rich to lead to higher SA. Otherwise, if information is distributed, then information should be enriched to lead to higher SA. In this research, SA is measured by calculating the situation understanding according to the demand and availability of attentional resources. The SA measurement also considers the current situation's stability, complexity and variability, as well as person's degree of alertness, concentration and mental capacity. Understanding is influenced by the available quantity and quality of information, as well as the familiarity to the situation type.

The SA Theory was especially adopted by the data information fusion (DIF) community [94] in the research on the combination of data from multiple inputs within the military domain. DIF techniques were approached by the Joint Directors of Laboratories Data Fusion (JDL DF) model, a widely used model for data fusion. JDL is represented as a hierarchical process of data and information fusion in two stages: low-level and high-level fusion processes. JDL defines spatio-temporal constraints and causality models. An extended description of different frameworks covering fusion levels of JDL DF including the *SA theory* is described in [94]. Usually, an Early Warning System (EWS) is an emergency system that follows some of the JDL DF techniques based on the SA theory [95, 96].

2.1.2 Early Warning System

The main component in DRR is an EWS, which is a distributed information system that is able to (1) monitor the physical world and (2) issue warnings if it detects abnormal situations [7]. An EWS is an integrated system that supports disaster mitigation, prevention and response. For example, an EWS can (a) forecast that a tsunami may occur within some weeks, enabling the authorities to mitigate the situation by preparing emergency plans; (b) warn people that a tsunami is about to hit the coast within some hours, preventing major losses; and (c) warn people that a road is blocked because of the impact of the tsunami on surrounding buildings, supporting the response phase.

An EWS needs to provide effective and timely information to allow people exposed to take actions to reduce risks and prepare to respond [7]. The concept of EWS was built upon the *Emergency Broadcast System* (EBS), which was intensively used in DM from the 60s to the 90s, and refers to the ICT technologies behind the

Emergency Alert System (EAS), which is the jointly DM coordinated system in the USA.

In the DRR literature, there are two main points of view regarding the EWS concept: the point of view of institutional and social aspects, which focuses on the integrated processes performed by multiple stakeholders, as in [31, 97-101]; and the ICT point of view, which focuses on the technological components of an EWS, as in [14, 16, 17, 34, 38, 102, 103]. The institutional point of view fits into the strategy and business levels of an enterprise architecture, while the ICT point of view fits into the application, technology and physical levels.

Figure 2.3. Main elements of an Early Warning System [7]

For example, these points of view are reflected by the UNISDR framework for the development of effective people-centered EWS [8], which describes the four main elements illustrated in Figure 2.3: (1) risk knowledge, (2) monitoring and warning service, (3) dissemination and communication, and (4) response capability. The response capability element describes an institutional requirement, e.g., building community response plans, while the monitoring element describes an ICT requirement, e.g., develop hazard monitoring. In this thesis, emphasis is given to the ICT point of view. Besides the four elements of the UNISDR framework, recent research showed that vulnerability aspects should also be considered by EWSs, such as environmental and social conditions, as well as economic and institutional capacities for dissemination and response [104, 105].

Figure 2.4. Workflow of an Early Warning System [17]

From the ICT point of view, the concept of EWS as a system-of-systems is used in many initiatives [7-17]. An EWS is an ordered chain of information systems that support sensing, detecting, deciding and brokering messages to appropriate targets. An EWS should support forecasting and signaling disturbances that affect

the stability of the physical world, providing timely information for the response systems to allocate resources and to take the response actions to minimize the impact of the events [17].

An EWS is developed as a distributed information system [7] and, usually, its design is guided by the requirements of a response system [18]. The common workflow of an EWS [17] is illustrated in Figure 2.4, and comprises the five main activities: sense, detect, decide, broker and respond. Each activity is supported by one or more subsystems, which are components that provide the required functions. The *SA theory* [87] is used as theoretical foundation for these activities:

- (1) Sense: equivalent to monitor or acquire data, i.e., to be able to perceive the elements from the current context, thus, to sense the data.
- (2) Detect: the process of comprehending the context elements and their relations as a whole, i.e., a situation. Detection is used to identify the situation(s) triggered by hazardous events.
- (3) Decide: the process of determining the actions that should be triggered according to the situation(s) detected, i.e., supports decision making by projecting the wanted future status. This type of function is usually implemented by systems grounded in both normative and descriptive decision theories.
- (4) Broker: the process to integrate decision systems with response systems, acting as mediator that shares data according to specific message formats, enabling the dissemination of information.
- (5) Respond: the process to perform the actions in the field, i.e., the reaction to the detected situation (e.g., evacuation or isolation of an area). Usually, these actions adhere to emergency plans. The efficacy of a warning is measured by the process chosen from the response system, i.e., how the response system interprets the received warning to determine the actions required to evade the risk.

Modern EWSs comprise software and hardware that implement the five functions of an EWS workflow [106]. In particular, EWSs can benefit from the Internet-of-Things (IoT) technologies to realize (near) real-time data acquisition, risk detection and message brokering between data sources and warnings' destinations considering local sensor networks and the cloud [12]. Particularly, the evolution and widespread use of smartphones enable numerous services based on multiple sensors attached to these devices, e.g., accelerometers, temperature sensors and position sensors.

Examples of well-known EWSs are: the *Integrated Public Warning System* (IPAWS)⁹, which monitors multi-hazard events in US and is coordinated by the

⁹ <https://www.fema.gov/integrated-public-alert-warning-system>

Federal Emergency Management Agency (FEMA); the *Pacific Tsunami Warning System* (PTWS)¹⁰, which monitors the Pacific ocean and is coordinated by the *Japan Meteorological Agency*; and the *European Early Warning and Response System* (EWS)[100], which monitors public health conditions in the EU and is coordinated by the *European Centre for Disease Control* (ECDC). The EWS was updated under the Decision 1082/2013/EU and was originally used to exchange notifications about outbreaks and epidemics between Member States, updated in 2014 to enable risk management regarding biological, chemical and environmental hazards.

While the EWS has been developed as a simple web application, lacking automated monitoring, i.e., relying on manual activities to identify and send warnings, IPAWS and PTWS use sensor networks to collect in-situ and processed data from the physical world. Furthermore, IPAWS acts as a central platform of the EAS, the USA national EWS, which collects and integrates data from sensor networks and several existing alert systems. IPAWS combines existing EWSs to form a bigger EWS. Therefore, IPAWS is classified as a multi-hazard EWS (MHEWS), because it monitors different types of hazardous events and/or impacts of these events, considering their local and temporal relations, either in cascade or cumulatively over time, along with the potential interrelated effects. An MHEWS involves multiple disciplines for accurate comprehension of multiple hazardous events that can lead to disaster situations [30].

The importance of MHEWS is reinforced by the UN agenda for DRR, i.e., the Sendai framework [29], which defines a set of global targets to be achieved by 2030. Global target G refers to “substantially increase the availability of and access to multi-hazard early warning systems and disaster risk information and assessments to the people by 2030”. To achieve this goal, risk identification should be improved through a holistic approach for EWS integration. Sendai emphasizes the need to share disaster risk information and to increase data availability towards a multi-hazard integrated global warning and response system, which must be tailored to user needs across distinct and overlapping domains [30]. At the EU level, the low level of information sharing within and across organizations is the major problem to integrate EWSs, which led to the EU initiative to integrate all official EWSs at the global level [31].

2.2 Interoperability

Interoperability of EWSs is a crucial characteristic towards this holistic EWS integration approach, especially if big data generated from IoT technologies are

¹⁰ <https://ptwc.weather.gov/>

considered to improve EWS efficiency and effectiveness [14]. This potential can only be exploited if interoperability challenges are addressed. Interoperability is the “ability of two or more systems or components to exchange information and to use the information that has been exchanged” [107]. A broader definition considers human, social, political, and organizational factors that impact all components of an integrated system [108, 109].

2.2.1 Classification and Standards

Several interoperability classifications exist [27] and, at the application level, the interoperability aspects to consider are:

- 1) Coding and formatting: the binary encoding of the messages/streams that carry data (technological interoperability), i.e., the packaging of data in a message (syntactic interoperability);
- 2) Interpretation: the assignment of meaning to the data (semantic interoperability); and
- 3) Dialogue: the process synchronization for the exchange of messages (process interoperability).

Coding and formatting are covered by existing standards, but interpretation is only partially covered [110]. Syntactic interoperability refers to data format standards and communication protocols such as, e.g., the OASIS Emergency Data Exchange Language (EDXL) and the HL7 FHIR, which aim at enabling systems to communicate to each other. These standards enable ICT-based systems to communicate to each other through common data formats. Common data standardization is not enough to guarantee that the systems, and more importantly, their (direct or indirect) users, share the meaning of the messages.

For example, consider a patient tracking system that exchange messages between an *Electronic Health Record* (EHR) system and an *Enterprise Resources Planning* (ERP) system of a hospital, which provide services following the message structure standards of EDXL. The standard for tracking patients, coined EDXL-TEP, provides the property “vehicle kind” from the “transport type” entity within the reference model. If the first system generates a message with the value “car” referring to an “ambulance” and the other two systems follow a different definition for the “car” term (such as a personal car), then they will not share the same semantics. Because of this, the decision making can be affected, leading to erroneous procedures. In this example, the hospital is waiting for the patient in an ambulance and, therefore, expects that the first medical procedures are already taken. This is an example of a semantic interoperability problem.

Semantic interoperability refers to the study of meanings, the ability to automatically interpret shared data meaningfully and accurately according to agreed-upon semantics, i.e., a common information exchange reference model. Semantic interoperability focuses on terminology and deals with human interpretation in an unambiguously way, ensuring that the understanding of the information is the same for senders and receivers and “that the requester and the provider have a common understanding of the ‘meanings’ of the requested services and data” [111]. Semantic interoperability is crucial for EWS integration because it ensures the shared meaning of terms, dealing with human interpretation of heterogeneous information, avoiding misunderstandings when communicating about delicate issues. Semantic interoperability also enables seamless integration of different data sources and leverages risk identification.

Semantic interoperability is a topic covered by several researches [25, 27, 38, 112-122], such as methodologies and maturity levels [27]. The Information Exchange Packaging Policy Vocabulary (IEPPV), an Object Management Group (OMG) formal specification [123] introduced a methodology to achieve semantic interoperability by: (1) Apply structure and syntax of information exchange by reusing or extending standards from recognized standardization institutes, such as W3C, OASIS, OMG, NIEM, HL7, among others; (2) Specify the rules for information assembly and processing in an unambiguous way. Such rules should contain patterns for the aggregation of data and conversion transformations to shared agreement standards. These rules include parsing (transforming) messages into their data elements; and (3) Capture and hold the rules governing the operational interfaces. Usually, items 2 and 3 are aspects of the application lifecycle that are not properly explored by traditional development methods and technologies, in which the translation of policies and constraints to executable rules (encoded software) is typically based on textual requirements.

Figure 2.5. The European Interoperability Framework (EIF) [124]

In the European context, the *Interoperability Solutions for European Public Administrations* (ISA) program (along with its extension¹¹) should be highlighted, which produced the European Interoperability Framework (EIF). Since 2017 the

¹¹ https://ec.europa.eu/isa2/home_en

new EIF [108, 109], illustrated in Figure 2.5, was adopted by the EC as guidelines on enabling interoperability for digital public services, offering recommendations to improve the governance of interoperability activities. These recommendations are classified according to the four layers of the framework:

- (1) Legal: refers to the required actions to ensure organizations to work together, considering their different locations and, therefore, under different legal frameworks and policies, e.g., data protection laws. Partially addresses the dialogue aspect.
- (2) Organizational: refers to the required actions to align business processes, services and responsibilities, focusing on organizational relationships between service providers and consumers. Partially addresses the dialogue aspect.
- (3) Semantic: refers to both semantic and syntactic aspects for information exchange, ensuring the precise format and meaning of exchanged data, emphasizing metadata, master data and reference data management. Addresses the interpretation aspect.
- (4) Technical: refers to interface specifications and communication protocols, emphasizing the role of formal technical specifications deal with the integration of legacy systems. Addresses the coding and formatting aspects.

Among the twelve interoperability principles of EIF and their respective recommendations to improve interoperability of information systems, we highlight: (1) Openness: refers to the Open Government Data practices, ensuring that data are openly published when no restrictions are applicable, leveraging on the Open Source movement and open specifications; (2) Reusability: refers to reuse and share of data and solutions for higher collaboration, considering privacy and confidentiality restrictions; (3) Technological neutrality and data portability: refers to the avoidance of technological dependencies, ensuring data portability, being able to replace technologies without depending on specific manufacturers; (4) User-centricity: refers to the orientation to the user requirements, considering multiple communication channels, with single point of contact, involving users in the system lifecycle; (5) Security and privacy: refers to secure data and processes, covering privacy of personal information; and (6) Assessment of effectiveness and efficiency: refers to the evaluation process of the solutions, checking the adequacy of interoperability solutions.

Regarding interoperability to support collaboration, which is closely related to the dialogue aspect covered by the organizational layer of EIF, we should mention the concept of pragmatic interoperability [125]. “Pragmatic interoperability is the compatibility between the intended versus the actual effect of message exchange”

[126]. So, if pragmatic interoperability refers to the assertion of the results of executed processes w.r.t. their specification, then, pragmatic interoperability is the validation measurement that can be used to assess the dialogue aspect within the organizational interoperability layer.

2.2.2 The FAIR Data Principles

While EIF provides a reference architecture and higher-level recommendations for developing interoperability solutions, the *Findable, Accessible, Interoperable and Reusable* (FAIR) principles provide a set of best practices to achieve higher data reusability. Significant effort is required to find, make sense and reuse data. The FAIR data principles were elaborated to tackle this problem, describing a minimal set of requirements for data management and stewardship. By adhering to the FAIR data principles, the data produced by a solution can be findable, retrievable, possibly shared and reused and, above all, properly preserved [62]. Making an EWS adherent to the FAIR data principles allows general-purpose open technologies to use “machines to act on the data, which can dramatically improve the efficiency of disaster responses”, as concluded by the research on public health emergency to detect Zika outbreaks [127]. Furthermore, the application of FAIR in healthcare showed that these principles can provide a boost to big data-type applications that require the integration of data coming from different sources, achieving “interoperability without the need to all speak exactly the same language” [128].

The FAIR data principles have been rapidly adopted by several institutions worldwide [121], including the EU, G7 and G20, leveraged by the GO FAIR initiative [129]. For example, all projects funded by the EC under the H2020 scheme are obliged to apply a research data management plan based on the FAIR data principles. GO FAIR aims at implementing a bottom-up approach for the implementation of the European Open Science Cloud (EOSC)¹², which represents a vision to strengthen EU’s competitiveness in digital technologies, enabling users to benefit from data-driven science. Usually, there are interpretations that lead to think that FAIR and Open Science are the same things. Although both philosophies are related to data openness, it is not necessary to be open to be FAIR, even though this is desired in academic and scientific circles [127]. Furthermore, measuring how a digital resource adheres to the FAIR data principles, i.e., the *FAIRness* level of the resource, is not trivial [130]. The GO FAIR metrics initiative¹³ aims at defining a set of basic metrics to measure each principle, as well as an approach for the definition of new metrics according to specificities of the application domain. Therefore, the FAIR metrics should be executed to measure the level of *FAIRness* of a digital object.

¹² <https://eosc-portal.eu/>

¹³ <http://fairmetrics.org/>

Table 2.1. The FAIR data principles

FAIR	Data Principle	How to apply
F1	(Meta)data are assigned a globally unique and persistent identifier	Metadata should have a unique identifier for each dataset, e.g., persistent identifier (PID) or equivalent (DOI, ARK, RRID, PID) to unambiguously refer that resource. A PID denotes the “same referent indefinitely, and is never re-assigned to another entity”.
F2	Data are described with rich metadata (defined by R1 below)	Datasets must have rich metadata that, once indexed by a search engine, these metadata can help the user to find the data even if the user does not know the data identifier(s).
F3	Metadata clearly and explicitly include the identifier of the data it describes	Since we cannot assume that the data and their metadata are always together (for example, when the metadata is indexed by a search engine and therefore resides in a different platform), the association between data and metadata must be made by the inclusion of the metadata identifier in the data.
F4	(Meta)data are registered or indexed in a searchable resource	In order to enable data to be findable, their metadata must be indexed by search engines which, in turn, allow users to find the data through the metadata elements.
A1	(Meta)data are retrievable by their identifier using a standardized communications protocol	With the data or metadata identifier, users should be able to retrieve the data by means of a standardized communication protocol.
A1.1	The protocol is open, free, and universally implementable	Regardless of data and metadata licensing, the communication protocol used to provide access to them must be open, free and capable of being implemented by any interested party.
A1.2	The protocol allows for an authentication and authorization procedure, where necessary	When necessary, depending on data and metadata access restrictions, a mechanism for authentication and authorization should be permitted by the communication protocol.
A2	Metadata are accessible, even when the data are no longer available	A preservation strategy for data and metadata should exist. Minimally, metadata must always be accessible, enabling the creation of indexes for current and past datasets.
I1	(Meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation	In order to represent data and metadata we must use knowledge representation languages that are formal, accessible and widely applicable, such as RDF.
I2	(Meta)data use vocabularies that follow FAIR principles	Data and metadata should have references to the vocabularies that describe their conceptualization, ensuring that these vocabularies also follow these FAIR data principles.
I3	(Meta)data include qualified references to other (meta)data	Provide proper reference to datasets, enabling that datasets generated from other datasets to be linked. Enable the semantic link between these datasets by means of the adopted standards.
R1	Meta(data) are richly described with a plurality of accurate and relevant attributes	Provide metadata that enables potential users to validate whether it is possible to reuse data or metadata, and also, if they suit to their needs.
R1.1	(Meta)data are released with a clear and accessible data usage license	Data and metadata administrator should be able to explicitly define who can access them and under what conditions. This information are defined through the data licenses used.
R1.2	(Meta)data are associated with detailed provenance	Enable the specification of data provenance to allow the user to analyze the usefulness of the data or metadata and to give the credit to who produced, maintained, or edited the data. This specification should contain the data lineage (process of obtaining the data), whether data are raw or processed, data versions and the particularities or limitations about the data, e.g., the dataset creation date, in which laboratory conditions, who prepared the data, parameter settings, name and version of the software used.
R1.3	(Meta)data meet domain-relevant community standards	Meet specific community standards, following archiving and sharing best practices.

The FAIR data principles are organized in four categories, where each one represents a letter and contains a set of requirements based on best practices from data management and stewardship, applied to both data and metadata. Table 2.1 lists all FAIR data principles and explains how each principle should be applied. A core aspect of the principles is the reuse of standardization initiatives, specifically pointed out by the interoperability principles. *I1* can be addressed by using the core language of semantic technologies and linked data, the *Resource Description Framework* (RDF)¹⁴, which is a W3C standard. *I2* can be addressed by adopting ontologies that are well specified and documented.

Finally, the *I3* principle can be achieved by either adopting standardized vocabularies that are available as RDF, such as the W3C *Semantic Sensor Network and Sensor Observation Sampling Actuator* (SSN/SOSA) ontology¹⁵, or by creating alignments to standards. In the context of e-Health, the Interoperability Standards Advisory¹⁶ provides up-to-date reports regarding the best available standards and specifications that cover the main vocabularies. Currently, the *FAIRsharing.org* provides pointers to ontologies from different disciplines. *I3* principle also highlights the adoption of standards to describe datasets, such as the W3C *Data Catalog Vocabulary* (DCAT)¹⁷.

The main components of the FAIR data ecosystem are data policies, data management plans, identifier mechanisms, standards and data repositories [127]. GO FAIR provides a set of technology specifications and implementations for these components, classified as: (1) GO FAIR metadata; (2) FAIRification process; (3) GO FAIR tools; and (4) GO FAIR metrics.

The GO FAIR metadata guides the use of existing standards and vocabularies, such as OAI-PMH, *Dublin Core* (DC) and W3C DCAT, for metadata specification for data repositories. Therefore, the GO FAIR metadata specification aims to combine existing interoperability standards to meet the FAIR objectives. This specification also guides the minimal metadata needed towards dataset findability (F) and it specifies how to declare a data license (A) in order to be reusable (R). The *FAIRification* process specifies the steps required to transform an existing dataset to FAIR data, which is aligned to the ontology-driven conceptual modelling approach for domain modelling. The entire *FAIRification* process is supported by a set of open source tools, coined as *Data FAIRport*. These tools include mechanisms to create, publish, find and annotate data. In particular, the *FAIR Data Point* (FDP)¹⁸ implements the GO FAIR metadata, enabling the publication and access of both

¹⁴ <https://www.w3.org/RDF/>

¹⁵ <https://www.w3.org/TR/vocab-ssn/>

¹⁶ <https://www.healthit.gov/isa/>

¹⁷ <https://www.w3.org/TR/vocab-dcat/>

¹⁸ <https://github.com/DTL-FAIRData/FAIRDataPoint/wiki/FAIR-Data-Point-Specification>

data and metadata. Therefore, the FDP is a specification that allows data owners to expose their datasets in a data repository according to the FAIR data principles.

A data repository is a database infrastructure, which can encompass several database instances, enabling data collection, management and datasets storage. The main end users of data repositories are researchers that need to analyze, share and report data. Data repositories are usually designed to store and manage the outputs of research after the data are produced and the results are published. Therefore, they play a major role on allowing the reusability of data, for which interoperability is required.

The most popular protocol for metadata harvesting is the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), which is based on the Dublin Core (DC) and sets a basis for research data interoperability, while the Open Archival Information System (ISO OAI) is a well-known organization that promotes metadata interchange and dissemination. A comparison study with OAI experts showed that several data repositories were inventoried and six were validated according to OAI-PMH: EUDAT B2share, CKAN, Zenodo, Dspace, Eprints and Figshare [131]. Only CKAN is not natively compliant to OAI-PMH, but all of them provide APIs for exposing metadata records. Some of these data repositories have support for RDF to improve data interoperability. For example, RDF for CKAN is provided via plugins that allow it to expose and consume metadata according to the DCAT [132]. There are similar solutions for Dspace and Eprints. This study concluded that selecting a data repository is a challenging task and should be based on technical choices. According to these FAIR recommendations, a research data repository requires five main functions: (1) to provide general information about its service; (2) to provide a data policy; (3) to comply with legal aspects, e.g., open/restricted/closed access, terms of use and licenses; (4) to adhere to technical standards, using a persistent identifier (PID) system, thus, assuring data persistence, uniqueness and citation; and (5) to be certified by a quality standard or supported by a repository standard, e.g., reviewed by the re3data.org team. The *FAIRsharing* website¹⁹ guides the end users, both data producers and consumers, to discover data repositories that address these functions.

The FDP specification addresses the five functions described above, also allowing data owners to gather access metrics about the offered data and metadata, as well as allowing the interaction for both humans through a GUI and software agents through an API. The FDP provides both API specification and an implementation, which is optimized for machine discoverability and processing. The FDP provides metadata at five complementary layers: (1) FDP Metadata; (2)

¹⁹ <https://fairsharing.org/>

Data Catalog Metadata; (3) Dataset Metadata; (4) Distribution Metadata; (5) Data Record Metadata.

The FDP implementation is optimized for machine discoverability and processing. (Meta)data access starts with the FDP URL, which resolves to metadata that describes the FDP and supports further exploration of the FDP by iterating through the different metadata levels. For example, the metadata include a link to the next FDP level (catalogs), which in turn links to the dataset-level descriptions. The FDP metadata specification is based on existing standards and vocabularies (OAI-PMH, DC, DCAT and LDP). Therefore, the FDP specification aims to combine existing standards to meet the FAIR objectives. The FDP specification gives a recommendation for the minimal metadata needed towards dataset findability (F) and it specifies how to declare a data license in order to be reusable (R). The Personal Health Train project is applying FDP infrastructures to obtain information from the supply chain of patient process in the scenario of healthcare emergency services, where emergency simulations are executed [128]. Each interaction of the involved systems with the process, e.g., the ambulance registration system transporting a patient from his home to a hospital, includes the publication of the data in a FDP repository.

2.2.3 Model-Driven Software Engineering

The EIF reference recommendations along with the FAIR data principles aim at improving interoperability in a general sense, guiding the use of domain-specific standards, as vocabularies and data models. However, they do not describe how to develop or extend such standards so they can play the role of bridges to enable syntactic, semantic and process interoperability [133]. The manual implementation of bridges by directly connecting APIs or interfaces is error-prone and time consuming because they happen at a low-level, ignoring semantic interoperability. Model-driven software engineering (MDSE or MDE) can cover this gap because it improves the interoperability of software from a higher abstraction level [134]. Approaches that aim at supporting the definition of bridges through the application of MDE techniques are termed as model-driven interoperability (MDI) solutions. MDI solutions explicit the metamodels of the involved systems (or components of the systems) and align them by matching the related elements through MDE transformations. MDI separates the syntactic mapping from the semantic mapping [133].

The generic interoperability bridge approach is illustrated in Figure 2.6, showing the concept of *projector* as the software that performs syntactic mapping, being able to transform a model to another model. In MDE, a model is an abstraction and simplification of portions of reality, which is necessary to understand, communicate and reason. A model represents something from a

specific point of view and with a specific purpose. Models abstract from irrelevant details and enable more efficient analysis of past, current and future states of a universe of discourse, which is the part of the observed world being modelled, namely the domain [135]. A *projector* is a parser and can be either an injector or an extractor, where both use dedicated formats. The former is able to inject the input file a_i into model a_i , and the latter is able to generate output file b_i from model b_i . Projectors can be implemented in general-purpose languages or model-to-text/text-to-model transformation languages, such as Extensible Stylesheet Language Transformations (XSLT), ATLAS Transformation Language (ATL) and Query/View/Transformations (QVT) [136].

Figure 2.6. MDE transformations for syntactic and semantic mappings [133]

The semantic mapping can be implemented as a model-to-model transformation, representing the equivalences between the domain concepts of systems A and B . Therefore, *projectors* target syntactic interoperability and *transformations* target semantic interoperability. Transformations can be combined in transformation chains, thus, being able to decompose the problem in parts, which provides higher extensibility, reusability and maintainability of bridges.

In MDE, the transformation specification structure can be declarative, where relations among model elements without an explicit execution order are specified, or imperative, i.e., the sequence of steps that produce the result model is specified.

A hybrid approach is commonly used, mixing benefits from both declarative and imperative. ATL is an example of hybrid transformation language, using declarative expressions with the Object Constraint Language (OCL) and imperative expressions to describe rules as dedicated blocks composed of sequence of commands.

In addition to the MDE transformations, the MDE approach can be used to develop distributed systems following the Service-Oriented Architecture (SOA) [137], thus, leveraged by the principles of technology independence and separation of concerns. MDE is also suitable whenever different levels of abstractions are required, such as to support traditional large-scale enterprise software engineering, which splits the software development process in specification phase (modeling/designing) and implementation phase (code realization). Furthermore, MDE does not rely in a specific methodology or technology, and the application of MDE can be adequate for both classical waterfall, i.e., linear sequential, or agile methods, i.e., iterative and incremental [138].

Graphic modeling languages and notations support the software specification both for structural and behavioral aspects. For example, the Unified Modeling Language (UML) class diagrams can be used to represent structural aspects of the domain, while the UML activity diagrams can be used to represent the communication among software components, and the Business Process Model Notation (BPMN) can be used to represent the interaction of business processes with software interfaces. Usually, Domain-specific languages (DSL) are defined with the Meta Object Facility (MOF), or a MOF-like language like Ecore, which is aligned with the UML class diagram. Therefore, UML class diagrams are commonly used for DSL conceptual modeling.

The core UML structural diagram elements are *Class*, *Relationship* and *Attributes*. The higher-level *Relationship* category is the *Association*, which can be binary or n-ary. *Aggregation* and *Composition* are types of *Binary Association*, both representing a type of mereology, where a class is the "whole" object of another class (part). *Aggregation* is a relationship where the instance of the part can exist independently of the existence of the whole instance, e.g., a car is an aggregation of mechanical pieces. *Composition* is a relationship where the part cannot exist independently of the whole, e.g., a house as a whole and its rooms as parts, where a room cannot exist without a house.

For process modelling, the BPMN has a widespread adoption by academy and industry, being supported by several MDE tools. BPMN 2.0 is an OMG standard²⁰ and some software development methodologies prescribe the use of BPMN for processes documentation (as-is and to-be) and service compositions, where *Business Process Execution Language* (BPEL) specifications can be generated from BPMN

²⁰ <https://www.omg.org/spec/BPMN/2.0/>

models through MDE transformations. BPMN to BPEL approach is frequently used in the literature to tackle interoperability of collaborative networks, i.e., process interoperability [113]. BPMN is also commonly used by process simulation and mining approaches [139]. In this thesis we give emphasis to the event notations of BPMN, in particular the *Message* event, which is illustrated in Figure 2.7. In communication theory, a message is the most basic element for the exchange of information, which is a conceptualization adopted by BPMN that uses the *Message* element to depict the information content in a communication between two *Participants* (defined by a *PartnerRole* or a *PartnerEntity*). Causality is partially covered by these notations through the *triggers* element. Since a message “represents the content of a communication between two Participants”, *Message* events are crucial to achieve process interoperability.

Figure 2.7. Message Event Types in BPMN 2.0²⁰

The appropriate use of the stereotypes and notations of these languages provide higher expressiveness to domain-specific models towards common understating of processes’ characteristics, which, ultimately, can lead to the development of higher interoperable systems. Therefore, conceptual modelling of structural and behavioral aspects is a core practice to achieve higher interoperability, from the specification phase to the implementation. Although the MDE practice covers (partially) how to address semantic interoperability, the emphasis of the approaches are in the lexicon level, i.e., the terminology (syntactic), and not the conceptual level (semantic), albeit there is significant research on applying semantic technologies in MDE [22, 140, 141]. Semantic technologies are further explained.

2.3 Semantic Technologies

Semantic interoperability is a topic covered by extensive research since the 90s, leveraged by the *Semantic Web* in the 2000s, which was envisioned as a way to enable machines to understand data on the Web. Towards this goal, the *World Wide Web Consortium* (W3C) created a series of standards to promote common data formats and exchange protocols, extending the *World Wide Web*. One of the most prominent standards for the *Semantic Web* is the *Resource Description Framework* (RDF), a language that is cable of implementing *Linked Data*, which is the method of publishing structured data in a way that data can be interlinked and queried. The original Semantic Web stack includes RDF, SPARQL and other W3C standards, such as XML, *RDF Schema* (RDFS) and the *Ontology Web Language* (OWL). These

semantic technologies are classified as one of the main emerging technologies (Gartner hype cycle 2017), which will be in the “plateau of productivity” (widely adopted by the market) within 5 to 10 years.

In this thesis, we refer to *semantic technologies* as a broader concept, not only with the technologies of the semantic web stack, but also including methodologies and standards used to address semantic interoperability from diverse perspectives. Following, we summarize the *semantic technologies* commonly used by the *Formal Ontology* community, focusing on foundational ontologies, the role of ontology-driven conceptual modelling for MDE and how to integrate semantic models through ontology alignments and semantic translations.

2.3.1 Formal Ontology

The *Linked Data* and *Semantic Web* communities refer to an ontology as any model represented with RDF. However, the concept of ontology is broader and represents a major branch of metaphysics, from Aristotle theory of being, which studies aspects such as existence, reality and basic categories of being and their relations. This thesis adopts the term *semantic model* for a model represented with RDF and the term *ontology* for a model grounded in theories that provide a high-level category system of the most fundamental concepts and their relations [142]. Therefore, an ontology may be implemented with different technologies, albeit the use of RDF is common [124].

Figure 2.8. The Semiotic Triangle: good and bad ontologies [124]

The goal of an ontology is to represent entities, relationships, properties, rules and restrictions of the observed domain. It can be considered formal when it can be automatically processed by machine, enabling automated reasoning by the semantics described with a formal language. Ontology is a tool to address the problem of construct ambiguity and overloading from the triangle of meaning (or triangle of reference or semiotic triangle) [143], often called as *Ogden and Richards triangle*, illustrated in Figure 2.9. The triangle shows the relationships between the conceptual level (knowledge interpretation) with the symbol in a language (symbol, e.g. term) and with the real world phenomena that it refers to (referent).

A concept is described by a term, which may be overloaded in the language level (same term representing different concepts), but the manifestation of the construct refers to only one of the concepts [142].

Figure 2.8 also illustrates how the observed reality is perceived through presentation patterns, thus, perception here has the same meaning as in the *Endlsey's SA* theory, while comprehension is achieved through conceptualization. The representation of the concept is described with a language through a process called ontological commitment, i.e., when terms are committed to describe specific concepts. Good ontologies should cover the intended models that the language allows, while a bad ontology covers non-intended models. Therefore, while a good ontology can improve semantic interoperability, a bad ontology can cause the contrary effect, allowing wrong representations [144].

To deal with the problem of overloaded terminology and ambiguity, the formal ontology research recommends a rigorous definition of the concepts by classifying the terms in a high-level categorization system, which is called a foundational ontology. This process is called ontological analysis. As an example of a problem generated by the lack of ontological analysis, the term “disaster” is often referred to as an event and a situation (e.g., UISDR definition), which is an overloaded interpretation since event and situation are different concepts.

A foundational ontology is based on theories from logic, philosophy, linguistics and cognitive psychology and is able to formalize a high-level category system of the most fundamental concepts and their relations (e.g., kind, role, event) through axioms. A foundational ontology can be materialized through an ontological language that is used to design domain ontologies. A domain ontology is called well-founded when it is based on a foundational ontology. This means the ontological primitives used in the elaboration of the domain ontology are well-founded and therefore are deemed adequate to properly capture and formalize some subset of concepts used by a community to communicate about a domain of interest. As a special type of conceptual model, a domain ontology is intended to be used by humans for purposes of understanding and communication [145]. Domain ontologies are useful for various purposes, which include: (1) knowledge-based applications [142]; (2) semantic interoperability [146]; and (3) definition of shared vocabulary as community standard [147].

Some foundational ontologies were developed in the past 30 years, such as the Basic Formal Ontology (BFO)²¹, the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) and the Unified Foundational Ontology (UFO). We emphasize the DOLCE Ultralite (DUL)²² ontology, which is a lightweight version of DOLCE (DOLCE Lite-Plus) integrated with the Descriptions and

²¹ <http://basic-formal-ontology.org/>

²² <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl>

Situations ontology (DnS). DUL was widely used as a foundational ontology in the literature, grounding the W3C SSN/SOSA and the Smart Appliances REFERENCE (SAREF) ontologies, and used for representing emergency events and situation awareness [57]. DnS offers patterns for the ontological formalization of contextual situation to support the design of core ontologies [148, 149]. DOLCE was also applied in the context of ontological analysis of BPMN, emphasizing the ontological commitments behind event and activity stereotypes [150]. The importance of ontological analysis in the context of software engineering, particularly through ontology-driven conceptual modelling, is further exploited in this thesis.

2.3.2 Ontology-Driven Conceptual Modelling

When developing an ontology to be used as an artefact for software development, the formal ontology research emphasizes the need of formal specification approaches during the conceptual modelling phase within software engineering [151, 152]. Ontology-Driven Conceptual Modelling (ODCM) [153] leverages traditional conceptual modelling by “employing ontological theories in philosophical sense (i.e., axiomatic theories of categories and their ties) for developing languages, methodologies and tools for the discipline” [154]. In particular, ODCM was successfully applied in many domain ontology engineering initiatives using UFO, gaining attention from the formal ontology community [155].

Figure 2.9. Main categories of UFO-A: identity and rigidity principles [153]

UFO, initially inspired by DOLCE and GFO (among others) makes a fundamental distinction between *Particular* and *Universal*. The former is a category that corresponds to individual things, whilst the latter stands for patterns of

features that multiple individuals have. A *Particular* (e.g., “John”, “ISCRAM’15”), is an instance of a *Universal* (e.g., *Person*, *Conference*). *Particulars* are further categorized as *Endurants* and *Perdurants*. *Endurants* (covered by UFO-A) represent “wholly present” entities whenever they are present, i.e., they are in time, (e.g., a house, a person). *Events* (synonymous to *Perdurants*, covered by UFO-B [156]), in contrast, are particulars composed of temporal parts, i.e., they happen in time, in the sense that they extend in time accumulating temporal parts (e.g., a football match, an earthquake) [156]. *Events* are possible transformations from a portion of reality to another, i.e., they may change reality by changing the state-of-affairs from one situation to another.

The *Endurant* category is further refined into *Substantial* and *Moment*, illustrated in Figure 2.10 (top and bottom, respectively). *Substantials* are existentially independent *Particulars*, whilst *Moments* are *Particulars* that can only exist linked to other *Particulars*. Examples of *Substantials* include “John” and his house; examples of moments include a marriage and the color of an object. *Moments* are existentially dependent on other individuals. *Modes* are moments that depend on a single individual, e.g., an intention, whilst *Relators* are dependent on a plurality of individuals, e.g., an agreement. All *Substantials* follow an identity principle, a sort of function that allow us to count individuals and differentiate them. A *Sortal* is a type of *Substantial* whose instances all follow the same identity principle (e.g., person, rock), whilst a *Mixin* characterizes individuals that follow different principles (e.g., client, agent). A *Kind* characterizes essential properties of *Particulars*, while a *Role* characterizes contingent properties and is relationally dependent, i.e., its instantiation is determined by a relational property of the individual. The *Mixins* are specialized into *Category*, those that generalizes *Kinds* and other *Categories*, and *RoleMixin* that generalizes *Roles* from different *Kinds*.

Figure 2.10.
UFO: relations
between *Situation*
and *Event* [156]

Perdurants (or *Events*) are represented by possible transformations from one portion of the reality to another. An *Event* can affect the reality by changing its characteristics from a pre *Situation* to a post *Situation*. The *Event Mereology* classifies an *Event* as *Atomic* or *Complex Event*, having the first denoting an *Event* that begins (*begin-point*) and ends (*end-point*) at the same *Time Point*, being part-of a *Complex*

Event. An *Event* is a concept which is existentially dependent on the *Participation* of *Substantial* classes and it is composed by them. UFO-B also addresses *Time Interval Relation* between two *Events* (source and target) as a *Formal Relation* that is represented by each Allen's operator, such as *during*, *before*, *meets*, *overlaps*, *starts*, *finishes* and *equals*.

The *UFO Situation* concept reflects the definition from *Barwise*, as a snapshot from a part of the world in a specific *Time Point*, involving intrinsic aspects, such as *Endurants* participations and even other *Situations*. Once they are representations of the changes in reality, they have two relations to *Events*: (i) a *Situation* can *trigger* a set of *Events*, and (ii) a *Situation* can be *brought about* a set of *Events*. These relations are illustrated in Figure 2.10 and the axioms were formalized in [156]. Therefore, it is possible to model how a change in the world (an *Event*) makes the state-of-affairs change from one pre-situation (the *Situation* which *triggers* the *Event*) to a post-situation (a *Situation brought about* the *Event*).

UFO also includes the main categories that allow the representation of different types of relations (*Relation Universal*). For example, the *componentOf* is a relation used to describe mereology that is similar to the composition stereotype from UML. Similar to GFO and GOL, the main UFO categories are used for MDE through the ontological language *OntoUML*, which is a UML profile to represent a modelling language that differs from epistemological and logic languages because of the inherited ontological assumptions from UFO [155]. For example, *OntoUML* provides the *relator* stereotype (a *Moment*), which can be used to solve the cardinality ambiguity problem of UML [157]. *OntoUML*'s rich syntax guides users throughout the modeling activity, which fosters domain knowledge discovery. Therefore, *OntoUML* is a language designed to support the creation of well-founded domain ontologies.

OntoUML was applied in a variety of domains, such as e-Health, and includes a semantic MDE methodology in which models are ontologically well-founded. It addresses model assessment through a lightweight formal method, where the formal semantics of *OntoUML* specifications is transformed to the Alloy logic language, which supports visual simulation of multiple instances (worlds) of a model through the Alloy analyzer [158]. Furthermore, ODCM with *OntoUML* includes design patterns, temporal constraints with OCL and mappings to languages such as UML, OWL and SWRL. Besides these capabilities, *OntoUML* has also mature tooling support with the Menthor designer²³ for model construction, verbalization, code generation and validation, incorporating the aforementioned efforts from its precursor, named *OntoUML Lightweight Editor* (OLED).

²³ <https://github.com/MenthorTools/menthor-editor>

Figure 2.11.
OntoEmergePlan: a
 well-founded
 emergency plan
 ontology [71]

OntoUML is a mature language and has been used to describe core ontologies, pattern languages and DSLs [155], applied in several application domains, such as emergency planning covering organizations, events and risks [159]. For example, *OntoEmergePlan* [71] is a well-founded domain ontology developed with OntoUML (*Menthor* tool) that considers the analysis of the literature about emergency procedures. The main goal of *OntoEmergePlan* is to support the generation of emergency plans, serving as a reference model for common understanding and communication. It brings definitions to respond competency questions such as what is an emergency, emergency types, processes, plans, plans' compositions, installations and evacuation routes, involved business activities, types of risks and hazards, among others. *OntoEmerge* is further reused in this thesis.

The main concepts required for plans generation are illustrated in Figure 2.11. stereotypes correspond to UFO categories. For example, *OE Process* is classified as *Mode*, which is a special type of intention, the *Organization Emergency* (OE). *Organizations* are responsible for protecting their workers and their property from hazards. Thus, a self-protection process must be established by defining a set of activities needed to address the hazards. The emergency plan, condensed into a self-protection plan called *OE Process Definition Document*, is essential to guide to the necessary protection of workers. The plans may include identification of hazards, responsible, resources to carry these actions, among others. The concept of *Installation*, which refers to an establishment that corresponds to the total area

under control of a *Holder* that develops a *Business Activity*. An *Environment* (such as forest, city, farm) is exposed to *Hazardous Event* that can cause *Impact* (such as destruction, pollution, death). The *Environment* is composed of *Component* (such as river, building) and an *Installation* is a *Physical Component*. The *Holder* can be a *Person* and/or *Organization*, playing the role of *Businessman*, i.e., an entrepreneur that develops a *Business Activity*, or *Owner* that has *Possession* of an *Installation*. The same *Holder* may play both roles simultaneously, illustrated by the annotation [overlapping, complete]. An *Installation* must have a deployed self-protection plan.

After designing the ontology through the ODCM practice, the common approach with OntoUML is to generate code from the conceptual models by means of MDE transformations, transforming OntoUML models in RDF serializations.

2.3.3 RDF Implementation

RDF is a knowledge representation language for artificial intelligence (AI), which helps computer systems to understand data and solve complex tasks. RDF can be used to address the interoperability principles of FAIR because it is the current popular solution to the knowledge-sharing problem and, therefore, satisfies the FAIR principle *I1*. The basic concept of the RDF language is the representation of triples, i.e., *subject–predicate–object*, which are statements that compose a labeled directed multi-graph. The basic elements of RDF are: (1) *classes* (or instances of classes), which can be used as *subject* and/or *object*; (2) *object properties*, which can be used as *predicate* when the *object* is a *class* (or instance of a class); (3) *data properties*, which can be used as *predicate* when the *object* is a *data type*; and (4) *data types*, which are primitive types such as integer, string and float, usually defined with the *XML Schema Datatypes* (XSD)²⁴.

RDF can be serialized in diverse standards, such as XML, Turtle (TTL) and JSON for Linked Data (JSON-LD)²⁵. JSON-LD is a lightweight syntax to serialize RDF, structured way of using JSON that provides semantics to JSON data at web-scale, conceived to introduce Linked Data in web-based programming environments. JSON-LD is a W3C standard recommended by schema.org and Google, gaining popularity to semantically enrich data embedded in HTML pages by adding the script type tag *application/ld+json*, which enables search engines to consider these data in their search mechanisms. JSON-LD introduces features as the *Internationalized Resource Identifier* (IRI), annotations with different languages and graphs expression.

The core part of JSON-LD is the set of syntax tokens that are prefixed with “@”, e.g., *@context*, *@id*, *@value*, *@type*, *@graph*, among others. JSON-LD can be serialized with different styles, such as compacted and flattened, were the

²⁴ <https://www.w3.org/TR/swbp-xsch-datatypes/>

²⁵ <https://json-ld.org/>

former is recommended for smaller messages while the latter is recommended for humans to read. Besides JSON-LD and TTL, there are ongoing serialization initiatives that aim at providing compacter messages, such as the RDF-HDT format²⁶, which applies a compact binary storage format, but requires more processing to serialize and de-serialize the messages.

RDF can be used to represent both data and metadata and can be stored in *triplestores*, which is basic concept behind the FAIR Data Point data repository specifications. The most popular *triplestores* are *Virtuoso*²⁷ (open source) and the recent *Amazon Neptune*²⁸ (commercial), which share a number of characteristics²⁹, such as semantic query capabilities with the SPARQL language³⁰. SPARQL is the semantic query language standardized by W3C.

In particular, OWL adds predicates on top of RDF and is commonly used for ontology implementation. *Lightweight semantic models* use only classes and their relations, while *heavyweight semantic models* additionally use axioms to narrow the semantic interpretation [14]. Usually, a *lightweight ontology* is a semantic model described with OWL Lite, using only a subset of these predicates and semantic rules to simplify ontological reasoning, since OWL models, usually, have processing performance issues. A *heavyweight ontology* is a *semantic model* fully described with OWL-DL predicates, i.e., *lightweight semantic model* “enriched with axioms used to fix the semantic interpretation of concepts and relations” [160].

Examples of basic predicates are qualified cardinality restrictions, i.e., *property min n Class*, and existential quantifier as in predicate logic, i.e., a propositional function that can be satisfied by at least one member through a logical constant (*property some Class*) interpreted as “there exists,” “there is at least one” or “for some”. It enables local restrictions, i.e., properties that can be used by instances of a particular class or at least one value of a certain type (*some values from*).

OWL also provides relevant properties to support mapping different semantic models that may overlap, i.e., for ontology alignments. For example, for the instance level (Abox), the main predicate is *owl:sameAs*, which allows linking one instance to another instance that represents the same object. The *owl:equivalentClass* and *owl:equivalentProperty* are the properties used to map equivalent classes and properties on the terminology level (Tbox). Although these properties are helpful to represent 1:1 mappings, they cannot represent complex mappings such as n:n.

²⁶ <http://www.rdfhdt.org/>

²⁷ <https://virtuoso.openlinksw.com/>

²⁸ <https://aws.amazon.com/neptune/>

²⁹ <https://db-engines.com/en/system/Amazon+Neptune%3BVirtuoso>

³⁰ <https://www.w3.org/TR/rdf-sparql-query/>

2.3.4 Ontology Alignments and Semantic Translations

Over the past 20-30 years, numerous semantic models were developed to improve the semantic interoperability of diverse types of applications, such as healthcare systems, supported by several standardization organizations, such as HL7. These semantic models were adopted by various initiatives and are used by several information systems. If on one hand the creation of these semantic models aimed at addressing interoperability in isolated use cases, on the other hand these numerous semantic models, which sometimes even compete, made semantic interoperability more difficult to achieve in a broader sense. In an integration scenario through service interfaces, if each system relies on a different data structure that follows a semantic model, and these semantic models overlap some concepts, then harmonizing these conceptualizations is required. Therefore, achieving real semantic interoperability among such systems rely on an integration approach, which is usually done through ontology alignments (or ontology matching) [161].

Figure 2.12.
Example of ontology
alignments[58]

Ontology alignment techniques look for matches from different ontologies that contain overlapping and related concepts [143]. For example, the SSN/SOSA and SAREF ontologies have the same conceptualization for the term *Sensor* (further detailed in Chapter 3). The common *Linked Data (Semantic Web)* approach uses the *owl:sameAs* to define synonymous terms in the instance level (*Abox*) [162]. The main issue related to *owl:sameAs* is how to use it “when two entities are considered the same in some but not all contexts. Unfortunately, modeling the specific contexts in which an identity relation holds is cumbersome and, due to arbitrary reuse and the *Open World Assumption*, it is impossible to anticipate all contexts in which an entity will be used” [162]. The *Open World Assumption* refers to the presumption that the truth value of a statement may be true, no matter if it is known to be true,

which is opposite to the *Closed World Assumption*, which considers that any statement that is true is also known to be true.

Figure 2.13 illustrates the ontology alignments between ontology o and o' , where the class $o:Person$ is equivalent (*owl:sameAs*) to $o':Human$, the property $o:email$ is equivalent (*owl:sameAs*) to $o':has\ email$, and $o:SubjectArea$ matches (*owl:sameAs*) to $o':Topic$. These types of mappings are similar to the mappings used by MDE transformations.

The Ontology Alignment Evaluation Initiative (OAEI) is a benchmarking initiative for ontology matching [161]. OAEI emphasizes the need of user-assisted matching at runtime through semantic translation mechanisms. Semantic translation is the “process of changing the underlying semantics of a piece of knowledge. Given some information described semantically, in terms of a source ontology, it is transformed into information described in terms of a target ontology” [163]. Therefore, instead of using *owl:sameAs* approach to relate two ontologies, the semantic translation mechanism does not add any predicates for synonymous terms, rather, it transforms the instances (ABox) of the source ontology in instances of the target ontology based to mappings between their TBox. In the ontology alignment example (Figure 2.12), the semantic translation mechanism between o and o' would transform the instances of ontology o (input, prefix *oInst*) into instances of ontology o' (output, prefix *o'Inst*). For example, from this input data:

```
oInst:Joao a o:Chairman
oInst:Joao a o:Author
oInst:Joao o:email 'semiotics.ews@gmail.com'
```

The semantic translation $T:o \rightarrow o'$ would generate:

```
o'Inst:Joao a o':Chair
o'Inst:Joao a o':RegularAuthor
o'Inst:Joao o':hasEmail 'semiotics.ews@gmail.com'
```

According to the literature review on ontology matching techniques [164], a semantic translation mechanism can be implemented with a *model-based* technique, i.e., “exploit the semantic interpretation linked to the input ontologies”, along with a *constraint-based* technique, i.e., “consider criteria regarding the internal structure of the entities, such as the domain and range of the properties or the types of the attributes”.

Some methodologies to develop semantic translations using *model-based* and *constraint-based* techniques exploit the concept of a *semantic gateway* [58]. In the *semantic gateway* methodology, the main actor is the *gateway engineer*, i.e., the developer of the semantic translations. The following steps describe the main activities that a gateway engineer should execute: (1) identify the elements

(classes, object properties and data properties) from the source ontology that may be mapped to the target ontology – based on cognition (*model-based*). (2) describe the syntax rule(s) and “semantic rules for a case pattern that indicates a certain semantic information type” required to transform the element from the source to the target; (3) check whether the new rule(s) can violate any other existing rules, trying to find conflicts and contradictions and, if s/he finds, goes back to step 2; (4) specify the reverse direction of the translation rule, i.e., from target to source. This type of methodology is similar to mapping methodologies that are used to develop MDE transformations and *Extract, Transform and Loading* (ETL) data flows for *data warehousing* [165].

Formally, the translation mechanism is a derivation system defined in terms of inference rules [166]. The main measurements related to semantic translations are the *completeness* and *soundness* of the generated ontology compared to the source ontology, and the translation *efficiency* in terms of processing complexity. Therefore, a semantic translation system needs to be sound, i.e., only entailed sentences can be derived, needs to be *complete*, i.e., all entailed sentences can be derived [166].

The semantic translation accuracy measures how the inference rules “correctly translate and describe semantic information of an input message in the form of a target message” [58]. Furthermore, well-known information retrieval precision and recall measures are used to validate the correctness and completeness, respectively, of the matching systems [164]. Therefore, *soundness* can be improved through a higher translation accuracy and precision, while completeness can be improved through a higher recall.

A translation accuracy validation approach can use a classification to describe the accuracy level of the semantic information of test messages produced by the translation mechanism, grouped into describable semantic, indescribable semantic, and semantic violated messages. “*Describable semantic message* indicates a test message whose semantic information is clearly describable in the form of a target message. (...) The *indescribable semantic message* indicates a test message whose semantic information cannot be described in the form of a target message. (...) the *semantic violated message* indicates a test message which contains the semantic rule violations for a source message” [58]. It is possible to calculate the number of messages that are expected to match to each category with this approach, where accuracy is computed by comparing the number of matched to the expected translation output with the total number of test messages. A comparative validation approach can be used to measure the translation efficiency, by calculating the total translation time in a similar computational environment [58].

2.4 Context- and Situation-Aware Applications

With the advances on pervasive computing and network infrastructures on the past 20 years, research has been performed on context-awareness systems, where Service-Oriented Architectures (SOA) were predominantly applied [167-170]. Usually these context-aware systems use technologies to identify situations, which are classified as specification-based (top-down) or learning-based (bottom-up) [39]. In the former, the patterns that characterize a situation are defined a priori with expert knowledge, during design time. The latter technique applies artificial intelligence methods (e.g., machine learning) to identify situations from the available data. A hybrid approach is suggested to integrate represented and derived knowledge from the available data sources [171].

This thesis gives emphasis to the research that leverages the SOA approach by applying MDE to develop context-aware systems aligned to the *SA theory*, thus, for the development of *situation-aware* applications [140, 154, 168, 172-175]. This research applies a specification-based approach, where a graphical modelling language – the Situation Modelling Language (SML) – is used to represent situation types, i.e., the patterns that frame a situation of interest. SML follows the best practices for context modelling [176].

Ontology-driven modelling techniques are extremely suitable for context modelling [177], thus, SML is grounded in UFO and reuses some of the OntoUML elements [174]. MDE transformations can be applied to generate context-aware systems from SML models using one of the most common situation identification technologies: Complex Event Processing (CEP) [178, 179]. Following, we summarize the situation foundations behind SML, the main elements of SML and how SML models can generate situation-aware applications with distributed CEP engines.

2.4.1 Situation Foundations

In colloquial language, the term *situation* is often used with the same meaning as *context*. The term *situation* is also often used as a synonymous of *event*, especially in the *Complex Event Processing* (CEP) community. Formal ontology and human factors communities discuss whether the *situation* concept is an *endurant* or a *perdurant*. Following *endurantism*, a *situation* is a “snapshot” of the observed world, a static object. In contrast, following *perdurantism*, a situation has a temporal and spatial extent.

According to the *General Formal Ontology* (GFO), a foundational ontology used for domain modelling through the *General Ontological Language* (GOL), a situation is a special configuration that satisfies conditions of unity imposed by the participants’ properties [180]. GOL organizes time in two basic entities: time

boundaries and chronoids. A *Chronoid* ($chr(i)$) is a time interval that has a duration and has two time boundaries (left and right), denoted $tb(b)$, thus, $tb(b)$ depends on $chr(i)$. The theory of space of GFO is three dimensional, where a connected region of space is a topoid. “Regions of space are mereological sums of topoids, spatial part-of”. GFO provides the notion of situations through the *Situoid theory* [180]. The *Situoid theory* is based on the *Barwise Situation theory*, the same used by *Endlsey Situation Awareness*, differing from this theory mainly by distinguishing *states of affair*, *pictures of states of affair* and *infons* concepts. This theory is built upon the assumption that understanding situations depends on a theory of time and space, showing how to reuse an existing theory, extending it by adding new axioms.

In the *Situoid theory*, *Situation* is considered as the most complex kind of *Presential* while *Situoid* is considered as the most complex integrated whole. *Situoids* are more basic than *Situations*, as “parts of reality that can be comprehended as a whole (...), confined to a finite space and time”, while *Situations* depend on *Situoids*. Possibly, there is an infinite number of objects that are contained in *Situoids*. *Situoids* have the highest degree of independence and do not need other entities in order to exist. A *Situoid* has to be comprehensible as a whole, and is framed by a *Chronoid* (time) and a *Topoid* (space), being able to extend in time. “An association relation between *situoids* and certain universals is supposed to assure this notion”.

Therefore, a *Situoid* is an entity that does not need other entities in order to exist. *Situoids* are coherent and spatially and temporally connected, having a temporal and spatial extent. A *Situation* is the projection of a *Situoid* on a boundary of its framing *Chronoid*. The *state of affairs* is a state that distinguishes different kinds of states of affairs, such as instantaneous (at a single point in time) and enduring (in a time interval). States of affairs have a temporal extension, a set of time points (or time boundaries). An *Infon* consists of a *state of affairs* and contains information about *Situations*, where there must be a possible situation for any infon to hold, and another possible situation for an infon to not hold [166].

An initial effort to consider the notion of situation type in UFO was introduced in [173], based on GFO approach, which follows *endurantism*. Figure 2.14 illustrates a *Situation Type* is coined as *Situation Universal*, i.e., an *Universal* that is neither a *Substantial* nor a *Moment*. In UFO, an entity is classified as a *Substantial* (e.g., a person) related to the *Context* concept, which is interpreted as a *Moment Universal* that represents a particular context condition (e.g., a person’s location) and is different from the *Situation Universal* concept. The *Context Situation* concept was included by extending the *Situation Universal* concept, which “glues” the *Context* and *Entity* concepts *in time*. Figure 2.13 shows the representation of network availability as a *Relational Context* which inheres in *Network*, an *Intangible Entity*, and *Device*, an *Spatial Entity* (for a more detailed discussion, refer to Costa [181], page 92).

Figure 2.13.
Situation
conceptualization in
UFO (top) and
examples (bottom)
[173]

The notion of situation as a static object (a “snapshot” of the observed world) was also explored in UFO-B [182], as described before, where causality relations to events were formalized, as well as the activation relation to a disposition (a trope). The mereology relation pattern from events (atomic and complex) is also applied to Situation Types, thus, allowing the composition of situations from other situations. For example, Figure 2.14 illustrates how this pattern can allow a situation instance as a composition of other situation instances, such as a *Network Connected* (S_{20}) framing a WLAN connection (S_{18}) or a Bluetooth connection within a time window. How SML and CEP engines use the Allen’s operators for temporal relations are exploited as follows, as well as the lifecycle of situation management.

The main definitions used as situation foundations for SML are described as follows. A *Contextual Element* refer to a structural entity (an *endurant*) that participates in a context as a *contextual element* to avoid the overloaded use of the term *entity*, having similar meaning to an *Infon*. A contextual element can be perceived as a complete concept even if we were able to freeze time. A context is composed of one or more contextual elements.

Context is “what can be said about an entity in its environment, i.e., context does not exist by itself. The context of an entity may have many constituents, called context conditions” [172]. Context conditions of a person can be the person’s location, mental and physical states. Together, these conditions form the entity’s context. Context refers to the data elements being perceived from the observed world, i.e., the first level of the SA theory [88].

A context can be represented using a structural modelling language, i.e., context model is a conceptual model of context. Analogous to the ontological categories of *moment* in UFO (intrinsic and relational), we define two main

categories of context: *intrinsic context* and *relational context*. An *intrinsic context* belongs to the essential nature of a single entity, not depending on relationships with other contextual elements, e.g., the body temperature of a person. On the contrary, a *relational context* depends on the relation between different entities.

Figure 2.14.
Example of a situation
composition
instantiation [183]

An *event* (or *perdurant*) is an occurrence. An event can be atomic or complex. An *atomic event* occurs in a moment in time, does not last, i.e., its begin and end time points are the same. A *complex event* is an individual composed of other event instances, i.e., it accumulates temporal parts, extending in time [183]. An event is responsible for the transition between states of a contextual element. Examples of events are: an increase of a patient's body temperature, the arrival of an ambulance, a recommendation of hospitalization and a bite of a mosquito. Events are responsible for the lifecycle of a relationship (or *relator* in OntoUML), i.e., a relationship exists according to the occurrence of events. For example, the event of a health professional giving first aid to a patient can initialize an instance of a treatment relationship between the patient and a health unit. This instance depends on other events to keep their existence, such as the patient taking the medicine.

Finally, *Situation* is defined as a special configuration which can be comprehended as a whole, which satisfies conditions of unity that are imposed by certain universals, relations and categories associated with the situation [184]. Comprehension refers to the second level of the SA theory, i.e., the understanding on how the perceived data elements (contextual elements) relate to each other in a way that it can be recognized as a whole. Therefore, comprehension is the

reasoning capability of the phenomena observer – in the scope of SML, these are the software designers and domain experts. Universals refer to the homonymous category in UFO.

A *Situation* is a part of the reality that can be comprehended as a whole, it has duration and can be past (we use the term *deactivated*) or current (*activated*). Therefore, a situation is an individual that is composed by other individuals, including the states of contextual elements and other situations, constrained by formal and temporal rules. For example, in the situation of “John being infected with Zika”, the contextual elements are John, Zika and (possibly) an *Aedes* mosquito. The event that triggers this situation is the bite of the mosquito on John’s skin, followed by the event of the virus entering in John’s blood flow.

We classify *Simple* and *Complex Situation* as fundamental categories, avoiding the use of the term *type* to reflect the abstraction of a domain-related situation, the initial idea of ST. In order to address this abstraction of types of events, types of situations and their causal relations, we have employed the multi-level theory embedded in UFO-MLT [185]. We say that *situation* and *event* are individuals, while *situation type* and *event type* are first-order types. Thus, an *event* is *instance of (iof)* an *event type*, while a *situation* *iof* a *situation type*. “A mosquito biting a person” is an instance of an *event type*. “A mosquito biting John’s skin in a specific moment” is an instance of an *event* and, by transitivity, is *iof* that *event type*. Similarly, the *situation type* of “possible contagion of Zika within a geographical area” is a *powertype* of the situation of “a contagion of Zika in Brazil in 2016”. With this distinction, we can abstract causal relationships among types of *event* and *situation*, e.g., the *event type* of “a mosquito biting a person” can *cause* the *situation type* of “possible contagion of Zika”.

2.4.2 The Situation Modelling Language (SML)

In SML, a *Situation Type* (ST) is the representation of patterns of contextual elements in time. A ST allows the characterization of rules among these contextual elements and their changes caused by events. The Situation Modeling Language (SML) [158, 186] supports the specification-based approach for situation-aware applications. SML enables the definition of situation types as a set of constraints related to its participants’ properties and relations, including composition of situations and their relations in time. Situations can be slightly simple, such as the situation type “person with fever”, where the “person’s temperature is over 37 degrees”, or more complex combinations of situations in time. Figure 2.15 illustrates a complex situation type, the “tuberculosis disease situation” (TBDisease), which is composed by a number of sequential “tuberculosis symptom situation types” (TBSymptom) during a “tuberculosis infection” (TBIinfection).

Figure 2.15. Timeline of Tuberculosis disease detected from symptoms and infection [158, 186]

SML allows its users to abstract from lower-level entities and properties, focusing on the higher-level patterns that emerge in time. With this approach, separation of concerns through abstraction can be achieved, as well as the reuse of previously defined situation types for the definition of more complex ones. The temporal relations between situations also refer to the situations' activation states, so a complex situation instance can contain both active and/or inactive situation instances in its composition. Figure 2.16 illustrates the possible temporal relations between two situations according to their activation states: (1) active-active refers to the temporal relation when situation A and B are both active (currently happening), in which only *overlaps*, *includes* and *starts* relations (and inverses) are applied; (2) active-inactive refers to the temporal relation when situation A is active and situation B is inactive (past), in which only *finishes* and *coincides* relations (and inverses) are not applied; (3) inactive- inactive refers to the temporal relation when situation A and B are inactive, in which all relations are applied.

Figure 2.16. Temporal relations between two situations [158]

	Active - Active	Active - Inactive	Inactive - Inactive
A Before B B After A		A B	A B
A Meets B B Met by A		A B	A B
A Overlaps B B Overlapped by A	A B	A B	A B
A Finishes B B Finished by A			A B
A Includes B B During A	A B	A B	A B
A Starts B B Started by A	A B	A B	A B
A Coincides B			A B

Active Situation
 Inactive Situation

SML also provides a graphical language founded in principles of semiotic clarity, perceptual discriminability, semantic transparency, dual coding and visual

expressivity. An example of the simple situation type of “person with fever” is illustrated in Figure 2.17. It refers to the entity “person” and its property “temperature” from the context model, which is described using OntoUML [158]. In addition, SML takes advantage of the OntoUML modelling assessment approach within its ontological model-driven methodology. Therefore, during the specification phase it is possible to enhance the semantics of SML and OntoUML models through a cyclical verification and validation process. SML allows the situation types to be specified as the condition patterns of a rule and can be compared to the “when” clause of the Event-Control-Action (ECA) domain specific language [181].

Figure 2.17. Fever situation represented with SML [158]

Therefore, SML uses OntoUML (and UFO) to support context modelling for situation detection because OntoUML improves the expressiveness of the context models. Once the context model is described with OntoUML and the situation types with SML, the designer can make use of the cyclic verification and validation approach introduced in [174]. This approach allows syntax verification of the context model and visual validation through simulation (also called situation assessment). This approach uses the Alloy logic language to automatically generate possible instances of the STs and their participants as object diagrams that the user may inspect to find whether the model represents intended or unintended state-of-affairs.

2.4.3 Realization with Complex Event Processing (CEP)

After model verification and validation, where the designer repeats the activities until sufficient confidence in the models is achieved, the situation-aware software code can be automatically generated from the SML model to concrete platforms. Although RDF is applicable for a range of semantic applications, it may be not suitable for applications that require size-efficient formats and high performance analytics, such as the Internet-of-Things (IoT), which use a variety of data formats that allow specific and scalable manipulation of data for pattern recognition and knowledge discovery, such as CEP technologies [187].

In situation-aware applications research, the MDE approach is used to generate CEP software, which is capable of representing and applying complex queries to data streams to detect specified conditions over the events arrivals, triggering appropriate actions in real time [188]. CEP is a flexible technology that can be included in an overall data management strategy and architecture, improving data management solutions for real-time situation identification [189].

The *Drools* rules management system has a CEP engine³¹ (initially called as *Drools Fusion*) that enables the selection of events from a set, as well as correlation, aggregation, composition, streaming, temporal relation constraints and sliding windows. The SCENE platform was introduced in [190] as an extension of Drools Fusion that adapts the Drools Rule Language (DRL) to natively support situation types and the situation lifecycle management, involving composite situation pattern recognition based on the situation activation and deactivation instants (time points inherited from the lower-level events). With SCENE as implementation technology, the context model and situation types are coded in Java, while situation rules are mapped to DRL patterns. MDE transformations are applied over SML models to generate Java and DRL code in a (semi) automatic fashion.

SCENE was extended for a distributed rule-based situation management approach, where parts of the rule detection engine are processed in different rule engines that are able to share their working memory [191]. The Java ESPER³² was also adopted by this research, where mappings from SML metamodel to the Event Processing Language (EPL) were created by the Situation Mapping Layer for ESPER (SIMPLE) approach³³.

³¹<https://docs.jboss.org/drools/release/6.2.0.CR3/drools-docs/html/DroolsComplexEventProcessingChapter.html>

³² <http://www.espertech.com/>

³³ <https://github.com/Simonassi/SIMPLE>

Internet-of-Things Platforms

In this chapter, we describe the concept of an IoT Platform as a container of technologies that enable the rapid and reliable development of IoT-based applications. Section 3.1 describes the IoT enabling technologies, such as sensor networks integrated to publish-subscribe brokers, the microservice RESTful architecture and cloud computing. We emphasize the role of context and service brokers as field and cloud gateways because of their importance to interoperability. Section 3.2 describes an industry-oriented IoT platform, the *Microsoft Azure* IoT platform along with an example of application of driving habits (*MyDriving*). Section 3.3 presents the current research on semantic interoperability of IoT platforms, revisiting the main research initiatives with their lessons learned and open issues. This section also covers best practices for IoT ontology engineering and the standards W3C SSN/SOSA and ETSI SAREF. The role of semantic translations for semantic interoperability of IoT platforms is also exploited. Finally, Section 3.4 concludes this chapter by summarizing the lessons learned and open issues.

3.1 Enabling Technologies

IoT technologies are leveraged by cloud computing, having physical devices that generate, consume and exchange data with services that promote the integration between information systems and the physical world [192]. Some common capabilities for IoT-based applications³⁴ are: device management, authentication and authorization (w.r.t. users and permissions), publish/subscribe context data, data processing and storage capabilities as, e.g., NoSQL big data with cloud scalable databases, on-the-fly data collection from devices, vendor agnostic IoT device connectivity, device heterogeneity support and modularity.

³⁴ The terms *IoT-based applications*, *IoT system* and *IoT solution* are considered synonymous here.

An *IoT Platform* is a relatively new concept [193], which is a container of technologies that enable the rapid and reliable development of IoT-based applications, i.e., an IoT platform enables ICT solution engineers to bridge the gap between device sensors and connected applications through suites of components. This concept is highly exploited in both industry and academy [194]. The Gartner hype cycle (2017) shows that the IoT platform concept is in the peak of inflated expectations receiving a huge investment from several initiatives. IoT-based applications can benefit from IoT platforms to realize (near) real-time data acquisition, situation detection and message brokering between data sources and warning destinations considering local networks and the cloud [12].

Figure 3.1.
Common reference
architecture (from
Intel IoT platform)
[195]

Most of the commercial big players of cloud services provide a general-purpose IoT platform in a “plug-and-play” fashion, such as *Microsoft Azure* IoT solution accelerators³⁵, *Google Cloud* IoT core³⁶, *IBM Watson* IoT³⁷ and *Amazon AWS* IoT³⁸, among others. Figure 3.1 illustrates the reference architecture of *Intel* IoT platform [195], which reflects the common layered architecture of IoT platforms, showing how IoT devices (or *things*), networks and clouds communicate, indicating the main capabilities of each layer. Some IoT platforms are domain-oriented, such as *Google Fit* SDK³⁹ and *Apple HealthKit*⁴⁰ for healthcare applications, often referred as *medical IoT platforms* [196]. Often, a domain-oriented IoT platform specializes the reference architecture to include common domain characteristics, such as the case of industrial (manufacturing) platform derived from the *Bosch* IoT platform for Industry 4.0⁴¹. In general, these industry-oriented IoT platforms leverage

³⁵ <https://azure.microsoft.com/en-us/features/iot-accelerators/>

³⁶ <https://cloud.google.com/iot-core/>

³⁷ <https://www.ibm.com/internet-of-things>

³⁸ <https://aws.amazon.com/iot/>

³⁹ <https://developers.google.com/fit/>

⁴⁰ <https://developer.apple.com/healthkit/>

⁴¹ <https://www.bosch.com/products-and-services/connected-products-and-services/industry-4-0/>

their business model by adhering to open standards, such as the *Open DataBase Connectivity* (ODBC) to connect to databases, the *OAuth 2.0* for authentication and authorization, and programming languages as Java, C#, C, Javascript (among others).

Some IoT platforms were created in the context of research and innovation projects, which are open source and reliable, such as FIWARE⁴², WSO2⁴³ and *universAAL*⁴⁴ IoT platforms. In particular, *FIWARE* is an EC initiative supported by the FP7 and H2020 schemes through projects like FI-NEXT⁴⁵ and SmartSDK⁴⁶, which has been used in several application domains, such as smart city, smart farming and smart factory (Industry 4.0) [47, 197]. *FIWARE* provides a catalogue of *enablers*⁴⁷ as “open source platform components which can be assembled together and with other third-party platform components to accelerate the development of Smart Solutions”.

The *UniversAAL* IoT platform was initially produced for health assisted living IoT solutions, and represents data through semantic models described with OWL, serialized as XML or TTL, in a pluggable way⁴⁸. Although *UniversAAL* has an architecture that considers RDF for data representation, it does not include any standardized ontologies and do not provide ontology alignments rather than using *owl:sameAs*, lacking on semantic translations (see Section 2.3). Including or extending the semantic models require a lot of manual efforts. The *WSO2* platform is widely adopted by several enterprises and has an active developer community, but it does not have support for semantic technologies.

An *IoT platform component* (or *artifact*) is a combination of software and hardware to address the service(s) required by the IoT solution, usually organized according to the generic layered architecture (device, network, session, cloud and application layers), which is supported by management and security services, as explained in [198]. Figure 3.2 illustrates the four main components of an IoT system: the IoT device, including sensor networks, the middleware (IoT gateway), the cloud services (IaaS, PaaS, SaaS and data storage) and the IoT application. Event-oriented and message-oriented middleware are used as context and service brokers, and most of IoT platforms recommend the application of the RESTful microservice architecture along with distributed CEP engines for stream analytics.

⁴² <https://www.fiware.org/developers/>

⁴³ <https://github.com/wso2>

⁴⁴ <https://github.com/universAAL>

⁴⁵ https://cordis.europa.eu/project/rcn/207032_en.html

⁴⁶ https://cordis.europa.eu/project/rcn/205595_en.html

⁴⁷ <https://catalogue-server.fiware.org/enablers>

⁴⁸ <https://github.com/universAAL/ontology>

3.1.1 Using Middleware for Context and Service Brokers

The most necessary element of an IoT platform is a *Middleware*, because of the distributed nature of an IoT platform, such as the *Orion Context Broker Generic Enabler* of *FIWARE*. A broker implemented with a middleware for IoT solutions is commonly used as an *IoT Gateway*. Heterogeneous devices usually adopt different communication protocols, affecting the integration of data provided by these devices. IoT platforms use middleware solutions to develop IoT gateways, which “acts as a portal layer between two elements of one or multiple IoT systems, allowing them to share information by communicating between the adopted IoT protocols” [198].

An *IoT Gateway* is able to integrate IoT devices that apply different session protocols, such as the *Advanced Message Queuing Protocol* (AMQP), the *Message Queuing Telemetry Transport* (MQTT) and the *Constrained Application Protocol* (CoAP), making system components technologically interoperable. AMQP is an open standard protocol for *Message-Oriented Middleware* (MOM) that supports message orientation, queuing, routing (point-to-point and publish-subscribe), reliability and security. The MQTT protocol implements the standard ISO/IEC PRF 20922 and supports the *publish/subscribe* pattern. MQTT was designed to provide a “small code footprint” connection and to work with limited network bandwidth. CoAP is a protocol designed for machine-to-machine (M2M) interactions with constrained nodes and networks.

The publish/subscribe message brokering pattern is the most common technique applied for IoT solutions using either MQTT or AMQP [199]. In publish/subscribe, “*publishers* write to *Topics* and *Subscribers* read from the *Topics* on which they are registered. One *Publisher* can write to many *Topics* and one *Subscriber* can read from many *Topics*. Unlike the event-based pattern, the subscribers in this pattern are all interested in a type of event happening without knowing the publisher of the event” [198].

The publish/subscribe message brokering pattern requires a message broker [200], which can be implemented with a MOM or an *event-oriented middleware*. The main difference between *events* and *messages* is that an *event* reflects a state change in the environment monitored, e.g., the increase of a room temperature from 19 to 20 degrees, while a *message* reflects the data produced by a service, e.g., in smart home context, an emergency notification warning that the average temperature of all rooms is above a certain threshold. Therefore, while a context broker is implemented with an *event-oriented middleware*, a service broker is implemented with a MOM, which are architecturally similar to *Enterprise Service Bus* (ESB). In practice, the technological solution for these types of middleware are similar. Some on-premise platforms that can implement both middleware for messages and

events are the *Apache Kafka*⁴⁹, *RabbitMQ*⁵⁰, *Mule*⁵¹ and *StormMQ*⁵². Besides messages and events, *telemetry*, *alerts* and *ingestion* are terms commonly used to describe IoT data streams. *Ingestion* is the process of inserting data records into a database repository through a gateway. *Telemetry* refers to the messages with data produced by sensors, which are usually analyzed, locally or in the cloud, according to a set of rules.

Figure 3.2. Layered integration approaches of IoT Solutions [198]

The *Microsoft Azure* IoT platform provides the *Event Hub* as a low latency *event-oriented middleware*, which implements the publish/subscribe pattern and has been designed to receive a high volume of data, being able to ingest a high throughput of millions of events per second, using partitions to enable horizontal scalability, i.e., it enables scaling as a single IoT gateway. *Azure* IoT also offers the *IoT Hub*, which is an *Event Hub* with additional features, such as security features at the device level and bi-directional connection between the broker and IoT devices. Since the *IoT Hub* is used in the validation of this thesis, it is further described in next Section.

⁴⁹ <https://kafka.apache.org/>

⁵⁰ <https://github.com/rabbitmq>

⁵¹ <https://www.mulesoft.com/>

⁵² <https://github.com/stormmq>

To tackle scalability and resilience, some IoT gateways can support scaling both in horizontal, i.e., scaling within a single broker, and in vertical, i.e., scaling through interlinked broker clusters, where each cluster can be organized as a hierarchy with a head broker and edge brokers. The most advanced gateway deployment patterns for integration are (1) multi-protocol web services and (2) intelligent gateways [198]. IoT Hub is an example of an intelligent gateway that supports MQTT, AMQP and HTTPS⁵³. In addition, brokers “in the field”, also termed as *field gateways* (or local or device gateways), are often deployed in IoT device(s) or near them to avoid data propagation delays and flaws, allowing optimal network routing and data aggregation in the IoT device level.

Besides the middleware integration approach, IoT systems can be integrated via cloud environments (services and data sharing) or at the application level (“IoT system-of-systems integration”). In both approaches, the *Representational State Transfer* (REST) microservices architectural style is usually applied to enable the integration of loosely coupled applications using HTTP [201-204].

3.1.2 RESTful Microservices

REST is a design pattern for the interaction among resources on the web. It relies on stateless, cacheable client-server communication, mapped onto HTTP methods (GET, PUT, POST, and DELETE). The RESTful architecture can be considered as the SOA 2.0 architecture [205], which is an evolution of SOA 1.0 (based on SOAP), addressing technological interoperability.

Most of the current IoT solutions follow the event-driven architecture for data exchange among IoT devices, networks and applications through RESTful web services using the JavaScript Object Notation (JSON) as data exchange format [206, 207]. JSON is a lightweight serialization format for HTTP and JavaScript that is intuitive for humans to read/write and adequate for machines to parse, generate and interchange data. JSON is programming language-independent, scalable and fast; and it uses fewer resources and yields more compact models than XML [208]. Data structure schemas can be completely integrated with data in JSON. There are some approaches that compress the content to reduce the message volume, but these approaches consume more processing time since compression and decompression is necessary at the source and at the destination, respectively.

Finally, JSON is preferred by programmers due to the notations that resemble the C-family of languages (e.g., C++, C#, Java, JavaScript, Python) and is implemented by many parsers, having full support by several programming languages. JSON has higher parsing efficiency than XML and has been widely used in asynchronous web applications [208]. Because of these benefits, JSON is often

⁵³ <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-protocols>

used as payload notation for IoT solutions [209]. Besides JSON and XML, some other popular formats are the *Apache Avro*⁵⁴ and *Apache ORC*⁵⁵, which are formats oriented to *Hadoop*, and the Efficient XML interchange (EXI)⁵⁶, a W3C recommendation to compact XML.

Figure 3.3 illustrates a RESTful architecture that connects applications running in different devices, e.g., tablet, laptop, mobile phone and desktop, to service endpoints through HTTP. This architecture uses multiple serialization formats; JSON, for example, can be easily parsed to *Plain Old Java Classes* (POJO) or *Plain Old CLR Objects* (POCO) in the .NET environment, so that data can be stored in relational databases through an *Object Relational Mapper* (ORM). JSON messages describing hash tables can be stored in key/value databases and data described in RDF – serialized in JSON-LD or XML, for example – can be stored in *triplestores* (graph databases).

Figure 3.3. RESTful microservices' architecture [210]

In particular, the architecture illustrated in Figure 3.3 refers to the microservices' architecture, which is an alternative way to defining and combining services. *Microservices*, represented with hexagons in the figure, gained popularity with SOA 2.0, aiming at implementing small services, each running on its own process and communicating with lightweight mechanisms (REST APIs), addressing scaling, resilience, technology and vendor lock, agile methodologies and business agility. According to [210], the main capabilities of microservices are: (1) functional decomposition with “small” services; (2) design for failure, targeting service resilience; (3) evolutionary, being able to migrate one service at time; (4)

⁵⁴ <https://avro.apache.org/>

⁵⁵ <https://orc.apache.org/>

⁵⁶ <https://www.w3.org/TR/exi/>

service versioning; (5) service monitoring; (6) compatibility with agile methodologies; (7) automation using continuous delivery.

The SOA principles already tackled 1-5, though failed to fully address them [210]. The microservices architecture strengthens these five capabilities, including the success of the agile methodologies (#6) and continuous delivery (#7). Microservices also differs from traditional SOA, which aims at integrating various business processes from the enterprise point of view. The microservices architecture is a way of developing an enterprise business application with multiple small independent services, where the microservices collaborate among them to address the application client request. In SOA, a service is an endpoint that external applications can communicate with the underlying business application, and often is a part of a monolith deployment [189].

A microservice implements a single business function, and each service can be fully automated and independently deployed through a *container*, which is a package of the application components provisioned with the minimal resources required, including the OS, thus not needing to embed an OS (as a virtual machine requires). Microservices can be written in different programming environments and languages, using diverse data storage technologies⁵⁷.

A core concept of the microservices architecture is the API gateway, which is a proxy to register APIs exposing the functions implemented by the microservices. The main idea is to avoid direct connections among the microservices, which is an approach that causes the ‘spaghetti problem’, making it difficult to manage security, network and scale for high-load API calls. Microservices should only be exposed and connected to each other through the API Gateway, enabling full control under API calls.

The main technical difference between an ESB and an API Gateway is that an ESB implements two queues (outgoing request and incoming response) for a synchronous call, while an API Gateway is able to create a waiting thread on both sides using asynchronous calls. Besides that, some API gateways have features that differ from ESB, such as configuration languages with stateless execution for speed and scalability, post-deployment approval process leveraged by high available and self-service developer portal environments, and the provision of business statistics related to the APIs use (who and how much), allowing the measurement of the API portfolio according to the business objectives.

3.1.3 Complex Event Processing and Data Management

Besides brokers and the microservice RESTful architecture, IoT platforms are leveraged by the event-driven architecture (EDA), where distributed CEP is the

⁵⁷ <https://martinfowler.com/microservices/>

technology to implement situation identification [205]. Implementing real-time situation identification for critical big data applications is challenging, since (1) incoming data streams arrive from heterogeneous sources at high rates, which sometimes fluctuate; (2) data need to be quickly processed; and (3) event patterns that form a situation can often be quite complex. CEP addresses these issues and, therefore, CEP is a useful technology to address IoT requirements and constraints [211]. For example, usually, IoT data are time series that should be correlated using complex operators.

Cepheus⁵⁸ is the CEP engine adopted by the FIWARE IoT platform, based on Java ESPER, which uses a SQL-like language, called EPL, to describe the complex event patterns that frame a situation type. Different from SQL queries, which require stimulus to run and search over the stored data, EPL queries are constantly under execution by the CEP engine in a way that when new data arrive, the search engine can find the pattern with little latency. To support this mechanism, CEP stores “recent” data in an internal working memory. EPL execution resembles a *trigger* in SQL, though triggers face several implementation issues such as, e.g., being synchronous and locking data changes while executing. EPL offers operators such as time windows, temporal event sequences and other *Allen’s* temporal relations. ESPER has a .NET version called NESPER⁵⁹, which also allows the use of EPL queries.

Code Snippet 3.1. EPL query using event stream patterns

```
select tick.price as tickPrice, trade.price as tradePrice,
      sum(coalesce(tick.price,0) + coalesce(trade.price, 0)) as total
from pattern [every tick=StockTickEvent or every
      trade=TradeEvent].win:time(30 sec)
```

EPL queries also use relational algebra and query planning that allows many-to-many relationships through *joins*, which is also prepared for out-of-order data arrival. EPL patterns are dynamically-managed state trees, suitable to identify temporal relations based on event arrival order with presence (or not) of events and other situations. EPL is useful to find patterns among ordered events, to retain events with data windows (an expensive mechanism), to aggregate context partitions and has a syntax to describe filters that is easier to read than SQL where-clauses. Equivalent to data tables in SQL, in EPL classes defined in POJO (or POCO) are used in the internal data model. Code snippet 3.1 illustrates an EPL query using pattern-based event stream. It shows an ESPER example in the financial market domain that reacts to stock tick events or trade events within a window of 30 seconds, by listing their individual prices and the total.

⁵⁸ <https://fiware-cepheus.readthedocs.io/en/latest/>

⁵⁹ <https://github.com/espertechinc/nesper>

The main difference between CEP engines and stream processing engines is that a CEP engine allows the developer to implement queries using a high-level language such as EPL or DRL (rule-based approach of SCENE). A stream processing engine allows the developer to implement a processing graph, where each processor sends events to next processors. Examples of popular stream processing engines are the *Apache Flink*, *Apache Storm* and the *Apache Spark*, which are leveraged by *MapReduce* and *Hadoop*.

Usually, in most stream processing engines, the developer needs to write code to implement the complex operators (e.g., temporal windows) when required. Moreover, most CEP engines are tuned for low latency, often responding in the scale of milliseconds, while stream processing engines respond in the scale of seconds. Stream processing engines are better suitable to implement data lakes with *Extract, Load and Transform* (ELT) data flows, in which large volumes of data need to be processed in several distributed hosts without stringent latency requirements, while CEP engines are better suitable for IoT solutions, which, usually, require low latency.

Figure 3.4. Event-driven architecture of Microsoft Azure IoT platform⁶⁰

Recently, some of the big players in stream processing and IoT platforms are providing CEP capabilities to stream processing engines, such as the WSO2 Stream Processor, through SQL-like languages called *Streaming SQL*. In this context, the term *stream analytics* gained popularity with IoT platforms, in particular by the MS Azure IoT platform, which provides the *Azure Stream Analytics*⁶⁰ as a scalable CEP engine. The event-driven architecture employed by the Azure IoT platform is illustrated in Figure 3.4, showing the cycle of (1) event production from

⁶⁰ <https://docs.microsoft.com/en-us/stream-analytics-query/stream-analytics-query-language-reference>

<https://github.com/Azure-Samples/MyDriving/tree/master/src/StreamAnalytics>

applications and IoT devices (and field gateways), (2) event queuing and stream ingestion through brokers, (3) stream analytics with CEP to (4) store, present and act over the identified situations. In this architecture, stream analytics can feed or consume data from data lakes (blobs) and machine learning solutions. Although the *Azure Stream Analytics* query language covers the main features of a CEP language and provides a low latency distributed CEP engine, the query language lacks on a number of predicates that EPL implements and, thus, EPL is more powerful than the *Azure Stream Analytics* language.

The main metrics to measure the performance and scalability of IoT solutions are: *latency*, *peak rate*, *jitter* and *resource utilization*. *Latency* refers to the total transaction time of a message (sometimes also called *response time*), i.e., the time between the message is produced at the source and the generation of the causally dependent message(s) at the destination. For example, in an IoT solution where a temperature sensor sends data continually to a gateway running a CEP engine that has an EPL statement to detect temperatures above a threshold, the latency refers to the total time from the sensor to the detection event generation of the CEP engine. The latency is the basis to calculate the solution throughput, which can be a non-functional requirement for the IoT solution, such as “the throughput of the system has to be at least 700 messages per second” [212]. *Peak rate* is the aggregated peak of the output message rate that can be generated by the CEP engine (higher is better). *Jitter* is the variation of the observed throughput against the expected throughput. *Resource utilization* includes CPU and memory when uniform distribution of loads is tested, showing possible bottlenecks in the IoT solution. Scalability and reliability are metrics also often used to evaluate IoT solutions, and can also be formulated as non-functional requirements, such as “the system must be able to transmit data with at least 95% reliability”. All of the aforementioned parameters are highly dependent on the CEP solution that is chosen.

The generation of CEP code through MDE transformations from a higher abstraction level not only facilitates the specification of a situation types but also impacts directly the interoperability of a system. Using the terms from the *SA theory*, while the data models underlying an IoT solution refer to the perception level, the complex event rules refer to the comprehension level, which is crucial for semantic interoperability since these rules describe the meaning of a situation.

Other data management technologies that are crucial for IoT solutions include traditional relational database management systems and the NoSQL (not only SQL) databases, such as document-oriented, key-value and graph databases. Usually, relational databases are used to guarantee data persistency, having higher insertion latency and an average querying latency. Document-oriented database is specialized in storing text files, especially JSON, having low latency to insert but high latency to query, besides higher complexity to specify queries if compared to SQL. Graph databases stores data with graph structures (nodes and edges),

allowing fast processing of data with simple calculation of specific graph properties, e.g., the number of steps required to go from one node to another. A particular type of graph database is the *triplestore*, which is oriented to the RDF language.

3.2 Microsoft Azure IoT Platform

The *Microsoft Azure* IoT platform is used in the validation of the approach proposed in this thesis along with its main use case (*MyDriving*), and, therefore, the platform architecture (with emphasis on *IoT Hub*) are briefly described in this section. *Azure* IoT platform is a collection of technologies and cloud infrastructure that aims at accelerating business transformation with IoT, gaining insights from connected devices and turning these insights “into action with powerful applications built on the industry-leading platform for IoT development”⁶¹. *Azure* IoT platform provides several open source tools to improve IoT management, called IoT accelerators⁶². It supports connecting and monitoring devices remotely through the open-source remote monitoring solution⁶³, enabling the improvement of equipment operation and providing predictive maintenance through stream data analysis. The *Azure* IoT Suite is part of these accelerators and provides preconfigured implementations of common IoT solution patterns. Furthermore, it provides an environment for developing and testing IoT solutions with device simulation. An example is the *Azure IoT toolkit*⁶⁴, which provides devices, *IoT Hub* and modules’ management, demonstrating how device-to-cloud (D2C) and cloud-to-device (C2D) messaging are performed and enabling code generation for diverse languages (e.g., C#, Java, Node.js).

3.2.1 The Role of *IoT Hub* in the Reference Architecture

The *Azure* IoT reference architecture is illustrated in Figure 3.5, which follows foundational principles, such as data heterogeneity for multiple domain applications involving different software and hardware; security, which is employed across all subsystems, including privacy measures, device and user identity, authentication and authorization, real-time and historical data protection; hyper-scale deployments, i.e., scalable deployments from small number of devices to millions; and flexibility upon composability, allowing the combination of components and the integration with existing applications.

⁶¹ <https://azure.microsoft.com/en-us/overview/iot/>

⁶² <https://azure.microsoft.com/en-us/features/iot-accelerators/>

⁶³ <https://github.com/Azure/azure-iot-pcs-remote-monitoring-dotnet>

⁶⁴ <https://github.com/Microsoft/vscode-azure-iot-toolkit>

The architecture is organized into three layers: *things*, *insights* and *action*. *Things* are the elements responsible to send sensor data events to generate *insights*, which are used to support *actions* that improve a business or process. In general, a solution implementing this architecture provides data to the cloud through an IoT device, which can act as a *field gateway*, connected to a cloud gateway (IoT Hub), i.e., the cloud context broker. The architecture allows two paths: “warm” and “cold”, the former for real-time data, i.e., data required to be available for immediate reporting and visualization, and the latter for historical data, i.e., for long-term data storage and batch processing.

Figure 3.5. Microsoft Azure IoT Platform reference architecture⁶⁵

Stream processing refers to CEP capabilities, either implemented with *Azure Stream Analytics* or with a third-party CEP engine that receives event data from the *IoT Hub* according to the publish/subscribe pattern, while simple rules processing can be implemented with *IoT Hub Routes*. Business process integration includes data storage and notification services. UI and reporting tools can interact with *Stream Processing*, e.g., by receiving data and generating alerts. Finally, machine learning can be applied over the historical data to discover patterns on the data, supporting decision making.

Figure 3.6 illustrates the four device connectivity options for solutions using the *Azure IoT* platform: (1) direct connection to the internet for IP-capable devices, able to establish secure channels; (2) a *field gateway*, which is able to pre-process data (e.g., aggregate streams) before transferring to the cloud. Recently, *Microsoft* launched the *Azure IoT Edge Device* to be used as field gateway, which is able to act “as an intelligent edge and facilitates on premise implementation of routing,

protocol translation, machine learning, artificial intelligence, stream processing”; (3) and (4) are options to connect a device to *IoT Hub* via a third-party cloud gateway.

Therefore, the central component of *Azure IoT* platform is the *IoT Hub*, which provides several services, acting as an intelligent broker. An IoT device can send D2C message and receive C2D messages to/from an *IoT Hub*, which provides endpoints for resource and device identity management, device twin management and jobs management. Moreover, an *IoT Hub* exposes service endpoints for an application back end to communicate with the IoT devices via HTTP and AMQP. IoT Hubs can be scaled horizontally with several partitions and vertically with multi-cluster topologies and sharding⁶⁵. Partitions are used to reduce contentions that can occur with read/write concurrency of event streams in one IoT Hub. Multi-cluster and sharding are used for IoT solutions that require high-performance that a single *IoT Hub* cannot provide. These scaling options are limited by the operating throttles, i.e., rate and throughput limitations that the *IoT Hub* can support for each of subscription level⁶⁶. In summary, *IoT Hub* subscriptions can be extremely performative but with a high monetary cost. Besides the *IoT Hub*, *Azure* also provides *Event Hubs* and *Service Bus*, which are brokers with less capabilities than the *IoT Hub*.

An *Event Hub* instance can be used as a context broker for speed and instantaneous acceleration measurements, while another *Event hub* can be used as context a broker for fuel consumption and distance traveled. These data are then accessed through processing units (*Service Fabric*), which can be either stateless and state-enabled microservices. This capability enables IT operators to use simple workflows to provision, deploy, patch, and monitor applications, greatly reducing the IT operation burden.

Regarding data formats (syntax) published in the *IoT Hub*, *Microsoft* recommends to “adopt the JSON format, due to its readability and relatively low space required, however there are several binary formats, such as Avro, that can improve performance and reduce cost”.

Regarding cybersecurity, the “*Microsoft Azure and Data Compliance in the context of the Internet of Things*”⁶⁷ explains how to comply with privacy, security and ethics requirements in an IoT environment. It explains how *Azure* addresses key principles in general and IoT-specific (e.g., *IoT Hub*) cloud services. In particular, *Microsoft* recommends to build a threat model⁶⁸ considering all *Azure IoT* components and

⁶⁵ <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-scaling>

⁶⁶ <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-quotas-throttling>

⁶⁷ <https://gallery.technet.microsoft.com/Azure-and-Data-Compliance-bda1f632>

⁶⁸ <https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-architecture#threat-modeling-the-azure-iot-reference-architecture>

possible threats, such as processes, spoofing, denial of service, tampering, information disclosure and elevation of privilege.

Figure 3.6. Types of connectivity between IoT devices and the cloud with Azure⁶⁵

Furthermore, Microsoft provides a complete guide on designing secure healthcare solutions compliant to the General Data Protection Regulation (EU) 2016/679 (GDPR)⁶⁹. This guide illustrates how to apply thirteen secure development and operation principles: (1) Enable identity and authentication solutions; (2) Use appropriate access controls; (3) Use industry-recommended, enterprise-wide antimalware solution; (4) Effective certificate acquisition and management; (5) Encrypt all customer data; (6) Penetration testing; (7) Threat modeling services and applications; (8) Log security events, implement monitoring and visualization capabilities; (9) Determine the root cause of incidents; (10) Train all staff in cyber security; (11) Patch all systems and ensure security updates are deployed; (12) Keep service and server inventory current and up-to-date; (13) Maintain clear server configuration with security in mind.

3.2.2 MyDriving: Real-Time Vehicle Tracking

MyDriving solution⁷⁰ is the main open source use case of the Microsoft Azure IoT platform that provides a starting point for scalable, high-performance and cross-platform IoT implementation. *MyDriving* targets any person that requires a better awareness of its driving habits and how it affects safety and long-term vehicle performance. Real-time telemetry are collected from the vehicle, such as location,

⁶⁹ <https://gallery.technet.microsoft.com/Azure-A-Practical-Guide-to-5ebdc8bd>

⁷⁰ <https://azure.microsoft.com/en-us/campaigns/mydriving/>

speed, instantaneous acceleration, distance, fuel consumption, temperature and RPMs, and processed both locally (in the field) and in the cloud, providing meaningful information that can be accessed through various types of UI.

Figure 3.7.
Overview of the
Azure IoT *MyDriving*
solution⁷⁰

The end user smartphone plays two different roles in the *MyDriving* solution: (1) a field gateway, collecting, aggregating and transmitting data to the cloud; and (2) UI for data visualization, being able to show the current trip information in a map at real-time, as well as previous trips, besides the aggregated data from historical information, such as the total distance, average speed, places most visited and total number of hard accelerations for consumer experience. The *MyDriving* mobile app was designed like that because most drivers have a mobile phone and internet access via 3G/4G. In addition to the mobile device, the solution is leveraged by the *On-Board Diagnostic* (OBD) II sensor that can be deployed in any vehicle built in the past 20 years and is able to provide vehicle telemetry to the mobile phone via *Bluetooth*. Although the OBD II sensor provides more information and more accurate data, the *MyDriving* solution can be used without this sensor by using telemetry from the mobile phone, however limiting some of the solution capabilities.

The *MyDriving* solution architecture implements the *Azure IoT* reference architecture and is composed of several components that exemplify some *Azure IoT* enabling technologies, illustrated in Figure 3.6. Not all components are required to benefit from the *MyDriving* solution. The core components are: (1) the mobile application (Android, iOS and Windows) for field gateway and UI; (2) the cloud gateway (IoT Hub); (3) the application services (app backend) deployed in the cloud, which follows the microservice RESTful architecture, providing APIs to store and query trip data; and (4) the data storage (a SQL Server database) deployed

in the cloud, which interacts with the microservices through the store and query APIs.

The *MyDriving* solution is ready for the whole application lifecycle management, from development to deployment and maintenance. Furthermore, the *MyDriving* solution implements real-time data handling through the connection of the cloud gateway, in-stream processing and business experience components. The solution executes stream analytics queries over the input data, forwarding data to be stored in both structured (SQL) and unstructured (blob) databases, and *Event Hubs*.

MyDriving also allows big data analytics and machine learning by consuming raw and processed input data that are stored as documents in the blob storage. Big data analytics is enabled through Apache Hadoop and MapReduce programming model for distributed parallel processing with HDInsight. In *MyDriving* it is applied to transform raw unstructured data into structured data, e.g., populating the trip points table (a point for each 3 seconds or 10 meters moved) and to produce inputs for retraining the machine learning component. The role of the machine learning component is to identify different styles of driving, assigning one of these styles to the driver. In summary, it analyzes how many, how often and how strong the driver hits the breaks, including acceleration and engine speed variation, and classify whether it is a good or a bad driving style. It uses the K-means algorithm for clustering, a classification model that is able to recognize the driving style of any trip.

3.3 Semantic Interoperability

The concept of “collaborative IoT” (C-IoT) [213] gives initial guidelines for future solutions towards IoT integration. According to C-IoT, the past and current IoT solutions focus on point-to-point vertical solutions, giving emphasis to integration of sensors (in the *Sensing* layer) to gateways (in the *Gateway* layer) to services provision (in the *Services* layer). Figure 3.8 (a) illustrates this approach. The C-IoT considers integration within each layer, thus focusing on reusing different sensors that might have different roles in their solutions, as well as gateways and services, as illustrated in Figure 3.8(b). This type of approach imposes integration challenges related to a number of aspects, such as data representation, protocol communication, security and privacy.

The four main general interoperability problems for seamless integration of IoT solutions are [47]:

- (1) the connection problem: enable devices to connect to each other and to the cloud;
- (2) the understanding problem: mutual understanding of exchanged data, assuring interpretation (semantic interoperability);

(3) the scalability problem: how to extend the involved resources of the IoT solution with minimal performance impact, how to maintain the quality of service (QoS) of the IoT solution;

(4) the adaptation problem: how to change/reconfigure the involved systems at runtime.

Figure 3.8. (a) current IoT solutions.
(b) C-IoT solutions
[213]

These problems are interconnected and a solution to address one impacts the others. For example, improving the semantic interoperability of an IoT solution can result in more verbose messages, because of additional metadata annotations, which can impact the connection performance [47].

In general, IoT platforms address syntactical interoperability by adopting standardized serialization formats, such as JSON and the Efficient XML Interchange (EXI) [214]. Several initiatives applied semantic technologies for the IoT [199, 215–217] and some targeted the semantic interoperability for IoT platforms within the *IoT European Platforms Initiative* (IoT-EPI)⁷¹. The *IoT European Research Cluster* (IERC) plays a major role for the integration of the solutions produced by these projects, aiming at defining a common vision of these IoT technologies at the European level [218]. Few industry-oriented solutions exist, such as the AWS IoT platform, which is leveraged by the recent AWS *Neptune triplestore* launch.

Table 3.1. H2020 Projects for IoT development that adopted semantic technologies

Acronym	Name	Semantic technologies
INTER-IoT ⁷²	Interoperability of Heterogeneous IoT Platforms	The project targeted interoperability of IoT platforms. It developed an “interoperable framework architecture for seamless integration of different IoT architectures present in different application domains”.

⁷¹ <https://iot-epi.eu/>

⁷² <https://cordis.europa.eu/project/rcn/199587/en>

		In particular, it developed the Inter-Platform Semantic Mediator (IPSM) for real time semantic translations of streaming data.
BIG-IoT ⁷³	Bridging the Interoperability Gap of the Internet of Things	Similar to INTER-IoT, this project targeted the interoperability of IoT platforms, but with a focus on monetization through a marketplace. This project is involved to the ongoing extension of the schema.org for the IoT. The IoT platforms that require to integrate with BIG-IoT framework should adhere to this ontology.
FIESTA ⁷⁴	Federated Interoperable Semantic IoT/cloud Testbeds and Applications	This project targeted the interoperability of IoT testbeds, which are experimentation environments that are supported by some IoT platforms. It highlights that “recent advances in IoT semantic interoperability provide a sound basis for implementing novel cloud-based infrastructures that could allow testbed-agnostic access to IoT data and resources”. FIESTA created an ontology ⁷⁵ with alignments to the <i>Machine-to-Machine Measurement</i> (M3) and <i>OpenIoT</i> ontologies [219]
sybmIoT ⁷⁶	Symbiosis of smart objects across IoT environments	Similar to INTER-IoT, this project targeted the interoperability of IoT platforms, emphasizing the industry-oriented IoT fragmented environment. This project developed a semantic middleware that allows the creation and integration of virtual IoT environments across various IoT platforms, allowing the communication among several APIs.
VICINITY ⁷⁷	Open virtual neighbourhood network to connect intelligent buildings and smart objects	This project also targeted the interoperability of IoT platforms, proposing the concept of virtual neighbourhood, “where users can configure their set ups, integrate standards according to the services they want to use and fully control their desired level of privacy”. This project is developing a semantic discovery platform that extends some ontologies, such as the Web of Things (WoT) ⁷⁸ .

A common element among these projects is the adoption of JSON-LD as data exchange syntax. Often, IoT semantic solutions use JSON-LD for message payload [216, 220, 221]. The proposal for a Semantic Web Stack for IoT recommends JSON-LD instead of XML because of performance improvements [216]. The payload of a JSON-LD message varies according to the semantic models chosen and how they are and, although JSON-LD is built to be lightweight, this choice can affect the time to transfer and process sensor data. Therefore, ontology engineering for IoT plays a major role in semantic interoperable IoT platforms.

3.3.1 Ontology Engineering for IoT

Semantic-driven IoT solutions rely on IoT ontologies to represent sensor data observations. Over the past few years, numerous IoT ontologies were proposed to improve the semantic interoperability of IoT artifacts, i.e., the common understanding of capabilities among platforms, devices, gateways, applications and networks involved in IoT solutions [193]. Guidelines for IoT ontology engineering

⁷³ <https://cordis.europa.eu/project/rcn/200833/en>

⁷⁴ <https://cordis.europa.eu/project/rcn/194117/en>

⁷⁵ <https://github.com/fiesta-iot/ontology>

⁷⁶ <https://cordis.europa.eu/project/rcn/199859/en>

⁷⁷ <https://cordis.europa.eu/project/rcn/200252/factsheet/en>

⁷⁸ <http://vicinity.iot.linkeddata.es/vicinity/>

were proposed by the *IERC Cluster Semantic Interoperability Best Practices and Recommendations* (IERC AC4), which were recently improved by the lessons learned from the FIESTA-IoT project, covering best practices and methodologies from the semantic web community [218]. These best practices are listed in Table 3.2, where tools and methodologies are recommended through a set of best practices grouped into five categories, aiming at responding the listed questions as a checklist.

Among the numerous IoT ontologies proposed in the literature, the W3C SSN/SOSA and the ETSI SAREF are the most prominent approaches [199, 217], which were rigorously developed during many years by ontologists and domain experts, being applied in several IoT use cases and supported by standardization initiatives. Although they cover similar concepts from the IoT domain, the motivation to create SAREF from SSN 1.0 was to apply an easier vocabulary aiming at industry-oriented IoT solutions. SSN/SOSA and SAREF are aligned with each other [74].

Table 3.2. Best practices and recommendations for IoT ontology engineering

Category	Id	Semantic technologies
Ontology & dataset publication (formats, serialization) - Did you validate the syntax of your ontology? - Does your ontology follow an unified language/nomenclature to describe sensor data? - Did you correctly use the SSN ontology?	1	Unified lexicon format with wrappers/translators to deal with heterogeneous formats (e.g., SenML/XML, SenML/JSON, CSV, Excel)
	2	Interlink datasets through unified RDF serialization (e.g., RDF/XML, JSON-LD, Turtle, Ntriples)
	3	Unique terminology for IoT data, linking synonyms terms that have the same meaning, such as precipitation and rain. M3 language/nomenclature can be used for this requirement
	4	Always apply syntax checks of semantic documents (e.g., RDF or OWL) to facilitate interoperability, which can be performed with RDF and OWL Validators, such as TripleChecker, which detects properties used but not declared in the TBox or a wrong format date
Metadata for vocabulary - Is your ontology referenced on LOV? - Does your ontology follow LOV recommendation? - Did you add ontology metadata to explain the purpose of the ontologies?	5	A minimal set of annotations on labels and comments should be provided by the ontology, including metadata to describe authors, versions and other provenance information. LOV is a useful tool that can find ontologies through these metadata, which was specialized by the LOV4IoT catalogue [222]
	6	Sharing datasets related to IoT solutions should consider dataset representation standards, such as W3C DCAT and VoID. With these standards, besides publishing datasets as RDF data, the ontology used to model the dataset and reasoning rules applied to enrich the dataset are also published in a standardized way
Ontology quality - Did you provide documentation attached to your ontology? - Is your ontology dereferenceable? - Did you follow tutorials and methodologies to design ontologies?	7	The easiness to lookup terms is a desired property, making the ontology dereferenceable. This is essential to automate tasks when analyzing ontologies
	8	Many ontologies have open issues and common pitfalls, which may be addressed though creating new elements that fix the problem, such as merging different concepts in a same class
	9	Bear in mind the importance of ontology reusability, thus, ontology engineering methodologies and tutorials should be adopted.

Ontology & dataset reuse - Did you try to reuse the existing ontologies or datasets catalogues or from semantic search engines? - Are your ontology and dataset referenced on such tools? - Is your ontology shared online?	10	“Sharing your ontology on the Web for reuse and discovery and referencing it on ontology catalogue (...) or semantic search engines (...) benefits both in increasing the reusability of the terms in the vocabulary as well as improving the quality of the vocabulary. Linked Open Data (LOD) provides tools (...) to share and reuse datasets”
	11	Enrich ontology documentation by adding precise definitions to the elements, preferably described in the ontology itself. This approach allows the automatic generation of documentation websites from the ontology
	12	“Aligning and reusing ontologies is challenging. When the ontology is reliable (e.g. maintained, standardized or popular), we should directly reuse the concepts or properties for existing ontologies (e.g., <i>ssn:device</i>). Otherwise, re-defining the same concept or property can be done and add <i>owl:equivalentClass</i> , <i>owl:equivalentProperty</i> ”
	13	SSN ontology is the de facto ontology for the IoT, therefore, it is necessary to align the ontology with SSN through extensions, which enables the validation with the W3C SSN validator
Namespace management - Did you choose a good namespace? - Did you check that your namespace is available? - Do you ensure the sustainability of the namespace?	14	“Choosing a good namespace which is still available. It can be done with the Prefix.cc tool.”
	15	“Unifying namespaces for ontologies and datasets. To automate tasks, it would be easier to unify namespaces. Frequently, we do not have labels or comments attached to the concepts and properties. We have to automatically parse the URI to get the label hidden in the URI. Having unified namespace will be easier to get the hidden labels”.
	16	“Ensuring the sustainability of the ontology namespace. Frequently, the namespace of the ontology depends on the server where it is hosted. When the server is not available anymore, accessing the namespace of the ontology will provide the error page not found. To avoid this issue, the persistent URL (PURL) enables to maintain the same namespace while changing the host server”

3.3.2 W3C Semantic Sensor Network (SSN/SOSA)

The W3C SSN/SOSA is considered an ontological foundation for the IoT, covering the application of diverse types of sensors, widely used by initiatives as OpenIoT [223] and INTER-IoT [193]. The first version of SSN (1.0) was launched in the beginning of 2000s, updated in 2011, and it provides a comprehensive framework to describe sensors, devices, observations, measurements and other terms, enabling reasoning of individual sensors and the connection of sensors, such as wireless networks. SSN 1.0 is grounded in a set of existing ontologies and standards, such as CSIRO, SWAMO, SEEK Extensible Observation, SemSOS and OGC SensorML [193].

SSN enables reasoning, which can facilitate the development of advanced applications, for example, by reasoning about sensor measurements, considering constraints as power restriction and limited memory. The main concept of SSN 1.0 is the *ssn:SensingDevice*, which is a *ssn:Sensor* that reports measurements and observations of real world phenomena; and is different from other types of devices, such as actuators (*sosa:Actuator*). SSN 1.0 inherits classes and properties from the foundational ontology DOLCE-UltraLite (DUL).

Figure 3.9. Main modules of W3C SSN/SOSA⁸¹

SSN version 2.0 was released in 2017, in which SSN 1.0 has been refactored in two main modules: the *Semantic Sensor Network* (SSN)⁷⁹ and the *Sensor, Observation, Sample, and Actuator* (SOSA)⁸⁰ ontologies. “SOSA provides a lightweight core for SSN and aims at broadening the target audience and application areas that can make use of Semantic Web ontologies. At the same time, SOSA acts as minimal interoperability fallback level, i.e., it defines those common classes and properties for which data can be safely exchanged across all uses of SSN, its modules, and SOSA”⁸¹. Figure 3.9 illustrates these two modules taking advantage of DUL, as well as core ontologies for units of measurement (O&M) and provenance (PROV), through ontology alignments using *owl:imports* predicate. The SSN 2.0 specification also provides ontology alignments to SSN 1.0.

Figure 3.10 gives an overview of the SSN/SOSA main elements, which are organized into eight categories: *Deployment*, *System*, *SystemProperty*, *Feature*, *Condition*, *Procedure*, *Observation/Actuation/Sampling* and *Result*. While the elements in the upper part are part of SSN, the elements on the lower part are part of SOSA.

The central element of SSN/SOSA is the *sosa:Observation*, which represents the action of carrying out an observation procedure to calculate a value of a property of a *FeatureOfInterest*, linked to a *Sensor* to describe the *Observation* structure and behavior, linked to an *ObservableProperty* to describe the result, and linked to a *FeatureOfInterest* to describe which property was associated with it. Therefore, *sosa:Observation* is made by a sensor (*sosa:madeBySensor*), such as a wind sensor, that observes properties (*sosa:observedProperty*), such as wind direction and speed, measured in a specific time instant (*sosa:resultTime*); this result time can be different

⁷⁹ <http://www.w3.org/ns/ssn/>

⁸⁰ <http://www.w3.org/ns/sosa/>

⁸¹ <https://www.w3.org/TR/vocab-ssn/>

from the phenomena time (*sosa:phenomenonTime*) w.r.t. the *sosa:FeatureOfInterest* applied to a result (*sosa:hasResult*) that has a value (usually numeric).

Figure 3.10. Main classes and object properties of SSN/SOSA⁷⁹

The time predicate *sosa:phenomenonTime* has *time:TemporalEntity* as range, an element defined in the W3C Time Ontology⁸², which is an ontology of “temporal concepts, for describing the temporal properties of resources”, providing “a vocabulary for expressing facts about topological (ordering) relations among instants and intervals, together with information about durations, and about temporal position including date-time information”.

SSN provides the *ssn:System* class and the *ssn:hasSubsystem* object property for the realization of the composition pattern of systems. Similarly, SOSA allows for the representation of a device deployment as a *sosa:Platform*, which can be a *sosa:Sensor* or a *sosa:Actuator* or a *sosa:Sampler* or another *sosa:Platform*, and its assembly can be represented through the *sosa:hosts* object property.

A *ssn:System* has *ssn:Deployment(s)*, represented through *ssn:hasDeployment*, and a *sosa:Platform* is in a deployment (*ssn:inDeployment*), “meaning that the deployed *Systems* of the *Deployment* are hosted on the *Platform*”. Although the specification does not define this explicitly, we understand that *ssn:hasSubsystem* is a composition relationship, i.e., the part only exists if the whole exists, while *sosa:hosts* is an aggregation. The main class and property definitions that are used in this thesis are listed in Table 3.3, along with their super classes and restrictions.

When using the O&M ontology, the *sosa:Result* can be specialized as an *om:Measure* or *om:Point*. For geo-location representation, SSN/SOSA recommends

⁸² <https://www.w3.org/TR/owl-time>

to reuse the GeoSPARQL approach [224], an OGC standard⁸³, which “provides a flexible and relatively complete platform for geospatial objects, that fosters interoperability between geo-datasets. To do so, these entities can be declared as instances of *geo:Feature* and geometries can be assigned to them via the *geo:hasGeometry* property”. GeoSPARQL is widely used to represent geo-information and is mapped to lexicon standards, such as the OGC Geography Markup Language (GML), IETF GeoJSON and Google Keyhole Markup Language (KML) [225, 226].

Table 3.3. Main elements of SSN/SOSA used in this thesis

Element	Semantic technologies
sosa:Sensor	<i>rdfs:subClassOf</i> <i>ssn:System</i> Restrictions: <i>ssn:implements</i> <i>MIN 1</i> <i>sosa:Procedure</i> <i>sosa:observes</i> <i>ONLY</i> <i>sosa:ObservableProperty</i> <i>ssn:detects</i> <i>ONLY</i> <i>ssn:Stimulus</i> <i>sosa:madeObservation</i> <i>ONLY</i> <i>sosa:Observation</i>
ssn:System	<i>rdfs:subClassOf</i> <i>owl:Thing</i> Restrictions: <i>sosa:isHostedBy</i> <i>ONLY</i> <i>sosa:Platform</i> <i>ssn:implements</i> <i>ONLY</i> <i>sosa:Procedure</i> <i>ssn:hasSubSystem</i> <i>ONLY</i> <i>ssn:System</i> <i>inverse Of</i> <i>ssn:hasSubSystem</i> <i>ONLY</i> <i>ssn:System</i> <i>ssn:hasDeployment</i> <i>ONLY</i> <i>ssn:Deployment</i>
ssn:implements	<i>rdf:type</i> <i>owl:ObjectProperty</i> <i>owl:inverseOf</i> <i>ssn:implementedBy</i>
sosa:Procedure	<i>rdfs:subClassOf</i> <i>owl:Thing</i> Restrictions: <i>ssn:hasInput</i> <i>ONLY</i> <i>ssn:Input</i> <i>ssn:hasOutput</i> <i>ONLY</i> <i>ssn:Output</i> <i>ssn:implementedBy</i> <i>ONLY</i> <i>ssn:System</i>
ssn-sys:OperatingRange	<i>rdfs:subClassOf</i> <i>ssn:Property</i> Restrictions: <i>ssn-system:hasOperatingProperty</i> <i>ONLY</i> <i>ssn-system:OperatingProperty</i> <i>ssn-system:inCondition</i> <i>ONLY</i> <i>ssn-system:Condition</i> <i>ssn-system:inCondition</i> <i>MIN 1</i> <i>ssn-system:Condition</i> <i>owl:inverseOf</i> <i>ssn-system:hasOperatingRange</i> <i>ONLY</i> <i>ssn:System</i>
sosa:hosts	<i>rdf:type</i> <i>owl:ObjectProperty</i> <i>owl:inverseOf</i> <i>sosa:isHostedBy</i>
ssn:hasSubSystem	<i>rdf:type</i> <i>owl:ObjectProperty</i>

The SSN/SOSA specification provides a set of examples⁸⁴, such as the representation of a coal oil point reserve, an electric consumption of a smart home (an apartment), a tree height measurement, a seismograph, a wind sensor spinning cups, an IP68 Smart Sensor and an iPhone barometer. In this last example, the iPhone barometer is composed by a BMP282 atmospheric pressure sensor and the

⁸³ <http://www.opengeospatial.org/standards/geosparql>

⁸⁴ <https://www.w3.org/TR/vocab-ssn/#examples>

representation of an observation from this sensor is exemplified in Code Snippet 3.2, which uses the Turtle syntax to represent the triples: an observation (346344) that observed the atmospheric pressure property, has earth atmosphere as feature of interest, is made by a sensor (BMP282), has the simple result of 1021.45 hPa at a specific moment.

Code Snippet 3.2. Observation example of an atmospheric pressure sensor

```
<Observation/346344> rdf:type sosa:Observation ;
  sosa:observedProperty <sensor/35-207306-844818-0/BMP282/atmosphericPressure> ;
  sosa:hasFeatureOfInterest <earthAtmosphere> ;
  sosa:madeBySensor <sensor/35-207306-844818-0/BMP282> ;
  sosa:hasSimpleResult "1021.45 hPa"^^cdt:ucum ;
  sosa:resultTime "2017-06-06T12:36:12Z"^^xsd:dateTime .
```

3.3.3 ETSI Smart Appliance Reference Ontology (SAREF)

Recently, ETSI along with the EC, the Netherlands Organisation for Applied Scientific Research (TNO) and the Universidad Politecnica de Madrid (UPM), developed the Smart Appliances REference (SAREF)⁸⁵ ontology [227, 228]. At first, this ontology was built as a reference model targeting smart appliance solutions for the smart home domain, e.g., smart ovens and refrigerators [227]. However, SAREF has evolved to cover the IoT domain in general, being created in close interaction with the smart home market [228]. SAREF was acknowledged by the EC as the “first ontology standard in the IoT ecosystem, and sets a template and a base for the development of similar standards for the other verticals to unlock the full potential of IoT” [117]. The SAREF ontology provides building blocks that enable re-utilization of different parts of the ontology according to specific requirements. It is grounded on 47 “semantic assets”, i.e., standards, proprietary data models, protocols and other ontologies, such as SSN.

The motivation behind SAREF was that the market would continue to be too fragmented and powerless without a (protocol-independent) semantic layer that enables interoperability among the various smart appliances from different manufacturers. To that end, SAREF was created with the intention to interconnect different platforms, supporting the data exchange with different protocols.

SAREF facilitates the matching of existing assets, since it was developed based on standards, ontologies, data models and protocols of the IoT domain, providing a high-level mapping of them, available in SAREF’s first interim study report. One of these assets is SSN (the first version), which inspired the definition of the main elements of SAREF, namely *Device*, *Sensor*, *Unit of Measure* and *Time/Duration*,

⁸⁵ <http://ontology.tno.nl/saref/>

according to the high-level mappings provided in the SAREF initial documentation [227]. A *saref:Device* (e.g., a *saref:Sensor*) represents tangible objects designed to accomplish one or more functions in diverse types of locations (e.g., households and buildings).

For example, a *saref:Sensor* *saref:hasFunction* *saref:Function* of type *saref:SensingFunction*. The SAREF ontology offers a list of basic functions that can be combined towards more complex functions in a single device. For example, a *saref:Switch* can offer an *saref:Actuating* function of type “switching on/off” and a *saref:SensingFunction* of type *saref:LightSensor*, so if there is illumination in the environment then the switch turns off the light. Each *saref:Function* has some associated *saref:Command*, which can also be picked up as building blocks from a list. For example, the “switching on/off” is associated with the commands “switch on”, “switch off” and “toggle”. Depending on the *saref:Function(s)* it accomplishes, a device can be found in some corresponding *saref:State(s)*. Figure 3.11 illustrates these elements.

Figure 3.11. Core elements for representing device measurements in SAREF

The composition pattern of a *saref:Device* is represented with the *saref:consistsOf* self-relationship. For example, the WM30 wind sensor (a *saref:Device*) can be defined as a composition of wind direction and wind speed sensors. A *saref:Device* measures a specific property, represented by the object property *saref:measuresProperty* to a *saref:Property*. For example, a *saref:SmokeSensor* (a *saref:Sensor*) measures *saref:Smoke* (*saref:Property*), and analogously a *saref:WindSensor* measures *saref:Wind*. Regarding a measurement observed by a sensor in time, SAREF represents it through the *saref:makesMeasurement* object property of a *saref:Device* to *saref:Measurement(s)*, representing the relation between a device and the measurements it makes, ontologically equivalent to *sosa:Observation*. A *saref:Measurement* element is responsible to represent a measurement value (*saref:hasValue*), the unit of measure of this value (*saref:UnitOfMeasure*) and the property under observation (*saref:Property*). For example, a temperature

measurement (*saref:Measurement*) of 10 (*saref:hasValue*) Celsius degrees (*saref:UnitOfMeasure*) that refers to the temperature of a room (*saref:Property*).

A *saref:Device* offers a *saref:Service*, which represents a *saref:Function* to a network that makes the function discoverable, registerable and remotely controllable by other devices in the network. A *saref:Service* can represent one or more functions. A *saref:Service* must specify the *saref:Device* that offers the *saref:Service*, the function(s) to be represented, and the (input and output) parameters necessary to operate the service, supported by the ontology alignments with OneM2M ontology [199]. The main class and property definitions that are used in this thesis are listed in Table 3.4, along with their super classes and restrictions.

Table 3.4. Main elements of SAREF used in this thesis

Element	Semantic technologies
<i>saref:Sensor</i>	<i>rdfs:subClassOf</i> <i>saref:FunctionRelated</i> Restrictions: <i>saref:hasFunction</i> some <i>saref:SensingFunction</i>
<i>saref:FunctionRelated</i>	<i>rdfs:subClassOf</i> <i>saref:Device</i>
<i>saref:Device</i>	Restrictions: <i>saref:controlsProperty</i> only <i>saref:Property</i> <i>saref:hasFunction</i> min 1 <i>saref:Function</i> <i>saref:hasManufacturer</i> max 1 <i>saref:hasModel</i> max 1 <i>saref:hasState</i> only <i>saref:State</i> <i>saref:hasTypical consumption</i> only (<i>saref:Energy</i> or <i>saref:Power</i>) <i>saref:offersOnly</i> <i>saref:Service</i> <i>saref:hasDescription</i> max 1 <i>saref:consistsOf</i> only <i>saref:Device</i> <i>saref:measuresProperty</i> only <i>saref:Property</i> <i>saref:isUsed for</i> only <i>saref:Commodity</i> <i>saref:hasProfile</i> only <i>saref:Profile</i> <i>saref:makesMeasurement</i> only <i>saref:Measurement</i> <i>saref:accomplishes</i> min 1 <i>saref:Task</i>
<i>saref:hasFunction</i>	<i>rdf:type owl:ObjectProperty</i>
<i>saref:SensingFunction</i>	<i>rdfs:subClassOf</i> <i>saref:Function</i> Restrictions: <i>saref:hasSensingRange</i> some <i>saref:Measurement</i> <i>saref:hasSensorType</i> only <i>saref:Property</i> <i>saref:hasCommand</i> only <i>saref:GetSensingDataCommand</i>
<i>saref:hasSensingRange</i>	<i>rdf:type owl:ObjectProperty</i>

Numerous ontology-oriented IoT solutions were developed to accomplish specific domain applications, such as air-conditioning and weather reporting systems [199, 217], which are also called IoT vertical markets. As described above, SAREF is pointed by the EC as the main reference IoT ontology that provides an extensible characteristic based on ETSI standardization procedure for vertical markets [192]. SAREF standardized extensions to address these vertical markets [192] include smart energy (SAREF4Ener), environment (SAREF4Envi) and buildings (SAREF4Bldg) [229]. To avoid semantic interoperability issues when

extending SAREF, this ETSI standard procedure should be followed whenever a SAREF extension is submitted for acceptance. This procedure is based on the following best practices:

- (1) the extension is designed according to clarity, coherence, extensibility, minimal encoding bias and minimal ontological commitment criteria;
- (2) relevant stakeholders in the domain of interest should be involved in its development process;
- (3) the group/community that creates the extension should be committed to contribute to its maintenance;
- (4) it should not add concepts that are already present in SAREF or other extensions;
- (5) it needs to be properly documented and published.

Until 2021, ETSI plans to develop and standardize more four extensions: for automotive (SAREF4auto), e-Health and aging-well (SAREF4ehaw), wearables (SAREF4wear) and water (SAREF4watr)⁸⁶. SAREF4Envi⁸⁷ enables the representation of data about light pollution, exemplified with a photometer device that monitors light magnitude, sending the measurements to a server. SAREF4Envi messages serialized as a JSON-LD are further used in this thesis for validation purposes, which represent a photometer device providing measured data and other information. The measurements, linked through the *saref:makesMeasurement* property, represent the data underlining continuous monitoring. In this extension, the W3C WGS84 ontology⁸⁸ is used to represent the device position through the *geo:location* predicate, which is “the relation between something and the point, or other geometrical thing in space, where it is”.

3.3.4 Semantic Translations

A challenge to enable semantic integration between IoT artifacts relying on different ontologies, e.g., SSN/SOSA and SAREF, is how to translate from one ontology to another (semantic translations). We identified that although SAREF conceptualization was based on SSN 1.0, no ontology alignments or semantic translation rules were defined. Finding and mapping the correspondences between entities (atomic) or groups of entities and sub-structures (complex) of different ontologies are challenging tasks that require appropriate tooling support. Tools for ontology alignment were surveyed in [230], which also describes the conceptual differences amongst alignment, matching, merging, mapping and semantic translations.

⁸⁶ https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?wki_id=51404

⁸⁷ <http://ontology.linkeddata.es/publish/saref4envi/index-en.html>

⁸⁸ <https://www.w3.org/2003/01/geo/>

A common approach for semantic translations was adopted by the INTER-IoT project through the Inter-Platform Semantic Mediator (IPSM)⁸⁹, which is used to (1) deal with non-semantic artefacts, by lifting their semantics to OWL-based representation, (2) develop a central modular ontology, consisting of a core IoT ontology and domain specific modules, (3) generate alignments between ontologies of communicating artefacts and the central ontology, (4) use the alignments for translations, and (5) establish a multi-channel (1-1, 1-many, many-1) communication architecture that eases translations in all needed contexts. While complete ontologies are used to “build semantic understanding”, only conversation-specific alignments are stored and used for actual translations.

Code Snippet 3.3. Semantic translation example with IPSM language

```
<align:Cell rdf:about="http://www.inter-iot.eu/sripas#Cell3">
  <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    var:CTA a saref:Device ;
    saref:consistsOf var:CTB .
    var:CTB saref:measuresProperty var:CTC .
  </align:entity1>
  <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
    var:CTA a sosa:Platform ;
    sosa:hosts var:CTB .
    var:CTB sosa:observes var:CTC .
  </align:entity2>
  <align:relation>=</align:relation>
</align:Cell>
```

IPSM provides its own alignment format⁹⁰, based on an alignment API with a declarative ontology alignment language (XML-based), inspired by the *Expressive and Declarative Ontology Alignment Language* (EDOAL)⁹¹, to represent interconnections between semantic data of multiple ontologies. Code Snippet 3.3 illustrates an example of an alignment from SAREF to SOSA, transforming the *saref:Device* to a *sosa:Platform* element according to the patterns of some linked object properties (*saref:consistsOf*, *saref:measuresProperty*, *sosa:hosts* and *sosa:observes*). IPSM utilizes alignment files and provides a multi-channel environment for any artifact. Pairs of uni-directional alignments between the INTER-IoT central ontology, coined as Generic Ontology for IoT Platforms (GOIoTP)⁹² and artifact ontology are used to translate messages to and from the central ontology. This enables connection of new artifacts without affecting the existing channels, and requires each participant to provide only a pair of alignments. While complete

⁸⁹ <https://docs2.inter-iot.eu/docs/ipsm/latest/>

⁹⁰ <https://docs2.inter-iot.eu/docs/ipsm/latest/Configuration/Alignment-format/IPSM-alignment-format/>

⁹¹ <https://ns.inria.fr/edoal/1.0/>

⁹² <https://docs.inter-iot.eu/ontology/#iiot-Location>

ontologies are used to build semantic understanding, only conversation-specific alignments are stored and used for actual translations. Ontology alignments and translation channels can be managed through the REST Manager.

While the semantic translation mechanism of the IPSM approach covers the semantic interoperability in terms of common understanding of shared context data, which corresponds the perception level of the SA theory (presented in Section 2.1), other developments apply similar semantic translation techniques to transform RDF to CEP languages, such as EPL, in the context of IoT, aiming at automatically generating statements at runtime [231]. Code Snippet 3.4 illustrates in a SPARQL-like query how to represent the detection rule of “smoke and abnormal temperature within one room” situation type, which can be detected by readings (or observations) from smoke and temperature sensors. Code Snippet 3.5 represents this very same rule as an EPL statement that was produced with model transformations. The general idea is to use CEP technology for real-time processing, while using RDF to share the situation type rules. The approach based on model transformations has been applied by some previous work on semantic rule-based CEP [179, 232-235].

Code Snippet 3.4. SPARQL-like rule to detect fire based on smoke and temperature [231]

```
NAMED EVENT :FireEvent {
  MATCH :AbnormalTemperaturEvent -> :SmokeDetectionEvent WITHIN (5m)
  IF {
    EVENT :AbnormalTemperaturEvent { ?tmpSnsLoc a :Location .
                                         ?tmpSnsLoc :hasValue ?v }
    EVENT :SmokeDetectionEvent { ?smkSnsLoc a :Location .
                                   ?smkSnsLoc :hasValue ?v }
  }
}
```

Code Snippet 3.5. Fire rule represented in EPL, produced by model transformations [231]

```
select * from pattern
[every a=AbnormalTemperaturEvent -> b=SmokeDetectionEvent (v=a.v)
where timer:within(5 min)]
```

3.4 Concluding Remarks

An IoT platform represents a container of technologies that enables the rapid and reliable development of IoT solutions, providing suites of components to software engineers. The interoperability of IoT platforms, including components for third-party integration, rely on the main enabling technologies that most IoT platforms provide, which are: (1) publish/subscribe middleware (context and service brokers) for real-time data exchange, with components for cloud and at device levels; (2) RESTful microservices architecture for application integration via APIs; (3) CEP engines, including stream analytics, for situation identification based on

temporal predicates. (4) Data management systems for historical data storage, warehousing and analytics, including business intelligence and machine learning.

These technologies support the four main layers of an IoT platform architecture: device, network, middleware and application levels. This chapter illustrates how these technologies work by describing the components of the Microsoft Azure IoT platform along with its main use case (*MyDriving*), which is latter used in this thesis for validation purposes. The main components related to interoperability of Azure are: (1) IoT Hub, a context/service broker that support HTTP, MQTT and AMQP protocols, supports JSON as message payload and can scale horizontally and vertically via cloud infrastructure services (IaaS); (2) Field gateway (or IoT edge), a service that is able to pre-process data (e.g., aggregate) from close sensors, such as the *MyDriving* mobile application, which receives telemetry data from car sensors via Bluetooth in high-frequency, computes new features and transmits them to the cloud in lower frequencies. The concept of *edge computing* is also recent and represents the idea of providing more powerful technologies for the devices that are in the edge and have the processing capacities required, such as smartphones and laptops. *Azure* recently developed an IoT edge reference implementation derived from the field gateway of *MyDriving* solution.

In the IoT context, the lack of semantic interoperability among IoT solutions is an issue that several works tackled. The semantic interoperability of IoT platforms is being exploited by recent research- and industry-oriented initiatives, such as the INTER-IoT and BIG-IoT projects. Literature shows that ontology engineering plays a determinant role for developing high-quality and reusable IoT ontologies, requiring to follow a development methodology that considers best practices specific for the IoT. The most basic best practice is to adopt a standardized semantic model as, at least, a reference model. The main standardized IoT ontologies are the W3C SSN/SOSA and the ETSI SAREF. This chapter provided an overview of the main classes and properties of these ontologies, which are later used in the experiments. Although SAREF was recently developed (2015) based on SSN 1.0, we have identified that the specific rule definitions for alignments between them do not exist.

Ontology alignments and semantic translations are mechanisms that can be used to address the interoperability of systems that rely in different ontologies. This chapter also described the IPSM tool, which is the INTER-IoT component for semantic translations of data streams, which is later used for the experiments. IPSM offers a REST API that is able to receive and respond high-throughput data described as JSON-LD, translating from the input to the output RDF according to pre-defined set of rules. Therefore, this type of approach plays a determinant role to improve the semantic interoperability of the different IoT platform components, enabling IoT solutions to collaborate at all levels.

The adoption of JSON-LD as RDF serialization for data exchange is frequently found in the literature. However, the migration of JSON-based approaches to JSON-LD may cause additional message overhead because of the JSON-LD tokens. Moreover, it is necessary to analyze the impact on processing performance of JSON-LD (de)serialization by existing libraries, such as the JSON-LD processor and API implementation in JavaScript [236].

Review on EWS interoperability

Chapter 2 provided an overview of the problem context, describing the application domain (DRR), general aspects of interoperability, common semantic approaches and current research on situation-aware applications. The concept of IoT platform is explained in Chapter 3, describing its enabling technologies and how semantic technologies are used to improve IoT platforms' interoperability. This Chapter uses this background knowledge to support a systematic literature review about the interoperability of EWSs, aiming at scrutinizing the problem definition and, thus, detailing the challenges addressed by this thesis. Section 4.1 presents the knowledge questions related to the research goals, the methodology used for this literature review, the search strategy, as well as the data extraction and synthesis process. Section 4.2 presents the results, including a quantitative analysis. A qualitative analysis describes the historical perspective of EWSs and their related projects, analyzing the lessons learned from existing reviews and detailing the common EWS requirements. Section 4.3 revisits the *standard-based* EWS architecture and describes its evolution with semantic and IoT technologies, called *semantic IoT EWS* approach. Section 4.4 describes some of the current emergency data representation approaches, including lexicon and semantic models, pointing out their open issues. Section 4.5 revisits the challenges listed in Chapter 1, mapping them to a set of requirements that the approach introduced in this thesis (SEMIoTICS) aims at addressing. Finally, Section 4.6 presents the concluding remarks w.r.t the main lessons learned from this literature review.

4.1 Interoperability of Early Warning Systems

Enabling smart behavior in emergency scenarios for risk detection using sensors and actuators relies on the successful integration of the components participating in an EWS. Therefore, interoperability of these components plays a major role to achieve smart emergency services [14]. For example, in a public health context, a

system can detect the beginning of a disease outbreak based on data from mosquito traps [237] equipped with insect classification sensors, e.g., [238, 239], integrated to clinical data, i.e., electronic health records from hospitals (diagnosed patients and laboratory exams), environmental monitoring sensors and airline travelers datasets. A smart behavior in a smart home near the affected area is the actuation of turning on anti-mosquito appliances, such as classic insecticide device, LDE light bulb, thermal lantern and high-intensity black light. Smart sirens deployed in high risk areas could receive messages from such a system to alert the public.

The need of improving EWSs interoperability towards seamless integration is discussed in Chapter 2, in lights of the Sendai framework requirements for MHEWS [29].

Sendai requirements for EWS coordination and collaboration through a holistic approach impose efforts in all interoperability aspects, especially the semantic level, since the semantics inherited in messages exchanged among different components of an EWS, and among different EWSs, are crucial for the common understanding of the data. The International Network for MHEWS (IN-MHEWS) [22], a multi-stakeholder partnership related to EWSs integration, explicit the need on inventorying EWSs architectures that are implemented and in operation worldwide.

Therefore, this requirement is aligned to our goal of developing a comprehensive and non-biased synthesis of existing work about EWS interoperability for DRR. Our knowledge goal is to describe and explain the phenomena of DRR through the improvement of interoperability of EWS. Therefore, our prediction goal is to better forecast and reduce disaster risks by using EWSs. According to the DSM template for design problem, the top-level design goal of this thesis is:

Improve the semantic interoperability of IoT EWSs.

By designing the SEMIoTICS framework.

That enables semantic integration of a variety of components for processing in time- and safety-critical high quality situation aware applications.

In order to efficiently and effectively reduce the risks of disasters.

The main actors involved in global DRR supported by EWSs, which participate in the IN-MHEWS initiative, are the UNISDR, the World Health Organization (WHO), the World Meteorological Organization (WMO), the United Nations Development Programme (UNDP) and the European Centre for Disease and Control (ECDC). Besides these institutional parties, different types of stakeholders in DRR include the community, the national institutes (e.g., operation centers and military) and the regional institutes (e.g., health units, police and firefighter departments, civil defense), the international bodies (e.g., UN agencies), research

organizations (e.g., universities), and nongovernmental (NGO) organizations (e.g., Greenpeace). Other types of stakeholders include the international consortiums and communities that create and maintain standards for interoperability, as well as companies that provide ICT solutions for DRR, i.e., they develop and integrate systems, some compliant to interoperability standards to foster integration with other systems.

4.1.1 Systematic Literature Review Methodology

The Design Science Methodology (DSM) [64] recommends the execution of literature reviews to support the research problem investigation phase of the design engineering cycle. The research methodology adopted in this study is the systematic literature review (SLR) for software engineering [63]. This SLR applied an unbiased search strategy to respond the knowledge questions by finding “as many primary studies relating to the research questions as possible” [63]. Therefore, the goal of this SLR was to investigate the current problems and improvement opportunities in the context of interoperability of EWS for DRR supported by IoT technologies.

The SLR methodology has three phases. The SLR planning phase describes the guidelines followed, detailing the stakeholder goals and the reviewer objectives, defining the review protocol that includes the data sources used, the search strategy, the inclusion and exclusion criteria, and the initial data analysis. The conducting phase supports the identification of research by applying a cyclic procedure, first assessing primary studies and using the outcomes to detail the search strategy, i.e., tune search queries, thus, improving the study selection criteria (both inclusion and exclusion). Publication bias is avoided by enforcing bibliography management and documenting the search executions along with their results. Quality instruments were used to improve the study selection process by responding the quality checklist for quantitative studies of the SLR methodology, such as checking if the survey method introduced significant bias and checking the reliability of inclusion decisions. The outcomes provide the data synthesized to be reported in the final phase, which generates the final report.

4.1.2 Knowledge Questions

The knowledge questions addressed in this SLR are presented in Table 4.1. In DSM, knowledge questions are empirical questions that “do not call for a change in the world but ask for knowledge about the world as it is. The answer is a proposition, and when we try to answer a knowledge question, we assume that there is one answer only” [64]. The knowledge questions (Table 4.1) were derived from the top-level design goal.

A knowledge question can be considered as open (exploratory) or closed (hypothesis-testing) question, and classified as effect, trade-off, sensitivity or requirements satisfaction questions. In this SLR we focused on effect and trade-off questions, thus, the effects produced by the interaction of the artifact with the context and the trade-off with an alternative artifact. For example, KQ02 is an open effect question, asking for the impact of artifacts (project results) in the context (emergency interoperability), while KQ05 refers to trade-off questions, asking for the impact of similar artifacts (e.g., W3C SSN/SOSA and ETSI SAREF) in the context (EWS interoperability).

KQ1 asks for knowledge about existing IoT EWSs, while KQ2 asks for knowledge about interoperability initiatives of EWSs, which also includes the interoperability of emergency systems used as data sources and/or targets. KQ03 asks for knowledge w.r.t architectural aspects of EWSs, especially on how the components can be leveraged by semantic and IoT technologies. KQ04 asks for knowledge about the existing data representations, which are responsible by the semantics of messages exchanged. Finally, KQ05 asks for knowledge about the main open issues related to semantic interoperability of EWSs.

Table 4.1. Knowledge questions used in this systematic literature review

Id	Question	Description
KQ1	Are there surveys or literature reviews about EWSs?	Identify the volume of potentially relevant existing reviews and surveys for quantitative analysis. The knowledge goal of this question is to identify the terminology used in the literature (key concepts) regarding EWS, such as the classification of hazardous events (disasters), situation awareness and the most relevant EWSs used for DRR.
KQ2	What are the main initiatives (e.g., projects) for emergency interoperability?	The knowledge goal of this question is to understand the state-of-art of interoperability for emergency services. Questions derived are: a. What interoperability problems of EWS are reported in the literature? b. What are their possible causes or mechanisms that produce the problems?
KQ3	What are the common EWS architectures and their components' functions?	The knowledge goal of this question is to understand the state-of-art of EWS architectures and the role of IoT technologies? Questions derived are: a. What are the common components of the architectures? b. What are the stakeholders involved? c. What problems do stakeholders experience with them?
KQ4	What are the data representation mechanisms used for interoperability of emergency services that an EWS may interface?	What data representation approaches (e.g., data models, lexicon standards, protocols, semantic models and ontologies) and standardization initiatives (e.g., W3C, OGC, ETSI, OASIS) are used for DRR? a. How do they represent a situation in their data/message models? b. How do they characterize pre-disaster situations?
KQ5	What are the main open issues regarding EWS interoperability and the current solutions?	What are the mechanisms that produce the open issues and what treatments make use of the most well-known standardization initiatives, such as W3C SSN/SOSA, ETSI SAREF, OASIS EDXL and HL7 FHIR? What other initiatives should be considered as domain-relevant community standards?

The search strategy includes the derivation of queries from these knowledge questions to be used in the search engines provided by the data sources. The execution of these queries resulted on result items (samples) to be analyzed.

Therefore, this SLR is a sample-based research where each sample is studied as whole. The sample randomization was performed according to the inclusion and exclusion criteria, considering that one population element may be selected more than once. A sampling frame was defined as a list of population elements, i.e., the definitions presented in Chapters 2 and 3.

4.1.3 Search Strategy

The data sources for knowledge acquisition include scientific theories, design specifications, practical and common sense knowledge. These data sources were classified in four categories, according to DSM: scientific, technical and professional literature, and oral communication. This list includes eight digital libraries as scientific literature: *ACM*, *Science Direct*, *Springer*, *Google Scholar*, *IEEEExplore*, *Citeseer*, *Scopus* and *Wiley*. Technical specifications include reports and specifications from the standardization institutes, which are presented in Appendix B (main stakeholders in EWS interoperability). In particular, the *Cordis* system⁹³ was used to search for related projects funded by the EC.

The inclusion criteria prescribes the exclusive inclusion of works published in the data sources listed, and if two or more works describe the same approach, the latest or more comprehensive one is included. In addition, only works related to at least one of the concepts: DM (DRR and SA), interoperability and EWS were included. Surveys and reviews related to integration approaches were also considered. The exclusion criteria prescribes the removal of works that cannot be fully accessed using the *University of Twente* credentials, as well as paper conferences published before 2005 and workshop papers before 2010. The main reason is that if the research reported in these papers are still relevant and/or were improved, then the research would be published in a journal or in a more recent conference/workshop paper. In the primary studies analysis we noticed that several papers were in other fields rather than Computer sScience, e.g., Sociology and eEconomy. Therefore, we excluded all results where the main theme was not related to Computer Science.

A ranking method was applied to automatically classify the importance of the result to this SLR, giving a higher weight according to the source, publication date and publication type. The source weight order is journal, book chapter or magazine, conference proceedings and other. The publication date weight order is 2018, 2016, 2014, 2010 and below. The publication type weight order is full-paper, position paper and short paper.

The knowledge questions listed in Table 4.1 were transformed in queries executed in rounds, where each round has a list of search strings to be performed

⁹³ <https://cordis.europa.eu/>

in each digital library. A cyclical process was executed to create more detailed search strings to respond questions, such as KQ5, according to the output of the previous executions. For example, this string was created to provide information regarding the application of EDXL in the H2020 RESCUER project:

(“EDXL” OR “Emergency Data Exchange Language”) AND “RESCUER” AND SS02

4.1.4 Data Extraction and Synthesis

The data extraction process, which is a sub-process of the conducting phase, followed the steps below for each round:

1. During identification step, each query is executed in each data source. The total number of results is annotated;
2. During framing step, all duplicates are removed and items referring the same research are merged;
3. In the resulted list, each item is checked for compliance with the inclusion and exclusion criteria by inspecting the publication metadata, and, in some cases the abstract. If it complies, then it is marked to be included.
4. Calculate the weight of each work marked to be included. The list is ordered by this weight (descending) and each item is mapped to the data source(s) where it was found;
5. The ranked list of results is pruned, including only the top 50 results to be analyzed and discarding the rest;
6. During assessment step, each item of the list is preliminary analyzed (abstract, introduction and conclusion), being tagged with an importance level for this SLR, reflecting whether it should be fully inspected or discarded;
7. Finally, during the inclusion step, each item selected to be fully inspected is entirely revised and a synthesis is written, linking the publication to the respective knowledge question(s).

The output of this search strategy was used to review the SLR protocol and to select the basic terminology for this SLR. The identification of primary studies showed that the most relevant types of stakeholders in the context of EWS interoperability are national and international organizations, such as governmental and non-governmental (no-profit) organizations, standardization consortiums and communities, and IT solution providers.

4.2 Result Analysis

Figure 4.1 shows the results of this SLR process. The identification phase of our SLR process resulted in 2917 sources (most of them papers and book chapters) from digital libraries and 261 from other sources, according to the inclusion criteria. The framing phase resulted on 3082 merged, having 2715 discarded by our exclusion criteria, resulting on 367 works assessed. 75 technical specifications (reports and standards) and 292 papers, from 18 journals and 12 conference proceedings, were fully revised.

Figure 4.1. Studies selection process executed

From the 351 works that were fully inspected, the most important sources were classified according to their main characteristics w.r.t this SLR. Table 4.2 lists this classification with 5 categories: (1) the main surveys and reviews found; (2) the main EWS reference architectures; (3) the recent initiatives that apply IoT technologies to EWSs; (4) the references that focus on lexicon approaches for interoperability; and (5) the references that focus on semantic models.

Table 4.2. The most relevant sources according to the main aspects of this SLR

Category	Most relevant sources
Surveys and literature reviews	[7, 15, 38, 100, 106, 240]
Architectural aspects of EWS	[14, 16, 17, 61]
“IoT EWS” concept	[12, 14, 19-21]
Lexicon models	[16, 38, 40, 42, 43, 46, 241]
Semantic models	[14, 20, 21, 49, 50, 242]

An effective EWS must be people-centered and integrate knowledge about the risks, risks’ monitoring and warning, dissemination of meaningful warnings and public awareness [7, 104]. A historic perspective of the technical evolution of EWSs in terms of ICT is presented in [16], which gives emphasis to EWSs for tsunami detection but can be generalized to other types of EWSs. EWSs that monitor the sea behavior exist for a long time, passing through the data processing era (1970s), when digital recording methods started to be used; then, through the microcomputer era (1980s) and the Internet era (1990s), when sensor networks

have been extended and integrated in virtual networks; and through the ubiquitous computing era (2000s), when multi-sensor platforms started monitoring new environmental features. The 2004 Indian Ocean tsunami disaster (more than 200.000 casualties) and the Hurricane Katrina in 2005 (almost 2000 fatalities) lead to a series of activities to improve, create and integrate EWSs.

4.2.1 Related Projects Overview

The German-Indonesian Tsunami Early Warning System (GITEWS)⁹⁴ and the Distant Early Warning System (DEWS)⁹⁵ projects ran in parallel under the EC FP6 scheme (2005-2010), leading to the Collaborative, Complex and Critical Decision-Support in Evolving Crises (TRIDEC)⁹⁶ project, under FP7 (2010-2013). These projects developed an overall generic EWS architecture based on international standardization activities, the *interoperable standards-based EWS*, presented in Chapter 2. Furthermore, TRIDEC evolved the architecture to the *Semantic IoT EWS* architecture [14], a relevant approach for interoperable EWSs leveraged by semantic and IoT technologies. These two architectures are described in the following section.

Besides these projects that focused on EWS to forecast tsunamis, a number of initiatives targeted interoperability in DM. The Data Interoperability Solution At Stakeholders Emergency Reaction (DISASTER)⁹⁷ project, under FP7 (2012-2015) developed the EMERGEL (Emergency Elements) ontology [242], reported under the deliverable D3.22 (EMS Core Ontology), which is further analyzed in this Chapter. The solution proposed of DISASTER project targets the interoperability among different EMSs during crisis scenarios, being oriented to semantic technologies and SOAP standards, such as WS-* (Security, Police, BPEL, etc.).

The “Reliable and smart crowdsourcing solution for emergency and crisis management” project (RESCUER), under FP7 (2013-2016) developed a semantic model (RDF version) of EDXL standards for alerts (EDXL-CAP)⁹⁸ and for message enveloping (EDXL-DE)⁹⁹. EMERGEL ontology was considered by RESCUER to cover semantic mapping of incident information in different languages to integrate legacy systems (existing EMSs) in cross-border crisis situations. However, to cover the brokering of EWSs and communication with real-time monitoring devices, for example to issue alerts, RESCUER adopted EDXL standards. Similarly, the Development of Coordination Mechanisms During Different Kinds of Emergencies

⁹⁴ https://cordis.europa.eu/news/rcn/32224_en.html

⁹⁵ https://cordis.europa.eu/project/rcn/80517_en.html

⁹⁶ https://cordis.europa.eu/project/rcn/95578_en.html

⁹⁷ https://cordis.europa.eu/project/rcn/102279_en.html

⁹⁸ http://fpc.ufba.br/ontologies/edxl_cap

⁹⁹ http://fpc.ufba.br/ontologies/edxl_de

(COncORDE) project¹⁰⁰ also created a semantic model aligned to the EDXL standards to make IMS interoperable.

The Mediation Information System research [113] focused on the dialogue aspect of crisis management interoperability in collaborative networks. This research leverages interoperability by using distributed services exposed through an ESB, introducing a methodology to tackle process interoperability through a matchmaking mechanism that handles service selection and composition, and a message transformation engine to allow interoperability among services. This approach introduced a collaborative situation ontology that supports the process “cartography”, i.e., specify processes with BPMN and transform them in code (workflows) with model transformations [243]. Part of this research was developed within the “DRiving InnoVation in crisis management for European Resilience” (DRIVER+)¹⁰¹ project (2014-2020). DRIVER+ is a large scale (ongoing) project with more than 30 partners, aiming at: (a) creating a pan-European testbed for capability development of DM; (b) developing a portfolio of solutions for DM; and (3) facilitating the shared understanding of DM across Europe.

Other projects that are worth mentioning here include the “Interoperability of Data and procedures In large-scale multinational disaster Response Actions” (IDIRA)¹⁰², under FP7 (2011-2015), which produced definitions for interoperable information interchange for both written and spoken communication. The “Establish Pan-European Information Space to Enhance seCurity of Citziens” (EPISECC)¹⁰³ project, funded by EU (2014-2017) also addressed interoperability, focusing on adopting EDXL and OGC SWE standards as a lexicon approach, which emphasized the concept of “common information space”, a similar scope of the EMERGEL ontology.

The Securing Health Emergency Learning Planning (S-HELP) , under H2020 (2014-2017), developed a decision support system for multi-agency decision-making during cross-border emergencies, aiming to enhance the protection of public health and common grounds for cross-border organizational and legal interoperability. A relevant outcome of this project is an interoperability standard to enable communication and coordination across different geographical areas and cultural settings.

Semantic models were also produced by the “Open architecture for Accessible Services Integration and Standardisation” (OASIS, FP7, 2008-2012), the “Semantic Interoperability for Health Network” (SemanticHealthNet, FP7, 2011-2015), the “Adoption and take up of standards and profiles for eHealth Interoperability”

¹⁰⁰ https://cordis.europa.eu/project/rcn/185499_en.html

¹⁰¹ https://cordis.europa.eu/project/rcn/188608_en.html

¹⁰² https://cordis.europa.eu/project/rcn/98968_en.html

¹⁰³ https://cordis.europa.eu/project/rcn/188573_en.html

(ANTILOPE, H2020, 2013-2015), the “Effective Communication in Out-break Management: development of an evidence-based tool for Europe” (ECOM, FP7, 2012-2016), the “Developing the framework for an epidemic forecast infrastructure” (EPIWORK, FP7, 2009-2013), the (7) “Assessing SNOMED-CT for Large Scale eHealth Deployments in the EU (ASSESS-CT, H2020, 2015-2016), and the “Pan-European infrastructure for ocean and marine data management” (SeaDataNet, FP7, 2011-2015).

As usual, the semantic models produced by these projects were oriented to the use case requirements exploited by the projects for validation purposes (as case studies). Most of them could fulfill the requirements by adding predicates specific to the use cases. However, none of these semantic models is fully maintained and supported by standardization institutes, and most of them cannot be found in the Internet.

Lastly, this SLR identified an increasingly trend to use social media for brokering early warnings. This trend started to be popular from 2010 and numerous efforts have been made to support diverse types of emergencies, especially for epidemiological intelligence [48, 51, 244]. Diverse projects use platforms such as *Twitter* and *Facebook* to publish messages to achieve larger audience, such as the DRIVER+ project.

4.2.2 Existing Reviews and Surveys

The most popular survey of EWSs, reported by the UNISDR [7], lists more than 50 EWSs at that time (2006). The report of UNEP updates this list and classifies EWSs according to rapid/sudden-onset and slow-onset environmental threats [15]. A WHO survey investigated the development of MHEWS to improve monitoring risks for public health, mixing indicator-based and event-based surveillance techniques [240]. The data sources characterize if the surveillance system is event-based (reports, rumors and unstructured data) or indicator-based (traditional practices with official and structured reports). In both cases these systems implement a set of rules over the data to detect initial disasters, having epidemiological studies as the foundation to identify health outcomes such as infectious agents, symptoms, risk factors and transmission behavior. This survey highlights the importance of interoperability to achieve MHEWS and the challenge on integrating data from different domains.

The review on chemical EWS for public health also highlights the interoperability problem between systems that need to share common understanding of the data when exchanging alerts and reports during disasters [100]. Additionally, it enforces the relevance of intelligent automated EWS that are able to interoperate with other EWS, and timely legislation to facilitate cross-border communication. Interoperability is also highlighted by the review on

earthquake EWSs in Europe [61], which concludes that EWSs should be able to integrate sensor data from multiple data sources, as health monitoring systems, considering time- and safety-critical processing constraints. It highlights that an EWS should be able to incorporate multiple algorithms for fast and accurate situation detection based on multiple sensor types.

The UNISDR survey (2006) [7] mentioned above is the most cited review on EWSs. A relevant contribution of this survey is the list of international organizations involved in EWSs and their capabilities for each disaster type. It concludes with some recommendations, such as strengthen the scientific and data foundations for early warning (recommendation 4) and build national people-centered EWS (recommendation 2), which explicitly requires an approach for the development of MHEWS through integrated capabilities across all hazards. This UNISDR survey was updated in 2012 by the UNEP [106], which provided an extensive review of the EWS state-of-the-art, exploiting the role of earth observation to deal with environmental threats. One of the main issues discussed in this review is how to adopt and evolve interoperability standards, such as EDXL-CAP, for the integration of EWSs, improving decision making, coordination and communication links [106].

4.2.3 Common EWS Requirements

Chapter 2 described the four high-level requirements that an EWS must address: (1) to systematically collect data for situation identification (risk knowledge); (2) to monitor hazards and provide warning services; (3) to communicate and disseminate risk information; (4) to use emergency plans as response capability.

In addition to these requirements, other common requirements were identified as a result of the analysis of the existing reviews and surveys. Operating cost requirements describe that (5) an EWS must be affordable for acquisition, maintenance and upgrades, as well as (6) must allow remote management of resources; and (7) higher development automation for dynamic changes at runtime [33]. Furthermore, some general EWS requirements abstracted from the needs of epidemiological EWSs [245] include (8) to process measures of frequency and association, which play an important role to describe causality, e.g., how likely a person can be infected in accordance to a specific population, being able to identify cause-effect factors. The measures of frequency are descriptive multivariate functions (risk, odds, rate and prevalence), which consider variables such as the number of new and total cases, the number of individuals at risk and all individuals in a population, the number of individuals with and without the health outcome and the total person-time at risk. (9) Measures of association aim at comparing the association between a specific exposure and a health outcome. For example, the study about *Zika* outbreak risks in Europe [246] uses information about *Aedes*

mosquito populations, climate (temperature and precipitation), peak flow of air travelers and areas for mosquito-borne transmission of *Zika*.

Once domain experts (e.g., epidemiologists) draw results from these studies, they report their analyses through natural language description of the complex rules over the context elements required to detect the outbreak situations. These rules must be implemented by software developers responsible for the development and integration of EWS, which implies that the rules should be understandable for them. The main functional requirement is to improve the common understanding of such rules and the involved context, i.e., to achieve successful communication among domain experts and software modelers by decreasing the semantic gap among them.

An EWS must provide temporal reasoning over the events being monitored [247], which relies on temporal relations to correlate events. For example, the situation of a possible contagion of *Zika* is characterized by the situation of high risk of *Zika* infection overlapping the situation of a person diagnosed with *Zika* within the same geographical area.

In addition, (10) an EWS must implement temporal existential rules, i.e., rules that only match events occurring in a specific time window (ranges of time units). For example, the case of influenza-like illness (ILI) is defined by WHO as “an acute respiratory infection with: measured fever $\geq 38\text{ }^{\circ}\text{C}$ and cough; with onset within the last 10 days”. Therefore, ILI is characterized by the existence of fever and coughing within the past 10 days. Sliding time windows allows the definition of existential rules for EWS [248]. An EWS also needs (11) to allow the definition of descriptive multivariate and aggregation functions, for example to describe average emergency levels and sums in populations [249].

An EWS requires (12) a dynamic and adaptive approach to be able to change the applications at runtime [250]. Moreover, because DRR and healthcare are related domains, an EWS needs (13) to have high interoperability to receive upstream data from *e-Health* sensor platforms in accordance to electronic health records standards, e.g., HL7; and to send downstream information for medical staff, e.g., EDXL [16]. Finally, the specification of the EWS components must be supported by a modelling tool widely adopted in the healthcare community [251].

4.3 Interoperable Standard-Based EWS

As introduced in Chapter 2, an EWS implements a chain of subsystems that address functional EWS components: sensor, detection, decision, broker and response [17]. Similarly, a conceptual high level architecture of an interoperable EWS was developed [16], illustrated in Figure 1.1. This interoperable conceptual architecture, called *standard-based EWS*, represents the workflow of three

consecutive steps: *Upstream Data Acquisition* (sensor), *Decision Support* (detection and decision) and *Downstream Information Dissemination* (broker and response).

In this architecture, data are acquired from sensor systems (sense, *Upstream Data Acquisition*), and are represented according to interoperability standards, e.g., OGC SWE. Distributed sensor systems transform observations into digital signals, pre-process the associated data values to ensure that they represent relevant information for decision making and transmit these data values to a middleware. The *Upstream Data Acquisition* component is responsible for acquiring the data from the middleware, and for pre-processing and storing the streaming data in *Context Database(s)*.

The data stored are subjected to rules to detect situations of interest with support of *Models*, which represents the emergency situation identification mechanism and can be either specification-based or learning-based. Therefore, models can be represented as a set of patterns and can be deterministic (rule-based) or non-deterministic (machine learning).

The *Decision Support* component applies the intrinsic rules of the models over the context data. Once a situation is detected, the EWS considers the requirements of the alert targets to assess the risk and determine the emergency response. The *Decision Support* component supports this capability by allowing the EWS administrator to configure actions according to the situation identified, usually with support of a workflow management system. This allows the EWS to send notifications according to the pre-configured actions to different target groups through the *Downstream Information Dissemination* component, which is responsible for message brokering. For data representation, the architecture guides the adoption of standards for messaging formats, such as the OASIS EDXL standards. The EDXL-CAP is the common standard adopted for alert messages as downstream data representation, used by several EWSs, such as GITEWS and IPAWS.

The target groups can be either humans, who receive alerts through mobile applications or low-frequency radio, or actuators for automatic emergency response. The *standard-based EWS* architecture allows the integration of different EWSs through event- and message-oriented middleware, as context broker and service broker. Therefore, this architecture prescribes the adoption of components that are also core elements of IoT platforms. The term “IoT EWS” [12, 14, 19-21] denotes EWSs supported by IoT technologies. IoT technologies improve EWSs with new and more accurate features.

However, the large number of devices also creates challenges related to handling the volume, velocity and variety of generated data (three V’s of big data). Interoperability is crucial for the effectiveness of IoT EWSs and is related to data variety. The main limitation of the *standard-based EWS* approach is that it neglects

semantic interoperability, albeit the lexicon standards (e.g. EDXL) partially cover the semantics of the messages.

The importance of achieving common understanding, i.e., semantic interoperability, in EWS context is discussed in [114], which highlights the importance of capturing different conceptualizations, from multiple stakeholders involved in the disaster situations, during the design of EWSs. It also presents a summarized literature review about EWSs and describes three use cases in Kenya, Sri Lanka and Hawaii. In Hawaii, the integrated EWS is highlighted, being able to process different hazardous events, as tsunamis, hurricanes, floods and storms.

In particular, semantic interoperability is a crucial aspect for effective MHEWSs that can seamlessly integrate to other EWSs. For example, in March 2018 the *Abu Dhabi Police* activated a national traffic IoT EWS, which detects dangerous driving conditions based on environmental sensors, e.g., fog and air pollution sensors, and issues warning to drivers near the affected location via SMS and a mobile application. Suppose that a driver in *Abu Dhabi* is under monitoring of the *Philips IntelliVue Guardian* EWS¹⁰⁴, which is a hospital-oriented IoT solution composed of medical wearables and a cloud infrastructure to identify and alert health emergencies based on early warning scores.

In a situation where this system detects an abnormal behavior of the driver's heart, only the hospital information system running the *Phillips* response system, which relies on the *Philips* proprietary data model, will be able to understand the data. The national traffic EWS will not be able to reason upon these data in its detection system due to the lack of a shared semantics regarding the data representations used. The integration of these systems rely on manual activities necessary to relate both data models (from the national traffic IoT EWS and *Philips* EWS) and then, the proper implementation of message exchange between them.

Classical EWSs tend to use semi-formal or informal models, while only a few use semantic models ("semantic EWS"). Most of the semantic EWS approaches [14, 20, 21, 49, 50] extend the *standard-based* architecture by using semantic technologies to represent data, both upstream and downstream, taking advantage of inference capabilities. Usually, the concept of a common information space is exploited, linking semantic technologies to SOA [21]. Performance tends to be a drawback, which makes scalable time-sensitive data exchange and processing from heterogeneous data sources challenging.

The *Semantic IoT EWS* approach [14], illustrated in Figure 4.2, improves semantic interoperability with a scalable solution for data exchange and processing, by providing a balanced way to use lightweight and heavyweight ontologies. Lightweight ontologies are used in *Upstream Data Acquisition* and *Downstream*

¹⁰⁴ <https://www.usa.philips.com/healthcare/clinical-solutions/early-warning-scoring/intellivue-guardian-ews>

Information Dissemination to reduce the message payload during data exchange and processing.

Figure 4.2. The semantic IoT EWS architecture [14]

Heavyweight (formal logic based) ontologies are used in *Decision Support* to add semantics to the data, enriching the fusion of new and historical data by enabling the discovery (inference) of new knowledge. This new information is serialized as lightweight (association based) semantics to be included in the warning messages that are brokered through the downstream information dissemination component. This balance of heavyweight and lightweight ontologies enables scalable time-sensitive data handling from heterogeneous sources with effective responses.

Upstream sensor data are annotated only with SSN (1.0) predicates, which are applied at the broker (gateway) level, since sensors only need to interoperate with a control centre via a sensor's access node. This type of *Semantic Gateway* is a relatively new concept in the IoT domain [54, 55] and leverages on the evolution of low-cost gateway devices, including smartphones.

The *Semantic IoT EWS* approach introduced a heavyweight ontology, the *Decision Support Ontology* (DSO), by reusing terms from the SSN (1.0) and incorporating terms from the OGC O&M standard. Although DSO's goal is "to aggregate and align multiple ontologies to support compound EWS semantics and ontology commitments", it lacks the support for multiple domain ontology alignments at runtime, i.e., it does not provide a mechanism for describing and executing ontology alignments at runtime. Besides this, efficiently publishing large volumes

of semantically rich sensor data is a major architectural challenge [10], mainly due to the inefficient data transfer (throughput) of the current semantic technologies.

4.4 Emergency Data Representation

Measuring the interoperability of a system depends on the standardization of interfaces, data exchange formats and protocols used by the system [16]. Therefore, a fundamental aspect of interoperability is the mechanism to represent data. Numerous data representations have been proposed for DM, including lexicon standards and semantic models. For example, [252] uses an extensive list of information sources for disaster management metamodeling, including the term *EarlyWarningSystem*. Although these initiatives targeted the interoperability improvement in some way, this multiple creation of models is exactly against interoperability, since each new model usually overlaps (in some extent) to other models, requiring their alignments. Therefore, in this thesis the priority is given to standardization initiatives. The OASIS EDXL and the OGC SWE are the most popular lexicon and ready-to-use standard initiatives for interoperability of emergency services in the context of EWSs.

4.4.1 Lexicon Standards for Disaster Management

The OASIS EDXL standards are the most common used for downstream data representation of interoperable EWSs [16, 38, 40, 42, 43, 46]. In DEWS [253], the role of EDXL-CAP in logistics accidents is exploited. EDXL-CAP is one of the core elements of IPAWS and is a key standard for the representation of alerts for multi-hazards [254]. Other systems that are able to issue alerts and warning also adopts EDXL-CAP, such as the *Google Public Alerts*¹⁰⁵, which also provides an EDXL-CAP validator¹⁰⁶. The EDXL-CAP data model is illustrated in Figure 4.3, having *Alert* as a central element, which represents a message that is composed by *Info* element(s). Each *Info* element represents an emergency event that has a severity and an urgency (enumerations), can be composed by resources, which represent attached files (e.g. documents, pictures), and can occur in one or more locations (*Area*). Besides EDXL-CAP, EDXL allows the representation of emergency situations (EDXL-SitRep) and tracking emergency patients (EDXL-TEP). EDXL standards support requirements engineering for emergency interoperability, as in the profile-based approach [241]. Although EDXL is not implemented as a semantic model, it allows the inclusion of controlled vocabularies serialized as RDF/XML through the *ValueListURIType* element.

¹⁰⁵ <https://developers.google.com/public-alerts/reference/google-cap-requirements>

¹⁰⁶ <https://cap-validator.appspot.com/>

Figure 4.3. EDXL-CAP main elements¹⁰⁵

OGC SWE standards are often used for sensor data representation, and have been applied in diverse types of EWSs, e.g., for floods, earthquakes and tsunamis [13]. For example, the *Emergency Management Victoria* (Australia) uses these standards for multi-agency sensor data integration [36]. OGC SWE consists of four information models and four service models. Information models focus on the representation of sensor data with a sensor modeling language (*SensorML*) and *Observations & Measurements* (O&M). The Services model includes observation and notification. The OGC SWE standards were used as reference models for the development of the SSN (1.0) ontology. Therefore, OGC SWE is commonly used in EWSs for messaging format in data acquisition, although it also provides a conceptualization for notifications.

The National Information Exchange Model (NIEM)¹⁰⁷ is an US initiative that covers several domains directly related to DM, such as infrastructure protection,

¹⁰⁷ <https://www.niem.gov/>

justice, maritime and human services. NIEM is commonly used for data exchange of EMS and IMS, being mostly complementary to the OASIS EDXL standards. Some research already exploited the joint adoption of EDXL and NIEM for Unified Incident Command and Decision Support (UICDS) [255].

This SLR also reinforced the need to match equivalent terminologies in order to provide a higher level of interoperability for EWS with multiple data sources. For example, the EDXL-TEP provides mappings to HL7 (v2.7) standards¹⁰⁸ regarding the overlapping terms about patients. Similar to the model transformations and semantic translations development methodologies, these bi-directional transformations were developed according to the following method:

1. The designer identifies which elements in the target data structure are required, usually constrained by the model, e.g., a *Message Status* is always required to be instantiated in an *Alert* element in EDXL-CAP.

- a. For each required element, matching structure(s) are located in the source data structure.

- b. If a matching structure is found, which can be a direct (1:1) relation or mapping patterns, then the designer should describe a transformation rule.

2. The designer identifies elements in the target data structure that are optional, following the same instructions of step 1 for each optional element. These two steps cover the analysis of the whole target model.

3. The most relevant elements in the source model should be identified by the designer to select which elements should be reflected in the target, which usually depends on interoperability requirements. For example, in EDXL-CAP, if the use cases only need the location of the events and do not need to consider the files attached, then the designer would select the *Alert*, *Info* and *Area* elements, excluding the *Resource* element.

- a. For each element, the corresponding structure in the target data structure is searched.

- b. If corresponding structure is found, then the designer should describe a transformation rule. Otherwise, if the target does not provide a similar element, then the designer should consider extending the target model.

Since these lexicon data models aim at both standardize the syntax of messages and to represent common understanding, the natural evolution of these data representation schemas is the adoption of semantic technologies. Numerous semantic models were developed in this way [217, 223, 224, 256, 257].

¹⁰⁸<http://docs.oasis-open.org/emergency/TEP-HL7v2-transforms/v1.0/TEP-HL7v2-transforms-v1.0.html>

4.4.2 Semantic Models

Several semantic models for DM used the *SA theory* as background [9, 92, 166, 256], probably because it is often necessary to represent dynamic temporal aspects of emergencies. For example, the *Event-Model-F* [57] was developed to address the requirements of sharing event interpretations in emergency response, exploiting the role of causality of events and situations. Figure 4.4 illustrates the foundational aspects of causality of *Event-Model-F*, which incorporated DUL as foundational ontology. This conceptualization is similar to UFO-B, discussed in Chapter 2, and extends the causality relationship (object property *ufo:triggers*) with the *EventCausalitySituation* element, which is satisfied by the description of *Causes*, *Effects* and *Justification*.

Figure 4.4.
Causality pattern
between events and
situations [57]

A current ongoing initiative that uses the EDXL standards as baseline for DM ontologies led by the *US Department of Defense Public Safety and Emergency Management Communications* [257]. This initiative addresses a number of interoperability requirements through the development of the *Public Safety and Emergency Management Communications Ontology*, which was grounded in the BFO foundational ontology. This work also emphasized the important role of the EDXL-CAP standard for interoperability of emergency services, representing an *edxl:Message* as a *bfo:InformationBearingEntity* and extending the typification of alert messages, as illustrated in Figure 4.5(a). The ontology also considered NIEM and other data models, covering a comprehensive terminology, including types of incidents (used similarly to emergency events) and the response activities, such as evacuate, shelter and monitor (Figure 4.5(b)). This ontology has been carefully developed, having its quality leveraged by the application of ontological analysis and best practices from the BFO formal ontology, addressing different points of view of the main stakeholders, being adherent to relevant domain standards (EDXL, NIEM) and being oriented to practical use cases. However, the ontology is not available and the report [257] only describes some parts of it, which impacts its reusability.

The EMERGEL ontology is frequently cited in the literature [242, 256, 258]. The EMERGEL scope includes the domain of information artifacts used during disaster situations, considering languages among countries in Europe. A modularized approach was followed, having a core ontology defining the main concepts by extending DUL, similar approach of SSN/SOSA and SAREF. This “upper-level” part of EMERGEL represents a disaster as an event (*emergel:Disaster a dul:Event*), providing a taxonomy of disaster types, considering a disaster synonymous to an emergency. However, it overloads the definition of a disaster as an event, situation and incident, not making an explicit difference among these concepts, a similar issue of the EDXL standards approach.

Figure 4.5. (a) CAP Message Class Hierarchy; (b) Incident response activities [257]

EMERGEL introduces an object property for causality among disasters (*emergel:causes*) to allow traceability of a chain of events. Furthermore, EMERGEL exploits the concepts of agent, role and profile to represent the actors playing role(s) in a disaster situation. In addition, the ontology adopts the perdurantism approach, similar to DOLCE and UFO, describing an event according to time slices (boundaries), considering *Allen's* temporal relations (interval algebra) to relate events, and the participation role of physical objects in events. EMERGEL focuses on representing geographical aspects and spatial geometries. The common approach for RDF extension is adopted by EMERGEL, i.e., create new elements by specializing the existing classes and properties of the core module. All terminology used considered the translation (direct or not) to several European languages (e.g., *German, Dutch, Danish*) [242]. Experiments [242] showed how to extend EMERGEL for the representation of documents (e.g., emergency plans) and data repositories, proposing as future work the alignment to the W3C DCAT standard. Although the EMERGEL website is still on-line¹⁰⁹, the ontology

¹⁰⁹ <http://vocab.ctic.es/emergel/>

implementation (OWL or RDF) is not available, for example the URI (*curl.org/emergel*) is not accessible and, therefore, affects its reusability.

Figure 4.6.
CONcORDE
ontology of
incident
participants [242]

Different from the previous ontologies, the ontology produced by the CONcORDE project is available to be reused (OWL serialized as TTL)¹¹⁰. This ontology is also aligned to EDXL and aims at supporting interoperability of EMS and IMS, emphasizing the characteristics of incident participants for emergency response. Figure 4.8 illustrates the central *Participant* element, which has associated response activities that require certain profiles to be executed. It also highlights the need of interoperating IMS (EMS) with healthcare systems, such as health facilities like hospitals and clinics. Adopting the capability of interoperating emergency systems with e-health solutions has been growing, since a core aspect of emergencies is the threat to people's health. For example, research has been performed on processing early warning scores through vital signs monitoring [259]. According to the *General Electrics*, the interoperability of systems in healthcare ecosystems could help saving US\$ 30 billion per year [260].

Although the CONcORDE ontology follows the best practices on ontology implementation and reuses some top-level ontologies, such as SKOS, TIME and

¹¹⁰ <https://github.com/OntoRep/CONcORDE>

WGS 84, it is not grounded in a foundational ontology and it does not address the conceptualizations of warnings and alerts, lacking on alignment with EDXL-CAP.

As mentioned before, several ontologies covering the notion of situation awareness were produced [261]. Most of them are not available to be reused. A literature review about ontologies for DM [262] found that 13.3% of the resulted ontologies focused on the representation of SA, only behind of critical infrastructures (20%). The *Event-Model-F*, previously described, is one of the most relevant and high-quality SA core ontology applied in emergency scenarios, which is a high quality reference ontology for situation-related constructs.

4.5 Challenges and Research Questions Revisited

The main lessons learned from this SLR have been applied to refine/detail the research questions. The following challenges to achieve semantic interoperable EWS are described as high-level questions. To tackle each challenge, a set of corresponding requirements are listed, which have been addressed by the approach proposed in this thesis.

4.5.1 C01: Semantic Integration of a variety of data sources

How to make systems that play the role of data sources to understand each other, i.e., how to avoid semantic errors and distortions, when multiple ontologies, semantic models, standards and data models from different and overlapping domains are involved, considering their syntactic and semantic alignments?

(Req1) Enable the development of an EWS core context model as a well-founded ontology, taking into account existing data representations and possible restrictions from the EWS requirements.

Description: The framework must enable the development of the EWS core context model as a well-founded *Tbox* ontology, taking into account existing data representations and possible restrictions from the EWS requirements. For example, if the approach is used to develop an e-health EWS, the framework should guide the development and/or reuse of a well-founded e-health ontology. In addition, if there is a restriction like making the EWS compliant to HL7 FHIR lexicon standard, then the framework must support the mappings from this standard to the EWS context model.

(Req2) Enable the development of an EWS data acquisition component to be able to process data from different sensor types described with different semantic models, i.e., provide syntactic and semantic interoperability.

Description: The framework must enable the development of the EWS data acquisition component so the EWS is able to process data from different sensor types described with different semantic models, i.e., provide syntactic and semantic interoperability with sensor systems. Since ontology alignments, such as semantic translations, are recommended to address this requirement, a semantic translation mechanism for upstream data and a mechanism to reflect the resulted data harmonization into the EWS decision support component are required.

(Req3) Enable a data fusion mechanism for EWS, allowing it to combine incoming real-time data with historical data, both for situation identification and for downstream information dissemination.

Description: The framework must enable a data fusion mechanism for the EWS so it is able to combine incoming real-time data with historical data, both for situation identification and for upstream information dissemination. For example, if the approach is used to develop an epidemiological surveillance EWS that monitors mosquito traps with sensors, the framework should guide on how to link the mosquito counters (from the sensors) with prior epidemiological studies of the monitored area.

4.5.2 C02: Processing in time- and safety-critical applications

How to achieve appropriate performance and scalability for real-time upstream data acquisition, emergency risk detection and message brokering, in terms of total transaction time and quality of service?

(Req4) Enable the EWS to provide adequate performance, in terms of total processing time, which depends on the specific safety-critical restrictions regarding minimal response time. If the EWS requirements explicitly restrict the minimum time thresholds, the EWS should be able to process the data within this threshold.

Description: The framework must consider safety-critical or life-critical system constraints, where failure may result in death or serious injury to people loss or damage to property or environmental loss, thus, it depends on the type of disaster(s) that the EWS is designed for. For example, for an effective tsunami EWS, the time constraint for the system, which should be less than the travel time of a tsunami to reach the shore (10-30 minutes), depends on four characteristics: detection time (T1), assessment time (T2), evacuation time (T3) and tsunami travel time (T4). T4 is exogenous, thus, to make the EWS effective the time needed for warning and evacuation (T1, T2, T3) should be minimized [263]. Therefore, the approach must enable the EWS to provide adequate performance, in terms of total reaction time depending on the specific EWS restrictions. If the EWS requirements explicitly restrict the minimum time thresholds, the EWS

should be able to process within this threshold. For example, it is common that a traffic EWS needs to issue warnings within few seconds after a collision is detected, e.g., the real-time traffic alert system [264] sets a threshold of 3.6 seconds, so the emergency response procedure can start immediately. This threshold also depends on how the reaction time is measured.

(Req5) Enable the EWS to be scalable according to the data input volume and speed, i.e., be able to allocate resources according to the number of data sources, their input frequency and message size.

Description: The approach must be able to scale the EWS infrastructure depending on the data input rate and size, being able to automatically allocate resources. For example, suppose an EWS infrastructure that supports five medical devices concurrently where each one sends messages of 50Kb per second, thus being able to process $\sim 15\text{Mb/minute}$. If ten new devices are added to the solution, then the EWS should be able to support $\sim 45\text{Mb/minute}$, which impacts the entire cloud infrastructure operation throttles (rate limitations), such as the context broker rate of device-to-cloud requests, the application services memory and CPU, and database storage servers.

(Req6) Enable a dynamic and adaptive mechanism to modify the EWS at runtime, minimizing the effect on the running instances.

Description: The framework must be able to provide an interface for the EWS administrator to modify the rules underlying the situation identification mechanism of the EWS at runtime, minimizing the impact on the running services. For example, if a detection rule requires to be changed, such as threshold value, the EWS should provide a way to modify the rule while it is executing. Similarly, if a new target requires to receive the emergency notifications, the EWS must also provide an interface for this change, avoiding the impact on the availability of the EWS services. This requirement is derived from RQ02.3.

4.5.3 C03: Data analysis for effective responses

How to provide high quality situation awareness, i.e., perception, comprehension and projection, and decision making to improve emergency response decision support?

(Req7) Offer modelling capabilities for the representation of temporal relations among different types of events and situations. In particular, it must enable the EWS to use interval relationships, such as Allen's operators, to relate events for situation identification.

Description: For example, if the approach is used to develop an epidemiological EWS for tuberculosis, the designer should be able to model that a contagion risk situation is composed by the overlapping temporal relation between the situation of person *A* being treated in a hospital and person *A* meeting an infected person *B* in the very same hospital, followed by the infection of person *A* and the increase of number of cases above a certain threshold.

(Req8) Enable the representation of complex rules over context data, such as multivariate functions (risk, odds, rate and prevalence) and temporal existential rules (sliding time windows).

Description: For example, it is common that detection rules of a tsunami EWS refer to the height and period of consecutive waves, which should occur in a time window of some seconds, and calculates the risk that the next wave is a tsunami.

(Req9) Enable the EWS to issue warnings, as semantic enriched messages, to multiple targets through multiple channels (e.g., broker, e-mail, SMS). In particular, enable the modification of the information requirements of the targets at runtime through an UI.

Description: The EWS administrator must be able to configure the specific information requirements of each target and change the channel (e.g., broker, e-mail, SMS) at runtime through an UI when necessary. The target information requirements should be reflected in the downstream information dissemination component.

4.6 Concluding Remarks

The complexity and dynamic nature of emergency situations present a series of challenges, involving parties' interaction, including people (first responder groups and victims), environments, risks, hazards and multiple events. Those relations rely on complex rules, which are difficult to characterize during the specification of SA applications. Moreover, operational decisions must efficiently manage human and computational resources in short time, making the design of real-time decision making a challenge. The information collaboration among EWSs has great importance and needs to be addressed. This chapter presented the main outcomes of our systematic literature review about interoperable EWS architectures and their data representations. We highlighted the importance of the semantic approaches for the development of interoperable EWS, emphasizing the "semantic IoT EWS".

This research aims at improving the semantic interoperability of IoT EWS for smart emergency services, i.e., improve the semantic integration capability of

components of an IoT EWS and enable seamless integration with other EWS. To achieve this goal, we concluded that the following three challenges must be addressed: **(C1)** semantic integration of a variety of data sources, avoiding loss of semantics when multiple data representations from multiple domains are involved, considering their syntactic and semantic alignments; **(C2)** processing in time- and safety-critical applications to provide the required performance for data acquisition, risk detection and message brokering, in terms of scalability and total transaction time; **(C3)** data analysis for effective responses to enable high quality situation awareness (perception, comprehension and projection) to improve emergency decision support.

A limitations of this SLR is that all sources that could not be fully accessed using the *University of Twente* credentials were eliminated, as well as paper conferences published before 2005 and workshop papers before 2010.

The SEMIoTICS Framework

In the previous chapter, the main research challenges for the development of semantic interoperable IoT EWS were scrutinized and mapped to requirements that are applicable to any type of disaster. This chapter gives an overview of the main components and methodology of our treatment design to address these requirements. Section 5.1 introduces the treatment design, called the “SEmantic Model-driven development for IoT Interoperability of emergenCy serviceS” (SEMIoTICS) framework, aiming at addressing the general requirements. The SEMIoTICS framework consists of a reference architecture and a design methodology. The SEMIoTICS reference architecture is proposed, outlining the components’ technologies and how they interact with each other. Section 5.2 describes the SEMIoTICS semantic model-driven methodology, which is exemplified through a scenario of developing an IoT EWS for epidemiological surveillance of tuberculosis. Section 5.3 describes the role of the FAIR data principles in the methodology, showing how an EWS developed with the SEMIoTICS framework can produce FAIR data. It emphasizes the importance of leveraging the semantic model-driven methodology with standard-oriented approach. Finally, Section 5.4 presents the concluding remarks about issues and required improvements of SEMIoTICS’ components to address the research challenges.

5.1 Reference Architecture

The general requirements were derived from the challenges, as described in Section 4.6. Table 5.1 summarizes the derived requirements. The treatment design to address them is a “meta-system” termed as “SEmantic Model-driven development for IoT Interoperability of emergenCy serviceS” (SEMIoTICS), which is a framework to build semantic interoperable IoT EWS. SEMIoTICS consists of an architecture, which is illustrated in Figure 5.1, as well as technologies

and guidelines that are based on the MDE methodology for software engineering [133], as described in Chapter 2. The architecture has components to address each activity of the EWS workflow (Figure 2.4) necessary for real-time data processing.

Table 5.1. Requirements of the SEMIoTICS framework for semantic IoT EWS

ID	Textual description
<i>Req1</i>	Enable the development of the EWS core context model as a well-founded ontology, taking into account existing data representations and possible restrictions from the EWS requirements
<i>Req2</i>	Enable the development of the EWS data acquisition component so the EWS is able to process data from different sensor types described with different semantic models, i.e., provide syntactic and semantic interoperability with sensor systems
<i>Req3</i>	Enable a data fusion mechanism for the EWS so it is able to combine incoming real-time data with historical data, both for situation identification and for downstream information dissemination
<i>Req4</i>	Enable the EWS to provide adequate performance, in terms of total processing time, which depends on the specific EWS restrictions. If the EWS specific requirements explicitly restrict the minimum time thresholds, the EWS should be able to process within this threshold.
<i>Req5</i>	Enable the EWS to be scalable according to the data input volume and velocity, i.e., being able to allocate resources according to the number of data sources, their input frequency and message size
<i>Req6</i>	Enable a dynamic and adaptive mechanism to modify the EWS at runtime, minimizing the effect on the running instances
<i>Req7</i>	Offer modelling capabilities for the representation of temporal relations among different types of events and situations. In particular, it must enable the EWS to use interval relationships, such as Allen's operators, to relate events for situation identification.
<i>Req8</i>	Enable the representation of complex rules over context data, such as multivariate functions (risk, odds, rate and prevalence) and temporal existential rules (sliding time windows)
<i>Req9</i>	Enable the EWS to issue warnings, as semantic enriched messages, to multiple targets through multiple channels (e.g., broker, e-mail, SMS). In particular, enable the modification of targets' information requirements at runtime through an UI

5.1.1 The EWS Workflow

The SEMIoTICS architecture is illustrated as a workflow in Figure 5.1, grounded in the *Endsley's Situation Awareness* theory, presented in Chapter 2. A direct mapping to *SA* theory is highlighted through the colors of the SEMIoTICS components. The *perception* level (red) can be mapped to the *Input Handler* and *Context Data Manager*. The perception language and conceptualization can be mapped to the *Context Model* and to the *Abstraction* components, respectively. The *comprehension* level (yellow) can be mapped to the *Situation Identification Manager*, where the comprehension knowledge refers to the *Situation Model*. Finally, the *projection of future status* (green) can be mapped to the *Situation Reaction Manager* and the *Output Handler*, where the person's memory refers to the *Action Model*.

The SEMIoTICS framework is also aligned to the *Standard-based* and the *Semantic IoT EWS* approaches, presented in Chapter 4. In summary, the mappings from the *Standard-Based* architecture to the SEMIoTICS architecture are:

- From *Upstream Data Acquisition* to *Subscriber REST/RDF* and *Input Handler*.
- *Decision Support* component can be mapped to:

- *Context Data Manager*, which is enriched by the *Abstraction*, *Context Model* and *Context Database(s)* elements.
 - *Situation Identification Manager*, which is enriched by *Situation Model* and *Situation Type Rules*.
 - *Situation Reaction Manager*, which is enriched by the *Action Model* and *Emergency plans*.
- From *Downstream Information Dissemination* to *Output Handler*.

The mappings from the *Semantic IoT EWS* architecture to the SEMIoTICS architecture are:

- *MOM* serves both *Input Handler*, subscribing to receive data, and *Output Handler* (publishing emergency notifications).
- *Context-aware intelligent filtering* along with the *Ontology Management* service components can be mapped to the *Context Manager* supported by the *Context Model*.
- *Databases* can be mapped to the *Context Databases*.
- *Rule Management Service* along with the *Decision Tables* components can be mapped to the *Situation Identification Manager* supported by *Situation Type Rules*.
- *Decision Support Workflow Service* can be mapped to the *Situation Reaction Manager*, having emergency plans represented as workflows

SEMIoTICS is leveraged by the adoption of semantic technologies (Chapter 2) and IoT platforms (Chapter 3), using brokers, RESTful microservice architecture with JSON-LD, distributed CEP and workflow systems.

Figure 5.1. The SEMIoTICS framework architecture

The *Subscriber REST/RDF* component along with the *Input Handler* are responsible for the *sense* activity. They implement the *Upstream Data Acquisition* element of the *Standard-Based* architecture (Section 4.3). Data are acquired by the EWS by subscribing to a context broker, which should provide semantically enriched data (in RDF), either directly from the sensors or from a semantic

gateway, which can be deployed in the field (e.g., in a mobile phone) or in the cloud (e.g., in a semantic web application). The *Input Handler* checks whether the EWS is capable of understanding the data, i.e., if the data messages are compliant to the EWS *Context Model*. If the data cannot be understood by the EWS, then the *Input Handler* executes semantic translations to transform the data.

The *Context Manager* sends the (transformed) data messages to the *Context Data Manager*, which is responsible for storing the data in context database(s) and to forward the data to the *Situation Identification Manager*. This latter component uses *Situation Models*, which are transformed in *Situation Types* (rules) and executed by a CEP engine as a situation identification mechanism, covering the *detect* activity of the EWS workflow. A *Situation Model* uses the data elements from the *Context Model* for the situation types (rules) specification.

Once the CEP engine detects a situation, the data related to the situation are sent to the *Situation Reaction Manager*. Each *Situation Model* is linked to the emergency response business processes through the *Action Model*, which specifies the procedures to react to the situation, usually derived from emergency plans. The *Action Model* specifies the targets that need to receive the emergency notifications, considering the information requirements of each target and the channels to send the data to targets. Therefore, the *Situation Reaction Manager* is responsible for the *decide* activity of the EWS workflow.

The *broker* activity is supported by the *Output Handler* and the *Publisher REST/RDF*. They implement the *Downstream Information Dissemination* element of the *Standard-Based* architecture. Data are represented with standard-oriented emergency semantics, namely the EDXL-CAP standard. These messages are made available for the targets by publishing the data in a service broker, as an emergency notification publisher, and through specific channels, such as e-mail and SMS. We justify our choices by detailing these components and explain how they address each requirement, as follows.

Data stored in the *Context Database(s)*, the *Situation Type Rules* and the *Emergency Plans* can be accessed through a RESTful service, having an endpoint for each component. In the case of *Context Database(s)*, the endpoint only provides GET methods, since inserting and modifying data can only be performed by the *Context Data Manager*. On the other hand, both *Situation Type Rules* and the *Emergency Plans* have endpoints that provide PUT and POST methods. The former enables the insertion and change of detection rules, while the later enables the insertion and change of emergency procedures. The emergency procedures are represented as *Action Model* and linked to the *Situation Model*. Each endpoint provides data in an appropriate language, such as ontological language, event processing language and business process language. Therefore, SEMIoTICS enables an EWS to share not only the data, but also all its internal knowledge.

5.1.2 Handling Input and Output Data

The *Input Handler* is responsible for upstream data acquisition, including the harmonization of data that are represented with different syntaxes and semantics, while the *Output Handler* is responsible for brokering the emergency notifications to the proper targets. SEMIoTICS assumes the use of message- and event-oriented brokers through the *publish-subscribe* pattern. By following this approach, the EWS is capable of integrating with diverse data sources, including other EWS.

The *Input Handler* plays the role of subscriber through the *Subscriber REST/RDF* component, responsible for listening data that are published in a broker. The *Output Handler* plays the role of publisher through the *Publisher REST/RDF* component, responsible for sending the emergency notifications to a broker. This approach supports system scalability, since brokers are usually scalable (e.g., Azure IoT Hub), and system performance, since brokers can provide high operation throttles, e.g., Azure IoT Hub can provide 6.000 messages of 4Kb per second per IoT device. Therefore, this approach supports the realization of **Req4** in terms of scalability and **Req5** in terms of brokering performance.

As described in Chapter 3, the RESTful microservice architecture is based on the SOA principles to handle efficiently a large number of tasks or requests at the same time, being prepared to increase (or decrease) this number in the future. Therefore, the adoption of this architecture in the *Input* and *Output Handlers* also supports the realization of **Req5**, enabling the EWS to be scalable. Moreover, serializing data as JSON messages, instead of XML, also brings performance improvement for upstream data acquisition and downstream information dissemination, when compared to XML, because of lower verbosity and the possibility to use fast JSON processors, which supports the realization of **Req4**.

The use of RDF to describe the data from different sensor types enriches the semantic interoperability of the system and serializing the RDF as JSON, through JSON-LD, enables syntactic interoperability. This approach allows data integration during real-time data acquisition and information dissemination, thus, addressing **Req2** (input) and **Req9** (output), adopted by both *Input* and *Output Handlers*' interfaces. As a side effect, since JSON-LD is JSON-compliant and JSON is the most popular message format used in web-based programming environments, adopting JSON-LD facilitates the understanding of RDF by software developers.

The adoption of JSON-LD does not completely fulfil **Req2** because data can still be represented according to different ontologies. To tackle this issue, the architecture supports the identification of functional components that reflect domain-specific aspects of the EWS, recommending interoperability standards to connect these components, and identifying adaptor components to bridge different standards or standards and proprietary solutions. The framework separates adaptors for syntactic interoperability, in case input data are represented with a

lexicon standard, from adaptors for semantic interoperability, in case input data are represented with semantic models. Adaptors are implemented as syntactic and semantic translations. This syntactic and semantic translations approach is able to realize semantic integration (**Req1**).

When the EWS receives a message, the translation mechanism first checks if the message semantics matches to the EWS *Context Model* (core ontology), which should be aligned to SSN/SOSA ontology and EDXL-CAP semantic models. If not, semantic translations (previously configured) are executed to make the data understandable to the EWS. Different semantic translation mechanisms can be used, e.g., “raw” SPARQL queries, where expressiveness and performance aspects should be considered for selecting the most appropriated one. After the execution of the semantic translations, the data are loaded in memory as classes from an object-oriented language (e.g., POJO or POCO), which are used as data contract interfaces to post to the *Context Data Manager*. The “raw” JSON-LD message is also forwarded to the *Context Data Manager* for historical storage purposes.

For example, the EWS can acquire data from a sensor network that provides data represented with SSN/SOSA standard and another sensor network that provides data represented with SAREF. To enable the semantic integration of these different sources, the *Input Handler* simply forwards the messages from the first network, since they are already described as SSN/SOSA, and translates the messages (SAREF) from the second network to SSN/SOSA, harmonizing both input data. Mechanisms for semantic translations which are template-based, such as IPSM from INTER-IoT, can be adopted if its use does not affect the required translation performance.

On the output side, the *Output Handler* enables message brokering to multiple targets, according to the emergency procedures defined in the decision support component, realizing **Req9**. The *Decision Support Manager* provides the information about the message targets, including their information requirements and the channel(s) to receive the data. Therefore, each target has a list of information requirements. Each information requirement is mapped to a data element of the EDXL-CAP semantic model. For example, target A (e.g., civil defense) can choose that it requires the information about the severity, urgency and location(s) of the incident, while target B (e.g., local emergency manager) requires all available information. The data format of the notifications, which includes the EDXL-CAP payload encapsulation, are serialized as JSON-LD through alignments of the SEMIoTICS core ontology with the EDXL-CAP semantic models produced by the RESCUER project, described in Chapter 4.

5.1.3 Managing Context Data

The *Context Data Manager* is responsible for managing (i.e., allow storage and retrieval of) the context data that were harmonized by the *Input Handler*. The *Context Data Manager* receives the data in accordance with the *EWS Context Model* (the SEMIoTICS core ontology), which can be extended according to EWS functional requirements for domain-specific aspects.

The *Abstraction* component represents the *well-foundedness* of the EWS core ontology, thus, the foundational ontology used and its ontological language. The main benefit of the *Abstraction* component is to maximize the support for interoperability of high level categories w.r.t the most general concepts that are not dependent on the domain, such as kind, role, situation, event, process and system. The SEMIoTICS core ontology and SSN/SOSA are grounded on the UFO (through OntoUML) and DOLCE Ultralite (DUL), respectively. UFO and DOLCE share the same definitions for some conceptualizations, facilitating the alignment between the ontologies extended with them. This component addresses **Req1** and benefits from the harmonization and axiomatization of terms related to “situation awareness” in UFO and its impact on languages for context and situation modeling, presented in the following chapter.

The *Context Data Manager* is responsible for managing the data acquired, acting as a semantic knowledge base and holding a variety of data at different semantic levels. It follows the *Semantic IoT EWS* approach, applying the JDL data fusion model to its knowledge base population process, thus, being able to store pre-processed data (JDL level 0), i.e. raw data acquired as JSON-LD messages; and assessed objects (JDL level 1), i.e. data transformed according to the EWS context model (core ontology). This approach enables data provenance and traceability for future reuse, such as for audit reasons.

Since these data are serialized as JSON-LD messages, we chose to enable the *Context Data Manager* to use a document-oriented NOSQL database to store the messages. Therefore, a document-oriented database is recommended for a fast storing of the JSON-LD messages, which also supports **Req4**. Cloud infrastructures and sharding database approaches should be considered to scale the solution, addressing **Req5** through storage optimization. However, the adoption of these solutions should be aligned to the EWS data storage requirements and in accordance with data privacy laws, such as GDPR.

Equivalently, a *triplestore* (graph database, see Chapter 3) is used for data querying, taking advantage of SPARQL capabilities, supporting to address **Req3** realization when integrating historical data with real-time data. Adopting a sharding approach for *triplestores* should also be considered to address **Req5**. This hybrid approach is suggested by SEMIoTICS, where messages are stored in both document-oriented and *triplestore* databases. Although there are query languages

for JSON, the use of SPARQL is recommended for querying because data are already in RDF format (JSON-LD) and SPARQL is the current standard for semantic querying.

The option for conventional relational databases should also be considered for data warehousing, such as historical aggregated data, and avoided for real-time data processing because of the constraints about high data ingest rates, as discussed in Chapter 3 w.r.t the *Azure MyDriving* IoT application. There are some current initiatives on top of relational databases that can also be considered, such as *TimescaleDB*¹¹¹, an open-source time-series database over the *PostgreSQL* database management system.

5.1.4 Situation Identification and Reaction

The *Situation Identification Manager* is responsible for the detection of emergency situations, i.e. the mechanism that enables the EWS to comprehend the context data as a spatial-temporal whole, equivalent to JDL level 2. Pre-defined patterns of object relationships and the temporal relations of events and other situations are represented through the *Situation Model*, as a formalization of the emergency risk detection rules, which supports realizing **Req7**. SEMIoTICS allows the specification and implementation of complex rules through SML and a CEP engine, including aggregation functions and sliding time windows, which supports addressing **Req8**. CEP is a common component of IoT platforms to correlate data using temporal predicates, as events' relations. For example, EPL provides a set of features, including temporal patterns, that can be used within an ESPER-based environment.

The choice of the specific CEP technology for the EWS should be made according to the specific EWS performance requirements, such as throughput and latency, and event processing querying preference. For example, depending on specific requirements, the CEP engine can provide a query-based language (like EPL and Microsoft stream analytics) or a rule-based language (like SCENE). In particular, SEMIoTICS recommends to use either SCENE (Drools) or SIMPLE (EPL) approaches since they provide MDE transformation rules from SML. In either way, the *Situation Identification Manager* supports addressing **Req6**, since the complex rules under *Situation Types* can be deployed into the system at runtime with minimal impact.

This component also supports addressing **Req3**, since CEP engines have an internal working memory that allows the temporary storage of the incoming data. In addition, the majority of CEP engines are able to connect to external databases and retrieve data according to pre-defined configurations, such as the period which

¹¹¹ <https://github.com/timescale/timescaledb>

the data was imported and aggregation levels. This type of approach also depends on the specific EWS requirements. For example, while an epidemiological EWS for Zika requires the comparison of current data with prior epidemiological studies from the past 10 years, a cardiac EWS requires only the comparison of the incoming heart rate data within the past months, at most.

The *Situation Reaction Manager* is responsible for the reaction to a situation identified by the Situation Identification Manager. Decision support is enabled by the adoption of a platform that allows the end user to design business processes as data flows, the (semi)automatic generation of executable code for the processes, their deployment at runtime and monitoring executions. Several tools address these three requirements, such as microservices' orchestrators, BPM suites, workflow management systems, ESBs, big data integration tools, such as Activiti¹¹² and Node-Red¹¹³ for IoT solutions. An ESB must be applied for the coordination of activities when the EWS needs to integrate services from heterogeneous systems, exposing their functions through APIs that are managed with microservices' gateways. Security and adequate exposure of services are crucial aspects in such a solution. Although a microservices architecture with ESBs is perfectly compatible, this approach should be implemented only if the EWS needs services' orchestration, otherwise it is not worth to implement, maintain and operate such a complex environment. The *Semantic IoT EWS* employs a workflow management system for automating and managing emergency plans as business processes, which is a similar approach. Adopting this approach covers the dynamic deployment and execution of the data flows for decision making, which supports the realization of **Req6**.

Business processes can be triggered through the platform whenever a situation instance is activated. For example, emergency plans specify evacuation actions that can be implemented as data flows within an ESB and called as a subscriber of a service exposed by the service broker(s). Since ESB is a middleware that allows multi-broker and multi-cluster settings for scalability and resiliency, this component also addresses **Req5**. Industry-oriented event-driven platforms that can implement these middleware capabilities, such as *Apache Kafka* and *StormMQ*, run high performative protocols for IoT, such as AMQP and MQTT, supporting the realization of **Req4**.

Messaging activities within a process (or workflow), representing emergency plans, are used to issue warnings for distinct targets that are specified into each activity. Usually, an ESB provides a way to configure the messaging activity to post data to a RESTful API with JSON payload. The SEMIoTICS brokering function relies on using such a configuration, where the ESB messaging component posts

¹¹² <https://www.activiti.org/>

¹¹³ <https://nodered.org/>

data about the target’s requirements to the *Output Handler* along with the communication channels to be used, supporting the realization of **Req9**.

5.2 Semantic Model-Driven Methodology

The SEMIoTICS framework follows the semantic MDE approach (Chapter 2), giving emphasis to the semantic improvement at the conceptual modelling level and multiple-purpose MDE transformations. The SEMIoTICS MDE methodology is illustrated in Figure 5.2.

Figure 5.2. The SEMIoTICS semantic MDE methodology

The methodology targets design time activities in two phases:

- . **Specification phase:** it has two parts, namely a part that is about language definition (conceptual) and a part that is about design specification (using the languages defined). For the latter, iterative design cycles can be applied, with each next design cycle driven by the validation results from the previous cycle. The conceptual part presents the definition of real-world constructs in a foundational ontology, which are reflected in the modelling languages, where temporal and structural aspects are addressed by extending UFO. The design part adopts the improved modelling languages as graphical modeling languages for context

(OntoUML), situation (SML) and reaction (BPMN). Horizontal (same abstraction level) exogenous (different languages) transformations are applied at the specification level, from the context model to the situation model, and from the situation model to the action model. Vertical model transformations are adopted for code generation, transforming the models in specific implementation technologies.

. Implementation phase: while the conceptual part supports the modelling language constructs of the specification part, model transformations assure that appropriate technology implementations are generated. Therefore, as usual in MDE, the implementation is (partially) generated by model transformations. These transformations realize the specification as executable pieces of software, where IoT platform components are adopted.

5.2.1 Ontology-Driven Situation Modelling

In particular, the SEMIoTICS framework adopts one specific type of semantic MDE methodology, the *Ontology-Driven Conceptual Modelling* (ODCM) methodology, described in Chapter 2. ODCM for the development of situation-aware applications plays the role of specification-based approach for situation identification of IoT EWS, enabling that patterns of hazardous events to be represented through models. The modelling languages used to specify such applications must support not only the design of structural aspects, but also temporal aspects and their relations [186].

SEMIoTICS employs spatial and temporal logic mixed to distributed CEP technology. When using a learning-based approach, such as neural networks, a large number of features can affect the computation speed of the algorithm, thus, impacting on the real-time processing performance of the EWS. As described in Chapter 2, in pervasive computing, a hybrid approach is suggested to integrate represented and derived knowledge from the available data sources. Therefore, SEMIoTICS also considers the learning-based approach (e.g., machine learning), albeit not in a direct way, i.e., the algorithm resulted from the learning-based approach should be understood by the EWS administrator, and then, the complex rules underlying the algorithm should be inserted in the EWS through the semantic model-driven transformation approach. This type of using learning-based approach is nowadays called “explainable AI” [265].

SEMIoTICS considers both structural and behavioral models as suggested in [133]. We propose the use of a high expressive modelling language that includes clear real-world semantic distinctions to model the observed contextual elements and the situation patterns to be identified, i.e., ODCM approach with structural modelling grounded in UFO/OntoUML for context modelling. OntoUML was chosen to address the expressiveness requirement to specify the context of EWS.

The SEMIoTICS core ontology, presented in the following Chapter, was designed with OntoUML and includes basic elements for the specification of contextual elements, such as health units, patients, risks, installations and plans. These elements can be mapped from the context model specification onto implementation code, i.e., any other object-oriented language, such as POCO.

We adopted UFO [155] to support the conceptualization of the structural and temporal aspects because: (1) UFO is a well-established foundational ontology, especially for *endurants*, having strong foundations on philosophy, linguistic and cognition psychology; (2) UFO covers part of the required temporal aspects, such as events, participations, temporal relations and situations [182]; (3) UFO's ontological assumptions are materialized through syntax rules in OntoUML ontological language, which was used to develop the SEMIoTICS core ontology. As described before, we adopted three modelling languages to enable the design of IoT EWSs with different purposes: (1) the context, for structural modelling; (2) the situation, for patterns and rules modelling; and (3) the behavior, for action (as reaction to a situation detected) modelling.

SML is a relevant element of SEMIoTICS framework because through situation modelling one can describe specific patterns (a *Situation Type*), to be detected by the EWS. It is an appropriate language to characterize a set of rules and enables the reuse of sets of rules to define more complex situation types, relating them in time by their activation status (active or inactive). Furthermore, the situation types defined in a SML model, which is linked to OntoUML model through the contextual elements, can be assessed during design-time by the ODCM validation and verification process (the *Alloy* approach), where the designer can visualize possible instances of the model and make changes on it when necessary, enriching its semantics. This approach allows syntax verification of the context model and visual validation through simulation (also called situation assessment). The software designer uses this validation process repeatedly, until sufficient confidence in the models is achieved. Then, CEP code can be automatically generated from the SML model.

In SEMIoTICS, reactive actions to a detected situation are described as a behavior model with BPMN, which offers graphical syntax and ontological foundations of events [266]. Therefore, SEMIoTICS also includes the design of the reaction to a *Situation Type* with BPMN, i.e., the processes to be executed when a situation is detected (activated), as emergency plans. The designer can also specify template-based messages to be sent during the process execution. Once the context model is described with OntoUML, the situations with SML and the behavior with BPMN, the designer can also simulate instances of reaction models. The use of BPMN leverages SEMIoTICS for processes documentation, service compositions, process simulation and mining, supporting higher quality for the representation of emergency procedures. Since the BPMN *Catch Event* element

assumes that the process is waiting to be triggered in a “receiving mode”, being exogenous, homeomeric and cumulative [266], it can represent the situation activation (detection) event. Therefore, the *Catch Event* acts as an integration point between SML and BPMN models.

Figure 5.3. Basic mapping rule from SML (a) to DRL and (b) to EPL [186]

When the specification of an EWS reaches the required quality, an ontological model-driven engineering process is performed. The implementation is obtained through model transformations from the specification to the realization (implementation) in some specific technologies. Those transformations generate the EWS code, which becomes less error prone because of this automated generation. Different “flavors” of technology can be used, such as CEP rule-based and query-based systems. The idea is to avoid coupling the framework to a specific component, API or platform. Nevertheless, SEMIoTICS is leveraged by the model transformations that transform SML models to rules that can be executed by the SCENE (Drools) and SIMPLE (ESPER) platforms [186]. With these platforms, optimized algorithms of pattern matching are performed while avoiding unnecessary repetitions for unmodified conditions. It reduces the effort necessary for situation detection by benefiting from the truth maintenance system, which is responsible for ensuring logical integrity of facts in the working memory. Figure 5.3(a) illustrates a transformation (mapping) from SML to the language that SCENE uses, the Drools Rule Language (DRL), while Figure 5.3(b) illustrates the transformation to EPL, used by the SIMPLE approach.

5.2.2 An Example Case: IoT EWS for Tuberculosis Outbreaks

Suppose that an EWS is required to detect tuberculosis (TB) epidemics in a certain *geographical region*, e.g. a *city*. The system can use information from *hospitals* within the *city*, which provide relevant real-time data about *patients* who receive *treatments* in these hospitals. Examples of provided data include measurements of a patient’s body *temperature* and results of a patient’s medical examination. Below, we discuss the specification of a context model and a situation model for this example case.

The context model is provided as an extension of the OntoEmerge core ontology, shown in Figure 5.4. Notice that the concepts of the ontology are

Figure 5.5. (a) Possible contagion situation (SML); (b) Reaction model (BPMN)

Figure 5.6. (a) Mapping from SML to DRL. (b) TB disease situation (SCENE) [158]

For the implementation, Java classes are generated from the concepts in the context model, such as *patient* and its attributes, as well as its *treatment* relation to *hospital*. Situation types (e.g. TB disease) are also represented as Java classes within the SCENE environment, and situation rules are mapped to the adapted Drools domain specific language (DRL) for situations management, as illustrated in Figure 5.6. The warning reaction model is attached to the TB epidemics situation. In the

BPM suite, the reaction is designed and each task is an activity in which an SMS message is sent.

5.3 Standard-based FAIR Data for IoT EWS

The FAIR principles, presented in Chapter 2, are automatically observed if the SEMIoTICS methodology is followed. Although all principles should be followed, here we give emphasis to the interoperability principles. The first principle (I1) guides the use of a formal and broadly applicable language for knowledge representation of both data and metadata. In practice, it means that data model languages, such as RDF, are used to implement formal ontologies, which is already a core aspect of SEMIoTICS, discussed in last section.

The use of vocabularies for the representation of data and metadata that follow the FAIR principles (I2) emphasizes the need of adopting the terminology used by open, community-accepted vocabularies that are published in appropriate knowledge-exchange format. SEMIoTICS adopts this principle, albeit it is subjective and is related to a research topic extensively discussed by the formal ontology community, i.e. how to choose the best ontology for each specific application domain. SEMIoTICS extends this requirement by adopting the best practices of ontology engineering described in Chapter 3, more specifically the best practices #8 and #12, which regard common ontology pitfalls and ontology reuse, respectively. SEMIoTICS guides to apply these best practices as objective metrics when validating existing semantic models.

In addition to these quality criteria, SEMIoTICS also requires the comparison of the payload of the semantic models, i.e. the message size when serializing the data, to avoid verbose messages that will impact on the performance of the EWS and the involved costs. Finally, SEMIoTICS also guides to the use of semantic models that are standardized by recognized international organizations, such as W3C, OGC, ETSI, HL7, among others.

The use of qualified references (I3) refers to the relationships among (meta)data that need to have explicit semantic meaning, i.e., the relationship within and between data should be semantically rich. This requirement can be addressed when the ontology is grounded in a foundational ontology that explicit the types of relationships that certain elements can share. For example, in UFO, the mediation relationship refers to the intrinsic mediating property that a relator requires to exist, usually linked to a role. Although the majority of semantic models does not apply this type of foundational characterization, the use of a foundational ontology can guide the classification whether the relationship being analyzed has rich semantics or not.

SEMIoTICS follows a similar approach as the *Personal Health Train* (Chapter 2) by guiding the storage of all context data in one or more FDP repositories. This is performed through the REST/RDF component that is able to access the data stored in a context database and forward the data to a FDP data repository. In addition to the data processed by the ontology management component, the data produced/inferred by the EWS, i.e. the data messages created by the output handler component, can be stored in a FDP repository by an application that subscribes to receive all EWS messages.

Finally, the artefacts responsible to run the EWS, including the EWS components that are specified and realized by the MDE approach, as well as the situation types and emergency plans configured by the EWS administrator, should also be stored in a FDP repository. This way, not only the context data are transformed in FAIR data, but also the algorithms are transformed in FAIR data, being able to interoperate the “knowledge”, i.e. both comprehension and decision capabilities of the EWS, with other parties. An important aspect to be highlighted is that SEMIoTICS also adopts the accessibility characteristics defined by the GO FAIR metadata and implemented by the FDP, which includes security concerns with authentication metadata and authorization mechanism following the role-based access control, the most common approach to restricting data access for only authorized users. Notice that this orientation is complementary to the context data management approach of SEMIoTICS, since IoT solutions may use different data formats and languages “that allow specific and scalable manipulation of data for pattern recognition and knowledge discovery, and these representations may or may not be FAIR” [187]. Therefore, storing data in FDP repositories should be performed by resources that do not impact the real-time processing, such as dedicated jobs running in different servers.

SEMIoTICS adopted the current FAIRness metrics [130] as the FAIR validation approach of the EWS, aiming at measuring the EWS *FAIRness*. Each of these metrics specifies exactly what is being measured and why, including the required input for the validation, the measurement process and the definition of valid results. The use of these metrics by SEMIoTICS means that the validation plan of an EWS developed with SEMIoTICS should include the necessary tests to validate the principles according to the metrics.

5.4 Concluding Remarks

This chapter introduces the “SEmantic Model-driven development for IoT Interoperability of emergenCy serviceS” (SEMIoTICS) framework, a “meta-system” for the development of interoperable IoT EWS. SEMIoTICS includes a reference architecture and a semantic MDE methodology with ODCM that guides

how to apply different technologies at design time. The main requirements that SEMIoTICS needs to address are derived from the challenges identified in the previous chapter. Each SEMIoTICS component is described, explaining which requirements are expected to be realized. The semantic MDE methodology is detailed and exemplified through an example case about an epidemiological surveillance EWS for tuberculosis outbreaks. The methodology also considers the adherence to the FAIR data principles, to improve the interoperability of the data produced by an IoT EWS developed with SEMIoTICS. In summary, this Chapter shows how SEMIoTICS provides guidance such that general requirements on interoperable IoT EWS are satisfied and is in line with proven/important results from previous work.

Current proposals for IoT-based EWS only partially address the semantic integration of a variety of data sources along with processing in time-critical applications and data analysis for effective responses. The SEMIoTICS framework addresses this problem by applying different modelling languages, semantic models and technologies, supported by different theories, such as the *SA theory*.

SEMIoTICS was designed to be general enough to be applicable to any type of emergency, thus, not being oriented towards one specific domain. The SEMIoTICS framework presents a sound conceptual foundation, reflected by the languages used for the specification phase, which are a worthwhile investment for achieving semantic interoperability. Therefore, the semantic interoperability is improved from the conceptual modelling phase to the implementation. The expressiveness of the modelling languages adopted (OntoUML, SML and BPMN) supports a better understanding of the context data and their relationships. In particular, SML supports the representation of the situation identification mechanism and BPMN supports the modeling of reaction to different situation types. The MDE methodology allows the assessment of these models at design-time, enabling the early discovery of conceptual errors before coding. The adoption of a MDE transformation approach for code realization automatizes the manual repetitive activities, where benefits such as a higher code quality with less error prone implementation are expected. Therefore, SEMIoTICS bridges the research from the formal ontology theories to the big data technologies.

In the implementation level, the RESTful microservices architecture with JSON-LD supports both scalability in the cloud and an adequate total processing time performance. In addition, JSON-LD supports semantic integration as a serialization format for RDF messages, allowing the execution of multiple semantic translations of incoming streaming data at runtime. The multi-database approach of SEMIoTICS deals with both historical data storage and retrieval requirements, the former being realized by a high-throughput document insertion database and the latter being realized by *triplestore(s)* for semantic data queries (SPARQL) and relational database(s) for conventional data warehousing. Furthermore,

SEMIoTICS improves decision support by adopting a tool that enables the execution and monitoring of emergency plans at runtime. Finally, warning messages issued by an IoT EWS developed with SEMIoTICS follow the EDXL-CAP standard through its semantic model, being able to carry the adequate information according to the targets' requirements.

Improving the SEMIoTICS Foundations

The SEMIoTICS framework should have a sound foundation in order to support the definition of its reference architecture components and the semi-automatic generation of IoT EWSs. Specifically, the foundation is needed to enable semantic integration of different IoT-related ontologies, to allow the modelling of real-world situations for the disaster management domain, and to support the detection of situations of interest in this domain. This chapter presents the theoretical improvements that have been developed w.r.t. existing solutions. Section 6.1 presents a method for developing a mapping between two ontologies, and it specifically presents a mapping between SSN/SOSA and SAREF that was developed using the proposed method. The *Input Handler* component (performing semantic translations) in the reference architecture benefits from these semantic translations. Section 6.2 presents a proposal for a better ontological underpinning of situations, which includes the consideration of existing theories for situations and their impact on UFO. The *Context Data Management* component (handling contextual data) in the reference architecture benefits from these improved foundations. Section 6.3 presents a proposal for an improved version of SML, which was necessary to make it consistent with the proposal made in section 6.2, but also to satisfy IoT EWS requirements. The *Situation Identification Manager* component (detecting situations of interest) in the reference architecture benefits from SML 2.0. An example case illustrates the modelling of an IoT EWS for epidemiological surveillance of Zika virus outbreaks. Finally, Section 6.4 concludes this chapter by presenting the lessons learned and theoretical contributions of SEMIoTICS.

6.1 Sematic Translations for the IoT

As described in Chapter 3, one of the most popular ontologies for semantic interoperability in IoT is the W3C SSN/SOSA, which is considered an ontological

foundation for diverse IoT initiatives. The ETSI SAREF ontology evolved from the need of the smart appliances domain to have a reference ontology that can help achieve interoperability of smart devices and service platforms in an IoT ecosystem. The integration of systems that use these ontologies requires mechanisms to translate one ontology into another.

In this section we introduce the alignment between SSN/SOSA and SAREF, identifying mapping alternatives and proposing patterns that can be re-used to define various transformations with different trade-offs between quality and efficiency. The semantic translations are further validated in Chapter 9 in twofold: (1) by creating and answering competency questions regarding the SSN/SOSA use case for a wind sensor (*Vaisala WM30 wind sensor*¹¹⁴), and (2) by defining a semantic matching process leveraging the semantic mediator component (IPSM) of the INTER-IoT project compared to a “raw” SPARQL approach. This validation measures the semantic translations’ accuracy and efficiency.

6.1.1 Model-Driven Transformations Methodology

A methodology is required for designing, implementing and validating the semantic translations between SSN/SOSA and SAREF, supported by a tool for ontology alignment [58, 230]. Our methodology for semantic translations development follows a common software engineering approach that considers specification and implementation phases (design time). The specification describes in natural language the possible mappings and their respective rules, showing how an instance of the source ontology can be represented with a target ontology, i.e., transforming the ABox of ontology A (source) into ABox of ontology B (target). A mapping can have two roles: (1) to specify the rule for alignment (harmonization/translation) or verification of model constructs; and (2) to create missing elements.

Figure 6.1.
Methodology for
semantic
translations based
on model
transformations

¹¹⁴ <https://www.w3.org/2005/Incubator/ssn/ssnx/meteo/>

This methodology is based on a typical MDE transformation development process, presented in Chapter 2 [133], to describe transformation mappings in terms of a source and a target meta-model, such as the mappings performed between EDXL-TEP and HL7 (Chapter 4). As illustrated in Figure 6.1, the main difference between the traditional MDE approach and our methodology is that the ontology conceptualization (TBox) play the role of source and target metamodels, while the ontology instances (ABox) play the role of source and target models. In addition, RDF plays the role of metametamodel because it is the language used to implement ontologies. A MDE transformation can be developed as an unidirectional or a bi-directional transformation. The best practices show that, when creating a bi-directional transformation with a high number of meta-model elements, a cyclical process should be used [133].

The set of mappings between two metamodels (or ontology Tbox) is a transformation definition, which can be expressed using a transformation language. For example, these mappings can be used to transform from the *WM30 ontology* (an instance of SSN/SOSA) to an equivalent instance of SAREF. For the implementation of these mappings, declarative, imperative or hybrid model transformation programming styles can be used. For example, the IPSM (INTER-IoT) is a hybrid template-based tool, which allows the user (translation developer) to declare matching rules with traceability support, while it is imperative by executing the translations rule by rule in a pre-defined order. Internally, IPSM creates a SPARQL query for each rule. For each rule it executes the respective SPARQL to find instances, generating a new ontology instance with the transformed constructs, which is used as input for the next rule execution.

A pure imperative semantic translations implementation application can use a RDF library, such as Apache JENA¹¹⁵ (Java), and implement the mappings manually. Therefore, two implementation strategies are considered here: (1) through a hybrid approach such as the IPSM template-based approach, i.e., configuration files (XML format) following the IPSM transformation language (defined with RDF); or (2) through an imperative style, i.e., using a programming language with an API that provides RDF features (e.g., SPARQL), such as JENA (for Java) or dotNetRDF¹¹⁶ (for .NET).

Our methodology includes a specification strategy to avoid conceptual errors during the implementation, which is a common approach in software engineering. Moreover, directly implementing the mappings forces a preliminary choice of the technology of the transformation runtime, which couples the mappings with a technology, thus, limiting the reuse of the mappings. The SEMIoTICS strategy

¹¹⁵ <https://jena.apache.org/>

¹¹⁶ <https://www.dotnetrdf.org/>

addresses these issues by leveraging the MDE approach with a specification artifact targeted to translation developers.

The specification strategy for the transformation definition adopts the natural language with pseudo-code approach instead of a formal language. This approach uses a spreadsheet to list source and target elements, their data types, the transformation rule to be applied and one or more examples. The guidelines of the SEMIoTICS specification strategy explain the steps that the semantic translations developer should follow:

1. Specify an unidirectional transformation with a selected sub-set of the source ontology (TBox elements) that should be covered by the translations (a minimal set), and identify which elements of the target ontology are required to represent this set.
 - a. For each element of the source, identify possible structures (classes, object and data properties) in the target that have a match. Matching elements can be found by lexicon analysis (term, synonymous and structure) and ontological analysis (term definition and intended semantics).
 - b. Annotate whether the matching element(s) is (are) found: a direct 1:1 mapping or 1:n (a pattern in the target) or match not found. If a 1:n is found, then describe in (natural language) the rule that should be applied to generate the n elements in the target from the element in the source.
 - c. Repeat these steps for the optional elements in the target.
2. Simulate transformations with a set of pre-defined instances to validate the mapping. These instances must be selected according to real-world examples for better functional validation. If an error is found, then the mapping should be changed. Transformations should be validated cyclically until the mappings correctly generate the intended model.
3. Specify the opposite direction with the same elements of the first step, and validate/correct with simulations – repeat steps 1 and 2 from the target to the source.
4. Assess whether meaning is lost when executing the transformations in sequence, reflecting the translation accuracy [58], which can be checked by comparing the input ontology instance x to consecutive translation executions in each direction:

$$T(T(x)_{A \rightarrow B})_{B \rightarrow A}$$

- $T(a)_{A \rightarrow B}$ produces the ontology instance b (described with ontology TBox B) from ontology instance a (described with ontology TBox A).
- $T(b)_{B \rightarrow A}$ the opposite direction.

Therefore, the translation should be executed twice and consecutively, and the output should be the same (or similar) input ontology. This step supports finding possible conceptual errors, enabling the early correction of the mappings before implementing. The missing data should be compared to the elements annotated as “no match” (step 1.b), showing that these elements should be created. For example, while SAREF provides a *saref:hasManufacturer* object property, SSN/SOSA does not provide a similar element. Therefore, SSN/SOSA lacks on a representation for this property, which could be created as an extension of SSN/SOSA ontology.

The translations specification cycle should be performed as an interactive process that is executed while there are elements to be addressed with exhaustive verification and validation, following the ODCM approach. Once achieving the required quality level of the bi-directional transformation specification, the transformations can be implemented.

The semantic translation specifications are leveraged by the application of ontological analysis over the TBox of the involved ontologies, e.g., SSN/SOSA and SAREF. This is an analysis of the statements that describe their conceptualization. Therefore, a reverse engineering should be performed, applying ontological analysis with ODCM to support the understanding of the elements being used. Since SSN/SOSA is extended from DUL, and DUL is the lightweight foundational ontology of DOLCE, we used DOLCE’s categories as a reference to map from SSN/SOSA to SAREF.

For example, the object property *ssn:hasSubSystem* is a *dul:hasPart* and means “a schematic relation between any entities”. Based on this definition, we sought for the possible relations that have the same semantics for this mapping in SAREF, finding that *saref:consistsOf* has the same meaning: “a relationship indicating a composite entity that consists of other entities (e.g., a temperature/humidity sensor that consists of a temperature sensor and a humidity sensor)”.

Although this process is time consuming, error-prone and automatic techniques based on natural language processing (NLP) are commonly used for finding ontology similarity [230], sometimes such ontological analysis can produce more semantically accurate set of ontology alignments that do not rely only on lexicon approaches (terminology and data structure), but also on their intended meaning (descriptions), which requires human cognition [267].

6.1.2 Mappings Between W3C SSN/SOSA and ETSI SAREF

The methodology described in the previous section was executed, producing the mappings between SSN/SOSA and SAREF. In this study we considered the characteristics of a sensor, and for each characteristic we produced the mappings

between SSN/SOSA and SAREF by following the methodology of the previous section. The characteristics of a sensor, in the given order:

- (M01) main descriptive properties of a sensor;
- (M02) the composition/aggregation of devices with other devices (e.g., sensors, actuators);
- (M03) the property observed by a sensor;
- (M04) the observations (measurements) that a sensor makes;
- (M05) the geo-location properties of the devices.

Notice that a mapping can refer to an element that may have been referred in a previous mapping (e.g., *saref:Sensor* in M01 and M03), which requires a special treatment when implementing the translation, either by storing the original ontology in memory for lookups or by executing the mappings continuously considering replacing the elements translated in each mapping rule (as the IPSM approach). All descriptive data properties from RDFS are copied from source to target, since RDFS is used by both SSN/SOSA and SAREF. Each mapping is detailed below, their implementation (with the IPSM language) is available in Annex 1 and explained in Chapter 7.

M01. Sensor Properties

The *sosa:Sensor* and *saref:Sensor* are ontologically equivalent. While the main element of SSN/SOSA is the *sosa:Sensor*, a subclass of *ssn:System*, in SAREF the main element is *saref:Device*, which can be specialized as a *saref:Sensor*, with restriction *saref:hasFunction* a *saref:SensingFunction*. The properties of *sosa:Sensor*, inherited from *ssn:System*, are used for the transformations to *saref:Sensor*, which inherits *saref:Device* properties. *sosa:Sensor* has the restriction¹¹⁷ of at least one instance of *ssn:implements* representing the sensor's implementation *sosa:Procedure*, which means the "method specifying how to make an *Observation*" when instantiated by a *sosa:Sensor*. This is a similar concept to the *saref:SensingFunction*, which is a specialization of *saref:Function*. Therefore, this mapping defines the ontological equivalence of *sosa:Procedure* and *saref:Function*, which share the same conceptualization.

A *saref:SensingFunction* has a restriction of at least one instance of *saref:hasSensingRange* to some *saref:Measurement*. In SSN/SOSA, the representation of this conceptualization is performed by using the *ssn-system:hasOperatingRange* property, linked from a *ssn:System* to a *ssn-system:OperatingRange*, which has the restriction of instantiating a *ssn-system:Condition* (a *ssn:Property*). The *ssn:Property* and *saref:Property* classes are ontologically equivalent (exploited in mapping M03) and a *saref:Measurement* is linked to a *saref:Property* through the *saref:relatesToProperty* object

¹¹⁷ The restriction is the consequence of the minimal cardinality

property. Therefore, this mapping considers these patterns for sensor definition in both sides:

SSN/SOSA:

sosa:Sensor ssn:implements ?sosa_Procedure
ssn-system:hasOperatingRange ?ssn-system_OperatingRange
?ssn-system_OperatingRange ssn-system:inCondition ?ssn-system_Condition

SAREF:

saref:Sensor saref:hasFunction ?saref_SensingFunction
saref:hasSensingRange ?saref_Measurement
?saref_Measurement saref:relatesToProperty ?saref_Property

M02. Device Composition / Aggregation

The *ssn:hasSubSystem* and *sosa:hosts* are specializations of *saref:consistsOf*. M02 checks the composition relationship of a device, i.e., the components that are part of a device. In SSN/SOSA there are two ways to represent a device composition, either in terms of system with *ssn:hasSubSystem* or in terms of physical deployment with *sosa:hosts*. The former is applied between two *ssn:System* and the later between a *sosa:Platform* and a *sosa:Sensor* (or a *sosa:Actuator* or a *sosa:Platform* or a *sosa:Sampler*). In SAREF, the composition relation is enabled through the object property *saref:consistsOf*, relating two *saref:Device*. The first version of SSN provided a similar concept to *saref:Device* (*oldssn:Device*), but it was eliminated in the second version, which guides the transformation from *oldssn:Device* to an element that should be a subclass of both *sosa:Platform* and *ssn:System*.

Therefore, when a *ssn:hasSubSystem* or a *sosa:hosts* is used, a mapping for *saref:consistsOf* should be performed. The main issue here is the mapping back from SAREF to SSN/SOSA, because the transformation mechanism should be prepared to check whether the *saref:consistsOf* was used to describe the system composition (as a more general concept) or how the device is mounted (a physical/tangible aggregation). This issue is tackled by checking the class instantiating the *saref:consistsOf* object property. If it relates to a physical representation such as *saref:Sensor* (*saref:Device*), then a *sosa:Platform* with *sosa:hosts* should be created to represent it, otherwise, if it relates for example to temporal aspect (*saref:Time*) or a profile specification (*saref:Profile*) then a *ssn:System* with *ssn:hasSubsystem* should be created to represent it. Notice that a recursive algorithm should be performed with the components of the device using M01, allowing the translations for representing multiple composition, e.g., a medical device that is composed by a body temperature sensor and an ECG device, which is composed by several sensors (leads).

M03. Property Measured by a Sensor

The *sosa:observes* (or *ssn:forProperty*) and *saref:measuresProperty* object properties are ontologically equivalent, used to represent the property being measured by a device. For example, a wind sensor is able to observe both wind direction and wind speed. Therefore, the *sosa:observes* of a *sosa:Sensor* is mapped to *saref:measuresProperty* of a *saref:Device*. While *saref:measuresProperty* is broader for sensors and actuators, i.e., a restriction of *saref:Device* (*saref:measuresProperty* only *saref:Property*), which is superclass of *saref:Sensor* and *saref:Actuator*, *sosa:observes* is a restriction of *sosa:Sensor* (*sosa:observes* only *sosa:ObservableProperty*) and *ssn:forProperty* is a restriction of *sosa:Actuator* (*ssn:forProperty* only *sosa:ActuatableProperty*).

Therefore, this mapping requires a first rule to translate the property itself, where *saref:Property* is equivalent to *ssn:Property* (superclass of *sosa:ObservableProperty* and *sosa:ActuatableProperty*). When translating from SAREF to SSN/SOSA, when linked via *saref:measuresProperty*, it is required to check the type of *saref:Device* (*saref:Sensor* or *saref:Actuator*) to correctly transform into either *sosa:ObservableProperty* or *sosa:ActuatableProperty*, respectively. The other way (SSN/SOSA to SAREF) is simpler: both *sosa:ObservableProperty* and *sosa:ActuatableProperty* are mapped to the *saref:Property* when linked via *sosa:observes* or *ssn:forProperty*. The second rule transforms the relationship itself, translating from *sosa:observes* and *ssn:forProperty* to *saref:measuresProperty* (direction SSN/SOSA to SAREF); and from *saref:measuresProperty* to *sosa:observes* if *saref:Device* is *saref:Sensor* and to *ssn:forProperty* if *saref:Device* is *saref:Actuator* (direction SAREF to SSN/SOSA). At last, when transforming from SSN/SOSA to SAREF, this mapping needs to instantiate the *saref:isMeasuredByDevice* relationship from the *saref:Property* to the respective *saref:Device*.

M04. Observations (Measurements) that a Sensor makes

The prior mapping focused on the relationship that represents the property being measured by a device. This mapping focuses on the representation of the atomic observations (measurements) made by a sensor. Therefore, it describes the most important mapping for real-time sensor monitoring, a complex transformation that maps *sosa:Observation* and *saref:Measurement*, which are ontologically equivalent. Both *sosa:Observation* and *saref:Measurement* can be classified as atomic event (in terms of UFO) occurring in a time instant (represented with *sosa:resultTime* and *saref:hasTimestamp*, respectively), w.r.t. a measured value (a property of *sosa:hasResult* and *saref:hasValue*) made over a property (*sosa:observedProperty* and *saref:relatesToProperty*) in a specific unit of measure (a property of *sosa:hasResult* and *saref:UnitOfMeasure*).

The *sosa:Observation* is linked to a *sosa:Sensor* through the *sosa:madeBySensor* property, while *saref:Sensor* is linked to a *saref:Measurement* through the *saref:makesMeasurement*. Therefore *sosa:madeBySensor* and *saref:makesMeasurement* are

ontologically inverse. A *sosa:Observation* describes the value observed and its unit through the *sosa:hasResult*, which points to a *sosa:Result* that carries the *owl:hasValue* and *om:hasUnit* properties for a literal (e.g., an integer) and a unit of measure (e.g., degree Celsius), respectively. Similarly, a *saref:Measurement* is linked to the value and unit through the *saref:hasValue* and *saref:isMeasuredIn* properties for a literal and a *saref:UnitOfMeasure*, respectively. The *sosa:Observation* links to the property being observed (e.g., temperature) through the *sosa:observedProperty* property, while *saref:Measurement* represents the property being observed through the *saref:relatesToProperty* property. As described in M03, *ssn:Property* and *saref:Property* are ontologically equivalent. Finally, the temporal representation of a *sosa:Observation* is enabled through the *sosa:phenomenonTime* property, which links to a *time:Instant* and represents the specific timestamp through the *time:inTimePosition* property. Similarly, *saref:Measurement* represents the occurrence time through the *saref:hasTimestamp* property. Therefore, this mapping considers these patterns for the semantic translations of observations:

SSN/SOSA:

```
sosa:Observation sosa:madeBySensor ?sosa_Sensor
sosa:observedProperty ?ssn_Property
sosa:hasResult ?sosa_Result
sosa:resultTime ?time_Instant
sosa_Result qudt-1-1:unit ? qudt-1-1_Unit118
```

SAREF:

```
saref:Sensor saref:makesMeasurement ?saref_Measurement
?saref_Measurement saref:relatesToProperty ?saref_Property
saref:hasValue ?xsd_float
saref:hasTimestamp ?xsd_dateTime
saref:isMeasured in only ?saref_UnitOfMeasure
```

The main issue found in this mapping regards the *sosa:Observation* restrictions:

- *sosa:hasFeatureOfInterest* EXACTLY 1

The *sosa:FeatureOfInterest* represents “the thing whose property is being estimated or calculated in the course of an Observation”. For example, if an observation of a tree height is made by a sensor, then the tree is a *sosa:FeatureOfInterest*, while the height is a *sosa:ObservableProperty* and the

¹¹⁸ SSN/SOSA does not provide an element for unit of measurements because it is not in its scope and other ontologies provide a way of modeling results as quantity values. The SSN/SOSA examples use the Quantities, Units, Dimensions and Data Types Ontologies (<http://www.qudt.org/>), but other can be used, such as the Ontology of Units of Measure (<http://www.ontology-of-units-of-measure.org/resource/om-2/>).

value itself is the *sosa:Result*. In SAREF, this conceptualization can be achieved by relating two devices through the *saref:Property* element, which a *saref:Measurement* relates to (*saref:relatesToProperty*) and a *saref:Property* that *saref:isControlledByDevice* *saref:Device*. In the example aforementioned, a sensor makes measurement of the height (*saref:Property*) and this property is controlled by the tree (a *saref:Device*)¹¹⁹.

- *ssn:wasOriginatedBy EXACTLY 1*
Every *sosa:Observation* is originated by a *ssn:Stimulus*, which is “an event in the real world that 'triggers' the Sensor”. SAREF allows this representation through the *saref:Command* element, which is a “directive that a device must support to perform a certain function”, thus, related to the pattern:
saref:Device saref:hasFunction ?saref_Function
?saref_Function saref:hasCommand saref_Command
- *sosa:phenomenonTime EXACTLY 1*
The phenomenon time is not necessarily the same of *sosa:resultTime*, which represents an instant when the *sosa:Observation* is completed. The *sosa:phenomenonTime* refers to the time structure that the *sosa:Result* applies to the *sosa:FeatureOfInterest*, which can be an instant or an interval or other compound temporal entity. If it is an instant, then our interpretation of an observation as an atomic event (UFO) is correct. Otherwise, if it is an interval, then an observation can be classified as a complex event, which requires a composition relationship of observations, not presented in SSN/SOSA. Therefore, when transforming from SAREF to SSN/SOSA, *saref:hasTimestamp* is mapped to both *sosa:resultTime* and *sosa:phenomenonTime*. The main issue is in the other direction, for example if an observation has *sosa:resultTime* linked to a different time instant than *sosa:phenomenonTime*. This is a case where semantics is lost.

M05. Device Geo-Location

While in SSN/SOSA the GeoSPARQL approach is recommended to represent geospatial objects, in SAREF the conventional W3C approach based on the WGS84 ontology is suggested. In particular, GeoSPARQL provides a sophisticated way to represent latitude and longitude through the *geosparql:asWKT* property, which links to a *geosparql:wktLiteral* with value as the string *Point(#latitude, #longitude)*, where

¹¹⁹ Although this is a possible way to instantiate SAREF through this pattern, the term “controlled” is not the most adequate to reflect the feature of interest. Ultimately, this type of issue affects the ontology semantics, impacting the representation of common understanding.

#latitude and *#longitude* are float values. The WGS84 approach simply introduces a class *geo:Point* which represents latitude and longitude through the *geo:lat* and *geo:lon* properties, respectively. The example below shows an equivalent instance represented with SSN/SOSA and with SAREF of a mobile device located at latitude 39.43668175 and longitude -0.3343958.

SSN/SOSA:

```
sosa:Platform geosparql:hasGeometry ?geosparql_Geometry
?geosparql_Geometry geosparql:asWKT "Point(39.43668175 -0.3343958)"
```

SAREF:

```
saref:Device wgs84_pos:location ?wgs84_pos_Point
?wgs84_pos_Point wgs84_pos:lat 39.43668175
                    wgs84_pos:long -0.3343958
```

6.2 Situation Foundations for Context Modelling

Chapter 5 presented the MDE methodology adopted by SEMIoTICS for the semi-automatic generation of semantic IoT EWS from their specification artefacts, where the context model plays a crucial role for the disaster situation identification mechanism. Since the core ontology that was used to develop the context model depends on UFO categories and their representation in OntoUML, we argue that concepts related to situations in UFO have to be harmonized, as discussed in Chapter 2. However, converging these high-level constructs towards a common sense in a well-founded manner, i.e., grounded in philosophical interpretations, is a complex task. The idea behind the semantic improvement of UFO with well-established theories, in particular the *Situoid* and *SA theory*, is to leverage the semantic interoperability of IoT EWSs developed with SEMIoTICS. Therefore, ultimately, by improving the foundations of the DM core ontology, the IoT EWS using the ontology as a context model can provide a higher quality situation awareness and, thus, improved semantics.

As foundations to support the choices made here we studied the *Barewisean* situation semantics theory, its extension for the GFO ontology (the *Situoid* theory), the *perdurantism* theory behind UFO and the *Endsley's* SA theory for human factors [72]. In particular, the *situoid* theory [184] deals with this issue by including the concept of *situoid*, a *perdurant* element that is the composition of snapshots (*situations*) in time. *Situoid* is a combination of infinitesimal slices, where each slice is a *situation* framed by its initial and final timestamps, i.e., a situation is a projection of a *situoid* on time boundaries. A *Situoid* is composed by situations that satisfy the configuration of a set of rules among events changing the contextual elements.

6.2.1 Improving Situation Representation in UFO

Here, we propose the adoption of some elements of these theories in UFO. We chose an analogous approach of the *Situoid* theory [180] as a background to accommodate the concepts of *situation*, *situoid*, *state of affairs*, *infon*, *comprehension*, *action* and *event* from GFO. UFO still lacks the formalization of *situoids* and their relations to temporal aspects, considering the definitions from [181, 182]. By harmonizing and axiomatizing those concepts in UFO, the context models represented with OntoUML can be enriched. To make it possible, we describe here the harmonization of these concepts w.r.t. elements already presented in UFO, as *event*, *participation*, *temporal relation*, *situation*, *context*, *context situation* (and its specializations) and *event type*. Furthermore, we reused predicates from the W3C *Time Ontology*, which are prefixed with *time*, following a similar approach of SSN/SOSA.

In UFO a situation represents a snapshot, an instantaneous “picture” of the structural aspects observed, which is a definition adopted in the three theories used here. Therefore, in accordance with [180, 182] we consider a *ufo:Situation* as an *ufo:Endurant*, in contrast with [148], which defines a situation as an occurrent, a similar concept of *ufo:Perdurant*. While *ufo:Situation* is an *ufo:Endurant*, *ufo:Situoid* is a *ufo:Perdurant*, since it has a temporal extension, its *time:hasBeginning* (when it is activated) is obtained from the *time:hasEnd* of the *ufo:SituationActivationEvent*. The *ufo:Situoid time:hasEnd* (when it is deactivated) is obtained from the *time:hasBeginning* of the *ufo:SituationDeactivationEvent* related to the *ufo:Situation*. Both situation activation and deactivation events are *ufo:EventType* and we adopt *ufo:EventType* with their causality relation to *ufo:SituationType*, illustrated in Figure 6.2. A *ufo:Situoid* can be a composition of other *ufo:Situoid*, i.e., it can carry information of other situations that are related in time [180, 181]. This mereological relation (composition) of *situoids* and the possible temporal relations between two situations are included. Table 6.1 describes the fundamental axioms selected from the *Situoid* theory to be adopted by UFO.

Table 6.1. Axioms from *Situoid* theory [180].

Axiom 5.9
$\forall x(\text{Situoid}(x) \rightarrow \exists t \exists s(\text{Topoid}(s) \wedge \text{Chronoid}(t) \wedge \text{top}(x, s) \wedge \text{prt}(x, t)))$
Axiom 5.12
$\forall s1 \forall s2(s1 \leq s2 \rightarrow (\forall \phi(s1 \models \phi \rightarrow s2 \models \phi)))$
Axiom 5.17
$\forall s(\text{Situoid}(s) \rightarrow \exists \phi(s \models \phi))$

Axiom 5.9 refers to the difference between *ufo:Situation* and *ufo:Situoid*, where a *ufo:Situoid* has temporal and spatial extent, framed by a *chronoid* (*time:Interval*) and a *topoid*. A *chronoid* is a time interval formalized through the *time:hasBeginning* and

the *time:hasEnd* object properties, where the range of these properties are *time:Instant*, which allows the representation of a specific date/time as *xsd:dateTimeStamp* through the *time:inXSDDateTimeStamp*. An inerent rule for this temporal representation is the *prt(x,t)* predicate, that means the situoid is framed by the interval (*chronoid*), i.e., the situoid is framed by the interval between *time:hasBeginning* and *time:hasEnd* and the date/time instantiated by *time:hasEnd* has to be greater than (after) the one instantiated by *time:hasBeginning*. The concept of *Topoid* is addressed here by adopting the *geo:SpatialThing* predicate from WGS ontology. Similar to SAREF4envi approach, the *geo:location* property is used to represent the *top(x,s)* predicate. This predicate was chosen because it reflects the relation between something and “a connected region of space”, which can be a point (latitude/longitude/altitude) or other geometrical thing in space.

Axiom 5.12 formalizes the ordering relation between two *ufo:Situoid*, where two *ufo:Situoid* are identical if they are part of each other and they have identical *chronoid* and *topoid*. In summary, if two *ufo:Situoid* are represented with exactly the same information, being framed by the same time interval the same location, then these *ufo:Situoid* are the same. Axiom 5.17 defines that *ufo:Situoid* is existentially dependent on *infons* because an empty *ufo:Situoid* actually cannot exist in reality and should be disallowed in the theory. Here, we align the concept of *infon* with *ufo:ContextualElement*, as described in the definition (Section 2.4).

Table 6.2. Axioms from UFO *perdurants* [182].

Axiom S1
$\forall s:\text{Situation}, e:\text{Event} \text{ triggers}(s,e) \rightarrow \text{obtainsIn}(s, \text{begin-point}(e))$
Axiom S2
$\forall s:\text{Situation}, e:\text{Event} \text{ brings-about}(e,s) \rightarrow \text{obtainsIn}(s, \text{end-point}(e))$
Axiom S3
$\forall e:\text{Event} \exists s:\text{Situation} \text{ triggers}(s,e)$
Axiom S4
$\forall e:\text{Event} \exists s:\text{Situation} \text{ brings-about}(e,s)$

These axioms adapted from the *Situoid* theory must be accommodated with the existing axioms from UFO-B, the *perdurants* part of UFO [182]. Table 6.2 lists some of those axioms, which describe world changes through the causality relations between *ufo:Event* and *ufo:Situation*. Axioms S1 and S3 refer to the relation *ufo:triggers* between a *ufo:Situation* and an *ufo:Event*. S3 establishes that every *ufo:Event* must be *triggered* by a *ufo:Situation*, and S1 establishes that a *ufo:Situation* obtains at the *time:hasBeginning* of the *ufo:Event* that it *triggers*. Axioms S2 and S4 define the relation *ufo:bringsAbout*. S4 establishes that every *ufo:Event* *ufo:bringsAbout* at least one *ufo:Situation*, and S2 establishes that all *ufo:Situation* obtain the *time:hasEnd* of the *ufo:Event*. The main harmonization that we suggest w.r.t. UFO-

B is the use of UFO-MLT regarding the representation of the causality relationship (*ufo:triggers*) between the power types of *ufo:Event* and *ufo:Situation*, i.e., between *ufo:EventType* and *ufo:SituationType*. The main reason for this change is to enable the representation of causality from a type of event (e.g., hard vehicle collision) to a type of situation (e.g., urgent traffic emergency). Notice that this relationship is further inherited by a relationship instance between *ufo:Event* and *ufo:Situation*.

When developing an EWS that relies on data events we need to differentiate an event observed in the real world and its representation as a digital data. For example, the increase of a body temperature of a patient is an event observed in real world, while the specific measurement collected by a thermal sensor attached to the patient is the representation of this event. As described before, SSN/SOSA represents this type of event with the *ssn:Stimulus* element, which is “an event in the real world that 'triggers' the Sensor”. Here we adopt the *ufo:SituationActivationTriggerEvent* as a *ufo:AtomicEvent* (observed in real word) that is evidence to creation of a *ufo:SituationActivationEvent* that activates a *ufo:Situation* in the software system. The same approach is used for the *ufo:SituationDeactivationTriggerEvent*, an event perceived from real world, e.g., body temperature decreases to 37, that evidences the *ufo:SituationDeactivationEvent*, i.e., the digital data collected by the sensor and sent to the software, which is responsible for deactivating the *ufo:Situation* instance (e.g., fever) in the software system.

Figure 6.2.
Harmonization
proposal of
Situations and
Events in UFO

As mentioned before, an *ufo:Event* has a *time:hasBeginning* and *time:hasEnd*. The *time:hasEnd* of an event is equal to the *time:hasBeginning* of the situation triggered by this event. For example, the situation of “Zika transmission in one person” is triggered by the Aedes mosquito bite (a *situation creation trigger event*), therefore the *begin time point* of this situation is equal to the *end time point* of the event. The temporal properties of the *event* are also responsible for the *current* property of a *situation*. When an *event triggers* a *situation*, the *situation* instance is created and the *current* property is set to true as default. When a *situation deactivation event* occurs, the *current* property of the situation is set to false and its *end time point* is set to the same value of the *end time point* of the *event*, i.e., the *situation* instance is deactivated and made immutable. This enables to design a *situation* composed of deactivated situations (historical data), such as “intermittent fever” situation, which is a situation of repeatable deactivated “fever” situations within a time period.

An *atomic event* depends on a *contextual element*, i.e., a structural object plays the role of a *participant* of an *event*. For example, a person participates in the event of “body temperature changing” while the “body temperature changing” existentially depends on a person. This definition is aligned with UFO-B in terms of the existential dependency between objects and events [183]. Similar to the properties of situation derived from *event*, the participants of a *situation* are derived from the contextual elements participating in the *events* related to the *situation* (the *creation* and *deactivation trigger events*). This property is transitive for composite situations, i.e., participants of a composite situation are derived from the participants of the situations within the composite situation.

This conceptualization addresses the requirement of representing causality relations between an *Event Type* and a *Situation Type*, which is related to the harmonization of *Situoid* in UFO and how CEP approaches the relations among events. Such solution was inspired in the *Event-Model-F* ontology [57], which upon the comparison of diverse event ontologies, such as the *Event Ontology*, *SAWA*, *EventML* and the *Situation Ontology*. The main difference between our approach and the *Event-Model-F* causality pattern is the explicit representation of the event causality observed in the real world (*creation and deactivation trigger events*) and its relation to the event represented as a digital data (*situation creation and deactivation event*). Moreover, we extend the causality relationship (*triggers*) between the powertypes of events and situations, i.e., between *Event Type* and *Situation Type*. These features represent important characteristics that are required both from the conceptual level and from the implementation level when using CEP engines for situation identification.

Besides the *Situoid* theory and UFO, behind the *SA theory* there are ontological assumptions not only related to the concept of situation awareness, but also to perception, comprehension, projection and decision making. This harmonization effort also considers the relationship reification approach from UFO [157], taking

into account the most basic types of relations (formal and material [268]), as well as the distinctions among relation, relationship, event and fact [157]. It also refers to the study of *truthmaker*, which is related to the definition of the *situoid activation event*, i.e., the event that makes a situation true.

6.2.2 SEMIoTICS Core Ontology

The *OntoEmerge* ontology, described in Chapter 2, represents the initial version of the well-founded core ontology developed according to the ODCM practice, using OntoUML. *OntoEmerge* can be extended for specific domain applications on disaster prevention, preparedness, response and recovery, as demonstrated in [159] for emergency plans generation. *OntoEmerge* is a DM core ontology, since it defines some of the most general concepts of the emergency field, such as emergency events, risks, resources, goals, intentions and plans. In addition, its background includes the official DM processes of England, Australia, USA and Spain. Moreover, its design followed the ontological engineering process from [269], inspired by results from risk management research [270]. The initial validation of *OntoEmerge* [159] identified that one of the most important concepts provided is the *Emergency Event Type*, which is triggered by *Hazardous Event Type(s)* and is used to classify the operation procedures to be performed and resources for the emergency response (reaction).

OntoEmerge was extended to cover the representation of an IoT EWS and its components. At first, an ontological analysis regarding ICT solutions for DRR was performed based on [16, 232, 271-274]. The starting point was the analysis of the technologies taxonomy [271] provided by the H2020 FARSEEING project¹²⁰, which focused on promoting health with support of IoT technologies (ICT devices). This taxonomy was provided through a website¹²¹ and was designed to characterize and classify technology-enabled interventions, being used for comparison studies of technologies. Specifically, we exploited the sub-domain *Technology Type* (D2), which classifies the technologies as *System* (SY), *Device* (DE), *Sensor* (SE) and *Actuator* (AC). Types of systems include personal emergency alarm, monitoring and positioning, and wireless sensor networks. Types of devices include electrocardiograph (ECG) and respiratory monitor. Types of sensors include heart rate monitor and accelerometer.

This classification does not clearly distinguish the ontological identity of each category. For example, sensors and actuators are devices by nature, according to both SSN/SOSA and SAREF conceptualizations. In particular, the SSN/SOSA classifies the elements that a *sosa:Platform* can host as *sosa:Sensor*, *sosa:Actuator*, *sosa:Sampler* and (another) *sosa:Platform*. In SAREF, as previously presented

¹²⁰ <http://farseeingresearch.eu/>

¹²¹ <http://taxonomy.farseeingresearch.eu/>

(mapping M02), the *saref:Device* can be equivalent to a *sosa:Platform*, representing the composition of sensors and actuators within the same physical object that implements a system. A *saref:Device* is classified according to its functions (*saref:FunctionRelated*), which includes *saref:Sensor* and *saref:Actuator*. Therefore, these alignments emerged:

- SE equivalent to *sosa:Sensor* (therefore *saref:Sensor*)
- AC equivalent to *sosa:Actuator* (therefore *saref:Actuator*)
- DE equivalent to a *sosa:Platform* (therefore *saref:Device*)
- SY equivalent to *ssn:System*

Although the FARSEEING taxonomy lacks ontological foundations, it was useful to provide a starting point for the extension of *OntoEmerge* to represent the notion of IoT EWS. The *Functionality* sub-domain (D3) was also analyzed, since it presents categories as *Alert*, *Monitoring*, *Communication* and “*Automatic or manual*”. The ontological analysis of the relationship between a *Technology Type* and a *Functionality* is illustrated in Figure 6.3 (on the bottom). In this conceptual model we describe an ICT Solution for DRR, e.g., an IoT EWS and an EMS (or EMIS), as a composition of one or more *ICT Solution Components*. An *ICT Solution Component* is a *mixin* concept, i.e., its instances can have different identity principles, and is anti-rigid, i.e., its instances can cease to be an *ICT Solution Component* but keep existing. The *ICT Solution Component* implements a *Technology Function*, which is the relator between the *Functionality Role* w.r.t a *Technology Role*.

For example, the *OntoEmerge* extension (SEMIoTICS core ontology), which is aligned to SSN/SOSA, allows the instantiation of an IoT EWS composed by a smartphone (device) for monitoring the cardiac behavior of a person, which forwards data from an ECG device, which is used for monitoring the electrical potential difference among electrodes attached to the person’s chest, to a server (device) in the cloud, which is used for assessing the data to identify abnormal situations and issue alerts.

The alignment that enables the link of this ontology to *OntoEmerge* is the relationship between the *Monitoring Hazardous Event*, a specialization of *Monitoring*, that relates the IoT EWS element to the *Hazardous Event Type* (original from *OntoEmerge*). This conceptualization covers the monitoring aspect of an IoT EWS but lacks on the foundations regarding the relations among *Situation*, *Event*, *Situation Types* and *Event Type*, partially addressed in the harmonization of *Situoid* in UFO.

Another basic aspect that our core ontology covers is *Alerting (Functionality)*. We also applied the ontology alignment approach, this time for the EDXL-CAP standard. Although the EDXL-CAP semantic model produced by the RESCUER project is only a direct transformation from the lexicon standard to RDF

(“triplication” of EDXL-CAP meta-model), it was useful to align the SEMIoTICS core ontology with this standard, which ultimately improves the semantic interoperability of the IoT EWS with systems that use EDXL-CAP. Besides this semantic model, the BFO-based ontological analysis supported us on modelling the *edxl_cap:AlertMessage* as a *Message* that provides the *Alerting Functionality*. In BFO a *Message* is an *Information Bearing Entity*, which is an *Independent Continuant* (made of physical matter, equivalent to *Endurant* from UFO) and is similar to the *Physical Object* category of UFO-C. Therefore, *edxl_cap:AlertMessage* is classified as a specialization of a *Physical Object* that is *rigid*, provides an identity principle to its instances and is a functional complex, i.e., a *Kind*.

Figure 6.3.
OntoEmerge
extension for
technology types

These alignments to SSN/SOSA and EDXL-CAP using ontological analysis as a foundational basis is the approach recommended by SEMIoTICS to address specific requirements of an IoT EWS. For terms that are not presented in existing semantic models, we recommend the use of the consolidated *DM metamodel* (DMM) of [83], considering the ontological analysis approach. DMM is not aligned with the *SA theory*, but the definitions from DMM supported the ontological commitments behind SEMIoTICS core ontology. For example, the definition of disaster can be

considered a *situoid* that is *brought* by an *Emergency Event*, which is *triggered* by the *Hazardous Event Type* that causes the disruption in society resulting in a widespread destruction. This characterization coincides with disaster as “a crisis situation that far exceeds the capabilities” [275].

The main idea here is to present a balanced approach that mixes multiple-alignments of existing standards with the ODCM practice. In this way, some predicates can be either reused or created in the ontology. This approach is demonstrated in section 7 when validating SEMIoTICS through the development of an IoT EWS that required extensions to support the cardiac domain (e-Health) and to support traffic accidents domain. Finally, by using our semantic MDE transformation approach, the core ontology elements can be automatically coded.

6.3 Adapting SML for IoT EWS

An EWS must provide temporal reasoning over the events being monitored [247], which relies on temporal relations to correlate events. For example, the situation of a possible contagion of Zika is characterized by the situation of high risk of Zika infection overlapping the situation of a person diagnosed with Zika within the same geographical area. “Overlapping” is a temporal relation between the two situations, usually referred as an operator from Allen’s calculus for temporal reasoning.

Furthermore, an EWS must implement temporal existential rules, i.e., rules that only match events occurring in a specific time window (ranges of time units). For example, the case of influenza-like illness (ILI) is defined by WHO as “an acute respiratory infection with: measured fever $\geq 38\text{ }^{\circ}\text{C}$ and cough; with onset within the last 10 days”. Therefore, ILI is characterized by the existence of fever and coughing within the past 10 days. Sliding time windows allows the definition of existential rules for EWS [248]. An EWS for public health surveillance also needs to allow the definition of descriptive multivariate and aggregation functions, for example to describe average epidemic levels and sums in populations [249].

6.3.1 The Need of SML Redesign for SEMIoTICS

Originally, the semantic formalization of SML considered a mapping from the syntactic elements of the language to a logic framework (frame-based models). This framework was chosen because frame-based models deal with the temporal aspects discussed above, having frames and their properties as a modelling primitive. Two essential (disjoint, complete) categories of STs were defined: (i) *Simple ST* is interpreted as an open sentence formula which includes free variables (e.g., parameters), the situation itself and the variables representing the contextual elements (characterized by predicates); and (ii) *Complex ST* has the same

characteristics of a simple situation but it considers new predicates for each situation that is part of the composition of the complex (whole) situation, i.e., it allows the use of situation participant.

There are two ways to compose a *complex ST* with situation participants (references to other STs), i.e., relate the participants within the whole ST. (1) the use of *equals* relation between entity (or relator) participants of the ST referred by the situation participant, and (2) the use of temporal (Allen) relations between situation participants.

One important aspect of SML is the management of the *situation lifecycle*, i.e., the creation (activation) and the deactivation of an instance of a ST. One or more events in the context characterize the creation of an instance of a ST. For example, the event of increase over 37 degrees of a person's body temperature triggers the ST of fever, i.e., characterizes the creation of an instance of fever ST. This instance is said to be active (or current) while the constraints hold; and is said to be deactivated (past) when the constraints are not satisfied anymore. A fever ST instance holds in time while the body temperature of a person is greater than 37 degrees, but it is deactivated when the temperature measured over the time stops to satisfy the constraint. An instance is never reactivated, e.g., if a fever instance is deactivated because the temperature measured changed from 38 to 37 and right after the temperature measured is 38, a new instance of fever is created, rather than reactivating the deactivated instance. The temporal relations between STs depend on the status of each ST participant (current or past).

Although the initial version of SML (1.0) covered temporal reasoning and existential rules, it did not possess the necessary expressiveness. SML 1.0 did not consider foundational aspects, as the explicit distinction of context, situation and event, as well as their association (including causal) relationships. These distinctions are necessary to properly formalize the measures of association and causality. [174] discusses the verification and validation of SML with OntoUML, but SML 1.0 still fails to represent, for example, the mutability and cardinality of participants, relationships and self-reference of STs, functions and aggregation functions of collections. These elements are also required for the definition of the measures of frequency within epidemiological studies. These limitations have inspired our SML redesign. The SML redesign is discussed in terms of semantics (meaning of the elements) and its metamodel, as suggested in [133].

6.3.2 Revisiting the SML Metamodel

The abstract syntax of our SML redesign is described here in terms of its metamodel by discussing the improvements necessary to address the requirements of the epidemiology scenario. We refer to this new SML version as 2.0.

OntoUML improves the expressiveness of the context models. For example, OntoUML allows the dynamic classification of entities through modality types, one of the main benefits of an ontological language [177]. This is achieved by the clear distinction of rigidity: the instance must instantiate a class while it exists, and non-rigid concepts: the instance can instantiate a class contingently. Table 6.3 summarizes the mappings between the SML and OntoUML metamodels. In OntoUML a *Quality* is related to a *substantial*, while a *Property* is an attribute of a *substantial* (as a class property). Intrinsic context in the former SML was used to represent both *qualities*, i.e., properties that can be directly evaluated (e.g., temperature and location), and *modes*, i.e., properties that cannot be directly evaluated (measured) in terms of a single space (e.g., a person's headache). Since OntoUML provides this distinction, we absorb it by mapping the *AttributeReference* (SML) to *Quality* or *Property* (OntoUML), and by introducing *ModeReference* (SML), mapping to *Mode* of OntoUML. In addition, a relational context in SML is derived from the notion of *Relator* in OntoUML, which is a moment representing objectifications of relational properties.

Table 6.3. Mappings from OntoUML to SML metaclasses

SML	OntoUML
ContextModel	Model
AttributeReference	Quality/Property
ModeReference	Mode
QualityLiteral	ReferenceStructure/DataType
TypeLiteral	SubstantialClass
EntityParticipant	SubstantialClass
RelatorParticipant	Relator
ContextFormalLink	FormalAssociation
CharacterizationLink	Characterization
MediationLink	Mediation

SituationTypeComplex specializes *SituationType* and allows the use of *SituationParticipant* in its diagram, which is a reference for a *SituationType* and has properties *isCurrent*, *beginTime* and *endTime*. The property *isCurrent* reflects the *TemporalKind* enumerator, characterizing whether the *SituationType* is present or past. In addition, *SituationParticipant* has the ability of exposing each of the *EntityParticipant* that composes the referenced *SituationType*. A *SituationTypeComplex* also allows the use of each specialization of *AllenLink* (e.g., before, during, overlaps) to relate two different *SituationParticipant*. A *SituationTypeSimple* also specializes *SituationType*, but does not allow the use of *SituationParticipant* neither *AllenLink*.

A *SituationTypeElement* can be a *ReferableElement*, a *Literal* or a *SituationTypeAssociation*. A *ReferableElement* can be a *ModeReference*, an *AttributeReference*, a *Function* or a *Participant*. *ModeReference* and *AttributeReference* are

derived from OntoUML (mode and quality, respectively), while *Function* represents a relation between elements not defined in the context model for some reason (e.g., scope limitation). A *Function* can be a calculus or derivation function, i.e., a user-defined operation, between one or more *AttributeReference* and *QualityLiteral*, which play the role of parameters of the function. A *Participant* can be a *SituationParticipant*, an *EntityParticipant* or a *RelatorParticipant*. A *Literal* can be a *QualityLiteral* or a *TypeLiteral*, derived from OntoUML. A *QualityLiteral* can be used only when related through an *OrderedComparativeLink* or *EqualsLink*. A *TypeLiteral* can be used only when related through an *InstanceOf* to an *EntityParticipant*.

Figure 6.4. Main elements of SML 2.0 metamodel

Mutability was addressed in SML 1.0 to specify whether exists (or not) the occurrence of a *SituationParticipant* element in a ST with the *exists* logical quantifier, representing the “at least one” instance rule. In SML 2.0 we extended this capability to *EntityParticipant* and *RelatorParticipant* so it is possible to specify if an entity (or relator) exists in a ST. This is defined by the meta-attribute *immutable* in the metaclass *Participant*. Mutability is defined apart from multiplicity (through cardinality) because the goal of cardinality is to allow the specification of minimum and maximum numbers of instances. *Participant* cardinality is addressed through the meta-properties *max* and *min* ([1..1] as default) in the metaclass *Participant*. With mutability it is possible to specify an ILI situation for example, i.e., the model

of the rule “if exists fever and coughing within the past 10 days”. With cardinality it is possible to specify an intermittent fever. SML also allows the specification of descriptive multivariate functions, i.e., user-defined operations, through the *Function* metaclass. Examples of these functions are sum, square, division or even more complex functions, as multivariate polynomial integrals and derivations. Figure 6.4 illustrates the main elements of the SML 2.0 discussed here.

SituationTypeAssociation represents all possible relationships in a SML model. To create a relationship in MDG technology a stereotype needs to extend a metaclass named of *Association*, having the properties of the relation described as attributes of this *metaclass*. For example, the style of the line is described by the attribute *_lineStyle*, source and target multiplicities are set by *_SourceMultiplicity* and *_TargetMultiplicity*, respectively. Direction, reflexivity, symmetry, cyclicity and transitivity are defined through attributes too.

In OntoUML a *Quality*, *Property* and *Mode* are dependent on a relationship to a substantial (e.g., a *Kind* or a *Role*). This substantial is mapped to SML as an *EntityParticipant* (or a *RelatorParticipant*) and an *AttributeLink* relationship is created to link them. Besides *AttributeLink*, SML presents other two relationships that are derived from OntoUML: (i) *CharacterizationLink*, which reflects the characterization relationship between a mode and an entity (links a *ModeReference* to an *EntityParticipant*); and (ii) *MediationLink*, which reflects the mediation relationship between a relator and an entity (links a *RelatorParticipant* to an *EntityParticipant*).

SML 2.0 also allows the creation of comparative relationships through (i) domain-specific formal (as OntoUML formal relationship), (ii) primitive (equals, greater than, less than) and (iii) Allen (before, overlaps, meets, etc.) relationships. This is addressed by, respectively: (i) the *ContextFormalLink*, (ii) the *EqualsLink* and the specializations of the *OrderedComparativeLink* and (iii) the specializations of the *AllenLink*. An example of using a primitive relationship to specify the rule of the fever ST is: person (*EntityParticipant*) body temperature (*AttributeReference*) greater than (*OrderedComparativeLinkGreaterThan*) 37 degrees (*QualityLiteral*). The main difference to SML 1.0 is that each Allen relator was introduced as a metaclass specialization of *AllenLink*, improving the expressiveness of SML.

Regarding the composition of STs, an issue of SML 1.0 was to limit the relation of *EntityParticipant* (or *RelatorParticipant*) within a *SituationParticipant* to *EntityParticipant* (or *RelatorParticipant*) within the complex ST only with the *equals* relationship, which had the side-effect that the *EntityParticipant* (or *RelatorParticipant*) should be bounded by the temporal space in which the complex ST occurs. To address this limitation, SML 2.0 allows exposing not only the *EntityParticipant* (or *RelatorParticipant*) of the *SituationParticipant*, but also the attributes (*AttributeReference*) of these participants. Therefore, in addition to *equals*

relation from a participant of a ST, the new metamodel allows to use of *AttributeLinks* of a participant to be exposed in a *SituationParticipant*.

Dynamic classification was partially covered by SML 1.0, by using past situations to indicate whether a situation participant was an instance of some type in the past. This is a limited solution because it is not possible to address the cases in which a participant is no longer an instance of this same type in the present. For example, a ST of “healed patient” is composed by a past situation of “has any ongoing treatment”, which exposes the reference to the patient instance, used to link to the person entity (through a comparative relation *equal*). This model allows an instance of a person that is not being treated anymore but can still be a patient, which is an unwanted instance. To address this issue, we introduced the *instance of* relationship in SML, indicating if contextual element is *instance of* a type or not. In the example scenario, the negation of this relation can be applied to the person entity, characterizing that in the current situation the person is not an instance of patient anymore. This approach also solves the problem of representing “past specialization”, a common issue in conceptual modelling.

Another issue of SML 1.0 regards that specific formal relations were needed to specify that a past situation (of the same ST) occurred at some point in the past. For example, in [276] the formal relation “within the past” had to be created in the example of *AccountUnderObservation* ST, embedding the semantics of the Allen relation *before*, which is unwanted since it is considered a workaround. To address this problem, the *SelfParticipant* was added as a specialization of *SituationParticipant* with *temporality* set to *current*, allowing the designer to use *AllenLinks* to relate a ST reference to the ST being composed (the “self”, which is always the current situation).

Regarding the use of *Function*, the function parameters (input(s)) are represented by the *FunctionParameterLink*, which is a relationship from an *AttributeReference* or a *Literal* to the *Function*. The *FunctionResultLink* is the relationship used to specify the output(s) of a *Function*. *OrderedComparativeLinks* and *EqualsLink* can be used from a *Function* when the *Function* has only one output and the designer wants to specify a comparative relation with this output. For example, the ST “High Body Mass Index (BMI)” uses the “square” *Function* with a person’s height as input, producing the squared height (as output), which is used as input, along with the person’s weight, for the function “division”. The greater than relationship is used from the output of “division” to the literal “25” (if it is greater than 25, it characterizes a high BMI).

6.3.3 Example Scenario: EWS for Zika Epidemics

Consider as an example an epidemiological surveillance EWS for Zika outbreaks. A particular application may be responsible for receiving data from HBM sensors

attached to patients, providing information of body temperature, heart frequency and Zika symptoms (skin rashes, conjunctivitis, pain, malaise and headache). Another application may be responsible for collecting data from medical records, including whether a patient was bitten by mosquitos, and last lab exams taken. A third application may be a patient tracking system responsible in collecting data from patients being transported in ambulances. A fourth application may be responsible in collecting web-based data and the location of *Aedes Aegypti* infected regions (e.g., HealthMap). The integration of all these applications, that play the role of data sources, is supported by the SEMIoTICS framework. Notice that the concept of application here is broad so that an application can even be another integrated EWS, which may have several actors involved.

Figure 6.5.
“Possible Zika
contagion”
designed with
SML

In this scenario, we created situation models with SML 2.0 to exemplify the situation types specification that the IoT EWS must provide. Figure 6.5 illustrates the “possible Zika contagion” ST (ST1). On top, the “High risk Zika infection” ST is based on [246], relating three STs in the same geographical region (within 50 km): (ST2) “adequate conditions for *Aedes* proliferation”, dependent on temperature, precipitation and hydrology of the location; (ST3) “*Aedes* mosquitos reported in the past year”; and (ST4) “Person travelling from infected area”. On bottom, “Person diagnosed with Zika” (ST5) is activated if there is a positive result of specific exams for a person. The temporal relation characterizes that ST1 *overlaps* ST5, while the location of ST1 must be near the location of the infected person from ST5.

SML graphical notation was designed according to the easiness of assimilation and comprehension of the chosen symbols [277]. By following the main graphical elements of SML 2.0, the visual cognitive principles used in this exemple inherits:

- (1) semiotic clarity: correspondence between a semantic constructor and a graphical symbol achieved by the one-to-one mapping from the concrete syntax classes to the graphical elements; (

- (2) perceptual discriminability: the easiness and precision in which graphical symbols can be different to each other is achieved by distinct textures, forms, Colors and brightness of the graphical elements of SML. For example, a *SituationParticipant* is a diagram reference of a ST, when having the tagged value *isCurrent* set to true the label background is greyed, otherwise white;
- (3) visual expressivity: the number of visual variables among the elements is addressed by distributing the use of these variables among the graphical vocabulary to optimize the cognitive load required for interpretation. The symbols in SML are differentiated by two variables at least. For example, *ModeReference* and *TypeLiteral* are distinct by Color and shape;
- (4) graphic economy: the control of the number of distinct graphical symbols is addressed by a balanced approach between adding new symbols or only differentiate them by minor details, such as textual differentiation. Although using text is not considered a good practice for the perceptual discriminability, this choice was made for the existential composition of situations, to avoid the increase of the language complexity. When the *immutable* attribute of the specializations of *Participant* are set to false, the existential symbol ($\exists$) is added before the name of the Participant. Negation ($\exists!$) is also possible with *negationMutable*.

6.4 Concluding Remarks

In this chapter we introduced the main mappings between SSN/SOSA and SAREF to enable the semantic integration of IoT platforms. The mappings presented here focus on translating elements about sensor, device, its composition, functions and observations. Here we detailed five mappings to cover these properties, exemplifying how to produce a SAREF ontology from a SSN/SOSA ontology through our semantic translation mechanism, used by the *Input Handler* component.

Lessons learned include the analysis of these ontologies and how their main differences are merely a matter of distinct points-of-view of the same conceptualization. For example, a similar structure is used to represent sensor observations: while SSN/SOSA provides the *Observation* class as a central element, SAREF provides the *Measurement* element. In addition, we think that SAREF seems to be easier to understand when compared to SSN/SOSA, which is a characteristic sought by the SAREF task force when creating the ontology along with industry partners. Although, our focus was to address the main IoT elements, several concepts were not mapped, such as between *saref:Actuator* and *sosa:Actuator*.

Therefore, further research is needed to provide a more complete mapping between these ontologies, at least covering the required elements of each ontology.

In this chapter we also stressed the foundational aspects of the *Context Data Management* component. We discussed the challenges of developing a well-founded DM core ontology, as well as the requirements for including situation foundations in UFO. To address these requirements, we presented a harmonization proposal that includes the *SA theory* and the *Situoid theory*. We learned that deciding whether a concept should be placed, in the foundational layer or in the core layer is a difficult task. We advocate that *situation awareness* should be accommodated in UFO, along with *situation* and *situoid* concepts. Therefore, the notions of *situation* and *situoid* are part of the foundational ontology and not part of the core ontology level, as in [278]. In contrast, *victim*, *aid distribution* and *rescue* seem to be better suited for the core ontology. However, this classification is not trivial since the boundaries between a foundational and a core ontology are fuzzy and are considered as an open research topic [279]. To guide this choice, one must keep in mind whether a concept pertains to a more general (foundational) classification or to a specific field.

This chapter showed evidences that adapting a mature foundational ontology with high-level concepts related to situation is not a trivial task, requiring relevant research effort to combine existing theories through well-founded philosophical interpretations. Therefore, from the DM field, we argued upon the need of theoretical foundations for modelling language constructs to support the IoT EWSs specification. This was demonstrated by extending an existing well-founded emergency plan ontology, the *OntoEmerge*, towards the SEMIoTICS core ontology. *OntoEmerge* differs from ontologies, such as EMERGEL, because it inherits assumptions from UFO (through OntoUML), which is semantically richer. Nevertheless, *OntoEmerge* lacks a set of terms, mainly from the response and recovery phases, such as coordination, resilience and reconstruction. Therefore, some terms were incorporated in the core ontology through a mixed approach of ODCM with multiple-alignments, such as the one presented in [280]. SEMIoTICS core ontology was aligned to SSN/SOSA and to EDXL-CAP.

Finally, in this chapter we cover the *Situation Identification Management* component through the situation modelling approach that integrates the context model (the DM core ontology) with the complex rules that represent situation types. We analyzed the gaps of a specification-based language for situation detection, the first version of SML, regarding both the IoT EWS requirements and the impact of the *Situoid* harmonization in UFO along with the relations to the ontological language OntoUML. The SML 2.0 was produced, exemplifying its use for the modelling of an epidemiological surveillance EWS for Zika outbreaks.

The concrete syntax of SML indicates that the improvement of common understanding of rules can be achieved by the higher expressivity of the language

supported by the graphical symbols. Temporal reasoning specification is addressed by Allen's operators among STs. Temporal existential rule is addressed by mutability in SML (the existential symbol). It is clear that, although we used a CEP technology to support the SML redesign (SCENE), SML 2.0 is independent of any implementation technologies and is independent of a specific domain. SML's semantic and syntax address the **Req7** and **Req8**, which reflect CEP requirements, without referring exactly to them, therefore being general.

Case Study: The Valencia Port IoT EWS

The purpose of this Chapter is to demonstrate how the SEMIoTICS framework fulfils the functional requirements, illustrating its use through a case study. Therefore, this Chapter describes the functional validation of SEMIoTICS. This validation was performed in a case study of an IoT EWS that detects traffic accidents for the Port of Valencia. This case study was conducted in collaboration with the H2020 INTER-IoT project. Section 7.1 describes the requirements and use cases that the IoT EWS should address, such as integrating logistics and health data for accident detection based on monitoring trucks at the port area and the vital signs (cardiac behavior) of the drivers. Section 7.2 describes the INTER-IoT-EWS solution architecture, which adopts specific technologies that are guided by the SEMIoTICS framework, including an IoT platform and ontologies. It also describes how each component of INTER-IoT-EWS was developed with support of the SEMIoTICS methodology. Section 7.3 presents the INTER-IoT-EWS functional validation, describing the test cases executed in the laboratory environment and at the port area, and analyzes the results. Section 7.4 concludes this chapter with our main achievements, lessons learned and remaining open issues.

7.1 INTER-IoT: Detecting Accidents with Trucks

The SEMIoTICS framework was validated with a case study performed within the H2020 INTER-IoT project [281], which considers a semantic interoperable IoT EWS that is able to detect truck accident risks at the Valencia port area. For demonstration and validation purposes, INTER-IoT described a scenario to decrease the risk of fatal accidents at the port of Valencia, improving health prevention and enabling quick reaction by reducing time response [66]. This scenario is part of the INTER-DOMAIN use cases, which considers a variety of data sources from different domains and processing requirements for safety-critical applications, applying existing domain-specific ontologies and standards along with

their serialization formats. Accident risks are assessed by monitoring the truck drivers' vital signs with electrocardiogram (ECG) medical wearables, and the trucks' position with speed and accelerometer data, using the drivers' mobile phones as gateway. The detection targets health issues of drivers and collisions of trucks with dangerous goods. The IoT EWS is able to provide personalized emergency notifications to multiple target groups with distinct information requirements.

7.1.1 Requirements

The requirements elicitation for the emergency scenario under INTER-DOMAIN use case was performed by the INTER-IoT consortium, which was reported through two project deliverables, the requirements analysis (D2.3) [282] and the use cases manual (D2.4) [66]. The requirements from INTER-DOMAIN are mapped to the specific requirements that should be addressed by the IoT EWS, as follows.

(IIOT-FR1) Ability of IoT platforms to coordinate with emergency systems (id.251) and development of new services for emergency management (id.248) [282]. This requirement leads to the need of developing an IoT EWS with an IoT platform, where the chosen IoT platform should be able to integrate with emergency systems, enabling the detection of accidents and risks with trucks within the port area. The IoT EWS should be able to identify vehicle collisions and severe changes of the driver's cardiac behavior, alerting their urgency and severity to multiple targets through emergency interoperability standard(s).

- Acceptance criteria: verify if the IoT EWS was built on top of an IoT platform and the IoT EWS is able to interoperate with emergency systems by means of emergency interoperability standard(s).

(IIOT-FR2) Real-time support (id.20), allow exchanging complex medical measures (id.102) and definition of reference meaning for health information (id.106). The integration of the involved IoT platforms (haulier and port IoT) should be able to provide real-time sensing about health information of the driver, monitored through a medical device such as an electrocardiography (ECG) device that provides complex high-frequency ECG time series. ECG monitoring is necessary because cardiovascular emergencies are life-threatening disorders that must be recognized as soon as possible to minimize morbidity and mortality. The IoT EWS should be able to provide both raw and processed data, e.g., ECG sequence (time series) and heart rate (HR). These data need to be precisely described and integrated such that the port emergency control

system can consume them in conformance with emergency interoperability standard(s).

- Acceptance criteria: verify if the IoT EWS, at the application level, is able to process the health data to identify cardiac issues and is able to send data to emergency systems based on emergency interoperability standard(s).

(IIOT-NFR1) Provide syntactic and semantic interoperability among IoT artefacts (ids. 180 and 249) using device semantic definition (id.23). IoT platforms should be semantically and syntactically interoperable. The solution should be able to integrate the involved IoT artefact(s) in a way that their data syntax and semantics are understandable, i.e., can achieve common understanding among the participating parts. In particular, this integration should be performed by using the IPSM mechanism for semantic translations with the main IoT semantic standards, i.e., SAREF and SSN/SOSA.

- Acceptance criteria: verify if the IoT EWS can apply the IPSM mechanism to translate data syntax (lexicon) and semantics of exchanged messages, comparing it to other translation approach.

(IIOT-NFR2) Provide interoperability between things from different domains (id.76). E-health and logistics should be integrated at the INTER-IoT application and semantics level, including primitives for data interpretation of medical and logistics data through e-health and logistics ontologies.

- Acceptance criteria: verify if the use of semantic models to represent e-health and logistics data within the IoT use cases (listed below).

(IIOT-NFR3) The device(s) battery power should be able to be monitored (id. 149). The energy consumption (battery level) of the devices being used for the situation identification mechanism should be monitored.

- Acceptance criteria: verify if the IoT EWS is able to collect and process the energy consumption data, and if the EWS provides these data for the application level.

7.1.1 Use Cases and Situation Types

Four use cases were defined to illustrate how these requirements are addressed. In this thesis, a use case is a composition of events defining the interactions between the system and the roles played by actors, such as the participation of *substantials* in a *complex event* in UFO. Each use case is described here in terms of *situation types*.

(IIOT-UC01) Vehicle collision detection

The EWS monitors the truck's instantaneous acceleration and location, being able to monitor the truck behavior within the port area. The data are provided by both the mobile phone and by the ECG device, which enable the detection of possible vehicle collisions. According to the patent for “vehicle security with accident notification and embedded driver analytics” (US 9491420 B2) [283], a large change of the vehicle speed over a very short period of time, i.e., an increased instantaneous acceleration, is hard to attain when the vehicle is not controlled by a human. Therefore, any high acceleration instance occurs due to a collision, which simplifies the mechanism to detect accidents.

The common collision detection approaches use tri-axial (x, y, z) accelerometer sensor(s) deployed within the vehicle to collect data about the vehicle's acceleration, allowing to compute the *G-Force* over the sensor in each instant. The detection mechanism compares if the *G-Force* computed is above a certain threshold to detect a collision, which is usually 3G for devices deployed in the vehicle chassis [284-287]. Figure 7.1 (left, upper) illustrates the car acceleration axes and (right, upper) the approach for accident detection based on mobile accelerometer [288]. The collision detection is supported by computing the instantaneous cross-axial energy function (Equation 7.1), as in the *Fall-MobileGuard* solution [289], which provides a similar approach for fall detection, and the accident alert system [290], which uses a microcontroller running collision detection algorithm as a warning system attached into the car chassis.

(Equation 7.1)

$$\begin{aligned} \text{CrossAxialEnergy} &= x^2 + y^2 + z'^2 \\ z' &= z - 1G \end{aligned}$$

Where x, y, z are axial accelerations measured in m/s^2 . $1G = 9.806 \text{ m/s}^2$. Basic rule: $(\text{ComputeCrossAxialEnergy}(\text{accelerometerDevice}) > \text{Threshold_VehicleCollision})$.

Figure 7.1 (left, bottom) illustrates the acceleration measured by each device participating in the use case, i.e., the mobile phone and the ECG device, for each axis, while Figure 7.1 (right, bottom) illustrates the cross-axial energy computed. The peak at the end of the recording represents a possible collision. The classification of severity and urgency of the collision is based on a deterministic distribution, according to four ranges, described in Table 7.1. Variable A represents the cross-axial value computed when the accident was detected, while variable B represents the threshold ($\text{Threshold_VehicleCollision}$). For example, if the cross-axial value computed is within 20% above the threshold, the IoT EWS should issue the warning informing that the severity is *Minor* and the urgency is *Expected*. The severity and urgency values follow the enumeration provided by EDXL-CAP in *Info* element (see Chapter 4), where urgency varies among minor, moderate,

severe and extreme, while severity can be set as expected or immediate. The ranges were calculated according to the thresholds used by the US 9491420 B2 patent.

Figure 7.1. (a) Vehicle acceleration axes; (b) accident detection with mobile accelerometers; (c) Acceleration series; (d) Cross-axial with collision detected

The situation types defined for this use case are:

- (UC01.ST01) Collision detected based one acceleration observation from the ECG device with cross-axial function and identification computed at the field gateway level (mobile app).
- (UC01.ST02) Collision detected based on one acceleration observation from the ECG device with cross-axial function and identification computed at the cloud level.
- (UC01.ST03) Collision detected based on one acceleration observation from the mobile device (smartphone) with cross-axial function and identification computed at the field gateway level (mobile app).

(UC01.ST04) Collision detected based on one acceleration observation from the mobile device (smartphone) with cross-axial function and identification computed at the cloud level.

(UC01.ST05) Collision detected based on acceleration observations from the ECG device and the mobile device (smartphone) with cross-axial functions and identification computed at the cloud level.

Table 7.1. Ranges of emergency severity and urgency for collisions

Range	Urgency	Severity	Color
$B < A \leq B * 1.2$	Minor	Expected	Green
$B * 1.2 < A \leq B * 1.4$	Moderate	Immediate	Yellow
$B * 1.4 < A \leq B * 1.6$	Severe	Immediate	Red
$B * 1.6 < A$	Extreme	Immediate	Black

(IIOT-UC02) Cardiovascular risk detection

The IoT EWS is able to detect occurrences of arrhythmia in truck drivers, i.e., bradycardia and tachycardia, issuing alerts according to the possible cardiovascular emergencies. By allowing the IoT EWS to detect cardiovascular emergencies with trucks' drivers, the risk of accidents at the port area can be reduced. The IoT EWS monitors the cardiac behavior of the driver, tracking the heart rate during the trip inside the port area. The heart rate data are provided by the ECG device as a processed measurement from a function, which measures the intervals between R waves from the QRS complex of the ECG time series. Both processed heart rate and raw data ECG data are made available to the EWS, which must adhere to health data privacy best practices, complying to the EU General Data Protection Regulation (GDPR)¹²².

Table 7.2. Classification of vital signs levels with Early Warning Scores

Score	3	2	1	0	1	2
Respiratory rate (beats/min)	>35	31-35	21-30	9-20	-	-
SpO2 (%)	<85	85-89	90-92	>92	-	-
Temperature (C)	-	>38.9	38-38.9	36-37.9	35-35.9	34-34.9
Systolic BP (mmHg)	-	>199	-	100-199	80-99	70-79
Heart rate (bpm)	>129	110-129	100-109	50-99	40-49	30-39
AVPU	-	-	-	Alert	Verbal	Pain

¹²² <https://eugdpr.org/>

Thresholds used by the detection mechanism have been defined based on existing classifications to detect health risks, both regarding bradycardia and tachycardia¹²³. In particular, the approach based on *Early Warning Scores* [291] should be considered. For example, Table 7.2 illustrates the rules underlying each vital sign scoring function and set thresholds for both arrhythmia types, i.e., below 50 bpm for bradycardia and above 99 bpm for tachycardia. The heart rate score reflects the urgency and severity levels of the situation, e.g., for heart rate of 115 bpm, the score is 2. This information allows one to quantify the person's health level, where the score greater than 5 represents that the person requires intensive care and is in a near-death situation.

The classification of severity and urgency of the arrhythmia detected is based on similar deterministic distribution of UC01, illustrated in Table 7.3. Variable *A* represents the heart rate computed (in bpm) when the accident was detected, while variable *B* represents the threshold (*Threshold_Bradycardia* and *Threshold_Tachycardia*). For example, if the heart rate computed is within 10% and 20% above the tachycardia threshold, the IoT EWS should issue the warning informing that the severity is *moderate* and the urgency is *immediate*. The ranges are defined differently here because UC01 used a patent as reference, while UC02 used the early warning score approach.

- *Threshold_Bradycardia* = 40 bpm. *Threshold_Tachycardia* = 99 bpm
- Basic rule for bradycardia: $(\text{ComputeBPM}(\text{ecgECGdevice}) < \text{Threshold_Bradycardia})$
- Basic rule for tachycardia: $(\text{ComputeBPM}(\text{ecgECGdevice}) > \text{Threshold_Tachycardia})$

Table 7.3. Ranges of emergency severity and urgency for arrhythmias

Range	Urgency	Severity	Color
$B < A \leq B * 1.1$	Minor	Expected	Green
$B * 1.1 < A \leq B * 1.2$	Moderate	Immediate	Yellow
$B * 1.2 < A \leq B * 1.3$	Severe	Immediate	Red
$B * 1.3 < A$	Extreme	Immediate	Black

The situation types defined for this use case are:

(UC02.ST01) Bradycardia, heart rate computed by mobile app and identified by EWS (cloud): based on heart rate observation computed at the field gateway level and identification computed at the cloud level.

¹²³<http://www.mayoclinic.org/diseases-conditions/bradycardia/diagnosis-treatment/diagnosis/dxc-20321665>

<http://www.mayoclinic.org/diseases-conditions/tachycardia/diagnosis-treatment/diagnosis/dxc-20253919>

(UC02.ST02) Tachycardia, heart rate computed by mobile app and identified by EWS (cloud): based on heart rate observation computed at the field gateway level and identification computed at the cloud level.

(UC02.ST03) Multiple occurrences of bradycardia, heart rate computed by mobile app and identified by EWS (cloud): based on a pattern of multiple consecutive occurrences of UC02.ST01 within a pre-defined time interval. Figure 7.2 illustrates this complex pattern, where the situation type is activated when three consecutive bradycardia (UC02.ST02) are detected within an interval of 20 seconds.

Figure 7.2. Complex situation detected from consecutive bradycardia

(UC02.ST04) Multiple occurrences of tachycardia, heart rate computed by mobile app and identified by EWS (cloud): similar to UC02.ST03 but referring to consecutive occurrences of UC02.ST02.

(IIOT-UC03) Correlation of vehicle collision and cardiovascular risks

This use case exploits the possible temporal relations between UC01 and UC02 for detecting accidents that involve both collision and cardiovascular risk detection. This approach identifies that a collision is followed by an arrhythmia, which can represent a possible injury of the driver after the accident occurrence, as well as the other way around, i.e., an arrhythmia followed by a collision, which can represent that the accident occurred because of a health issue. For example, if a truck collision is detected from the mobile accelerometer data (UC01_ST03) and within 20 seconds a bradycardia is detected (UC02.ST01), then there is a high probability that the driver is injured and requires urgent medical support. This use case illustrates the integration of both data sources and how to execute complex behaviors that rely on temporal relationships. The situation types defined for this use case include (UC03.ST01) collision followed by bradycardia within an interval of 20 seconds.

The situation types defined for this use case are:

(UC03.ST01) Collision followed by bradycardia, observations computed by mobile app and situation identified by EWS (cloud): based on collision detected according to UC01.ST01 and bradycardia detected according to UC02.ST01, within an interval of 20 seconds.

(UC03.ST02) Bradycardia followed by collision, observations computed by mobile app and situation identified by EWS (cloud): similar to UC03.ST01 but in the opposite order, i.e., first bradycardia detected. For example, within 20 seconds time window, a situation of two consecutive bradycardia occurrences are followed by a collision detected.

(IIOT-UC04) Accident involving dangerous goods

This use case extends UC01, UC02 and UC03 by monitoring the type of goods being transported (logistics data), taking into account the dangerous goods classification of the United Nations Economic Commission for Europe (UNECE), namely (1) explosives, (2) gases, (3) flammable liquids, (4) flammable solids (substances liable to spontaneous combustion; substances which, on contact with water, emit flammable gases), (5) oxidizing substances and organic peroxides, (6) toxic and infectious substances, (7) radioactive material, (8) corrosive substances, (9) miscellaneous dangerous substances and articles, and (10) hazardous substances.

When detecting an emergency, the IoT EWS informs the type of dangerous goods being transported, indicating the need of adequate equipment to deal with the material transported for effective response. This use case illustrates the integration of data from both domains (health and logistics). Three situation types were defined for this use case according to each prior use case: (UC04.ST01-3) vehicle collision, cardiovascular risk, and vehicle collision with cardiovascular risk, respectively, with dangerous goods being transported, which are informed manually through the mobile app when a trip starts.

The situation types defined for this use case are:

(UC04.ST01) UC01 with dangerous goods: a situation type of UC01 with dangerous goods being transported, informed via the mobile app.

(UC04.ST02) UC02 with dangerous goods: a situation type of UC02 with dangerous goods being transported, informed via the mobile app.

(UC04.ST03) UC03 with dangerous goods: a situation type of UC03 with dangerous goods being transported, informed by the mobile app.

7.2 INTER-IoT-EWS Solution

To tackle the three main challenges for semantic interoperable IoT EWS and to implement the envisioned INTER-IoT use cases, we developed the INTER-IoT-EWS¹²⁴ solution. INTER-IoT-EWS enables semantic ECG and logistics data streaming with standardized ontologies serialized as JSON-LD, improving existing solutions for vehicle tracking, and extending some of the components of the *Azure MyDriving* use case (described in Chapter 3) and ECG monitoring.

7.2.1 Microsoft Azure IoT: Reusing MyDriving

Some IoT platforms were planned to be integrated within the INTER-IoT project, including research-oriented open source platforms, such as *FIWARE*, *universAAL* and *OpenIoT*, and the commercial proprietary platforms, such as *Microsoft Azure IoT*, *INTEL IoT* and *AWS IoT*. We performed some experiments to develop simple solutions using the research-oriented open source platforms, which required a greater effort with a long learning curve. In addition, some issues were found, such as technical bugs and documentation flaws, and several components are still on early stages of development, making the support of the involved community necessary. On the other hand, the commercial platforms showed to be simpler to use, with easier integration and deployment of components, as well as reliable documentation and full support.

In particular, the *MyDriving* solution fit well to the INTER-IoT-EWS solution because of its functional capabilities, i.e., it already provides vehicle trips tracking, being able of monitoring real-time location, speed and acceleration data through a mobile device connected to the cloud infrastructure. Furthermore, the *MyDriving* use case is open source and is almost a “production-ready” solution, following the microservices’ architecture with automatized deployment for continuous delivery. Therefore, the *MyDriving* solution was chosen, which lead to the reduction of the development efforts of INTER-IoT-EWS.

Part of the *MyDriving* solution is aligned to the SEMIoTICS framework, since it uses a cloud gateway (broker), the *IoT Hub*, to publish data provided by IoT devices. The *MyDriving* mobile app behaves as a field gateway providing data as JSON messages, which can be enhanced for brokering semantic data by following the SEMIoTICS approach. Therefore, we adapted the *MyDriving* data interfaces, replacing them with specific ontologies (RDF serialized as JSON-LD) to represent logistics and e-Health domains, i.e., the data interfaces between the *MyDriving* app and the *IoT Hub* were changed to adhere to the *Tbox* of the ontologies used.

¹²⁴ <https://github.com/jonimoreira/INTER-IoT-EWS/>

The main components adopted from the *MyDriving* solution were the mobile (*Android*) application, the *IoT Hub* and the application service along with its database service. Since the native *MyDriving* app is not capable of representing data in RDF, changes in the mobile app were required to transform the app to a *MyDriving for Linked Data (MyDriving-LD)* version. *MyDriving-LD* plays the role of logistics data provider, thus it requires an ontology that represents location, acceleration and the goods being transported, as well as relevant information about the trip (travel). The *MyDriving-LD* is not able to monitor ECG data and, thus, the integration with an existing IoT solution for real-time ECG monitoring was required.

7.2.2 IoT Real-Time ECG Monitoring: Shimmer3 ECG

Recently, a number of healthcare systems have been leveraged by the IoT technologies, which offer opportunities to improve patient monitoring and abnormal situation detection with support of medical wearables and cloud infrastructure. Some of these IoT healthcare systems emphasize the real time monitoring of cardiac behavior through ECG devices [259, 292, 293]. Modern ECG units can transmit the records of the electrical signals measured from the skin through wireless communications (e.g., Bluetooth), and are becoming smaller, more comfortable and energy-efficient. Examples of industry-oriented four-lead ECG devices for real-time outpatient monitoring are *Shimmer3 ECG*¹²⁵, *Berry PM ECG*¹²⁶, *BPL Touch-ECG*¹²⁷, *AliveCor EKG*¹²⁸ and the most advanced stick-on ECG devices, as *Isansys Lifetouch*¹²⁹ and *CardioCycle SenseOn*¹³⁰.

Real-time outpatient ECG monitoring is known as mobile cardiac telemetry [294] and can stream data continuously to caregivers, enabling the immediate alarm upon an abnormal situation detected, which can be implemented through an EWS. For example, the EWS for patients with cardiac diseases [102] runs in a mobile environment and collects a number of indicators from different sensors, such as ECG, blood pressure, body temperature and location. It is able to send the data to a service centre in hospitals and clinics using the internet. However, it lacks real-time streaming and abnormal situation identification based on these data. To the best of our knowledge, none of the ECG monitoring solutions support real-time semantic data streaming, a need emerged by the combination of the requirements IIOT-FR2 (monitoring real-time ECG) and IIOT-NFR2 (e-Health semantic model).

¹²⁵ <http://www.shimmersensing.com/products/ecg-development-kit>

¹²⁶ <http://www.shberrymed.com/>

¹²⁷ <https://www.bplmedicaltechnologies.com/>

¹²⁸ <https://www.alivecor.com/>

¹²⁹ <http://www.isansys.com/en/products/sensors>

¹³⁰ <http://cardiocycle.net/>

We performed a study to select an industry-oriented four-lead ECG device for real-time outpatient monitoring. The main characteristics needed were: (1) the solution must be one of the leaders in the market, thus, a recognizable, reliable and affordable four-lead ECG solution for patient-monitoring; (2) the solution should be able to record high-frequency (e.g., 256Hz) ECG not only on resting and ambulatory subjects, but also during activities; (3) the solution should provide configurable sensor module and open-source API for data acquisition in mobile environment (e.g., Android), being able to support JSON and RDF libraries; (4) preference is given to solutions that are built with the same technology as the *MyDriving* mobile app (.NET/C#, *Xamarin*).

Shimmer3 ECG covers all these needs and has high accuracy and usability, being able to transmit data from a *TinyOS* application (running within the ECG device) to a mobile phone application (*Android*) via *Bluetooth* [295]. It is used by the *Fall-MobileGuard* [289] and *BodyCloud* [296] approaches for detecting falls of elder people based on the cross-axial function. In addition, the *Shimmer3* ECG solution provides an API for mobile environment in .NET/C# with *Xamarin* with a series of implementation examples, including the *ShimmerCapture*¹³¹ mobile app (*Android* v3.0 or later).

Table 7.4. Data sources used by INTER-IoT-EWS.

External	Health	Logistics
Data	Driver's ECG, HR, accelerometer	Position, speed, accelerometer, goods
Device	Shimmer3 ECG (Capture), Mobile	Mobile (<i>MyDriving</i>)
IoT platform	MS Azure IoT	MS Azure IoT
Ontologies	SAREF, SAREF4health	SAREF, LogiCO, LogiTran, LogiServ

Therefore, we chose to integrate the *Shimmer3* ECG solution into the *MyDriving-LD* app, playing the role of health data provider. These data should be represented with an e-Health ontology for the generation of messages that encapsulates the ECG data. Both logistics and e-Health ontologies had to be integrated with an IoT ontology such as SAREF or SSN/SOSA. Table 7.4 summarizes the data sources involved, including the devices, applications, IoT platforms and ontologies used.

7.2.3 Solution Architecture

The INTER-IoT-EWS use cases involve the following actors and responsibilities: a truck driver, who works for a haulier company providing transportation of goods to/from the port, the haulier company, which is the owner of the IoT platform responsible for tracking the trips made by the trucks. The INTER-IoT-EWS

¹³¹ <http://www.shimmersensing.com/products/shimmercapture-for-android>

solution, which is a third party application for monitoring the data provided by the haulier IoT platform, integrates logistics and health data, providing emergency notifications to be consumed by the port emergency command centre, which is simulated in two ways: (1) INTER-IoT-EWS provides an UI based on *Google Maps* services, able to plot the data in real-time; (2) a third party partner of INTER-IoT project provides an emergency management system that consumes data from the port IoT platform and integrates with the emergency notifications from INTER-IoT-EWS, being able to coordinate the emergency response.

Figure 7.3. EWS to detect accident risks and accidents at the port of Valencia

Therefore, to address the functional requirements and implement the envisioned use cases, in INTER-IoT-EWS we integrated *MyDriving* and *Shimmer3* ECG solutions with RDF and JSON-LD technologies. The semantic field gateway (*MyDriving-LD* app) along with the *Shimmer3* ECG device represent the IoT haulier platform. The INTER-IoT-EWS cloud components, i.e., input and output handlers, context data, situation identification and reaction managers, implement the accidents detection and warning brokering mechanism. Determining vehicle collision (IIOT-UC01) and hazardous health changes (IIOT-UC02) are performed at the cloud level, i.e., the accelerometer and processed heart rate data provided by the mobile and the ECG device are sent from the mobile to the cloud, where the EWS implements the situation identification mechanism, including the composition use cases (IIOT-UC03 and IIOT-UC04).

Figure 7.3 illustrates the solution architecture of INTER-IoT-EWS. The *Shimmer3* ECG device collects data from the truck driver and sends them to the *MyDriving-LD* application, which includes logistics data and forwards these data to the cloud, acting as a semantic field gateway, publishing JSON-LD messages to the

Azure IoT Hub. In the main configuration, the INTER-IoT-EWS *Input Handler* subscribes to receive all messages published in the *IoT Hub*, checking each message type (IoT, health or logistics) and certifies whether translations to harmonize the data with the SEMIoTICS core ontology are necessary. If so, the *Input Handler* requests the translations to IPSM [163], forwarding the input message and the alignment required (pre-configured) to execute the translations. The IPSM is responsible for syntactically and semantically translating the data streams: from JSON-LD to the INTER-IoT JSON-LD syntax, which is a structured JSON-LD (an approach using two *@graph*) with middleware (INTER-MW) information.

Besides the configuration with semantic translations executed by the IPSM REST API, which is illustrated with the dotted red line in Figure 7.3, the INTER-IoT-EWS solution also provides an internal translation mechanism, which is the standard approach of using SPARQL to query input data. This configuration is termed as *Setup 1*, being totally agnostic to the INTER-IoT framework, i.e., does not require any component from INTER-IoT. The *Setup 2* is the configuration illustrated in Figure 7.3, connecting the *Input Handler* with the IPSM for semantic translations; and *Setup 3* uses the INTER-IoT middleware framework approach (INTER-MW), which requires the configuration of a “bridge application”. This “bridge application” subscribes to the *IoT Hub*, being able to receive the messages, and executes automatically the semantic translations with IPSM, posting the translated message to the INTER-IoT-EWS *Input Handler*.

Figure 7.4.
sosa:Observation as a
POCO class
generated from
SSN/SOSA

After the semantic translations, the *Input Handler* is responsible for the syntactical translation to internal POCO classes that follow the SEMIoTICS core context model, which are used by both *Context Data* and *Situation Identification Managers*. The classes are organized according to the core ontology alignments in the `INTERIoT-EWS.Context.DataObjects` namespace, e.g., for the IoT (SSN/SOSA) and emergency alerts (EDXL-CAP). Ideally, model transformations should generate automatically these classes from the OntoUML models. Figure 7.4 illustrates the *sosa:Observation* as a POCO class in C#, which is a core element serialized within a message and is instantiated with the sensor (through the *sosa:madeBySensor* property) as well as the observation identifier and the original message identifier. The observation identifier refers to a new identifier, which usually follows the terminology best practice of prefixing the instance identifier with the ontology instance URI. For example, for SAREF instances, the *sarefInst:[InstanceName]* pattern is used, whereas the very same instance translated to SOSA follows the *sosaInst:[InstanceName]* pattern. The message identifier links one observation to the original message.

Figure 7.4 (left) also shows some of the ontologies used in this case study, mostly derived from the SEMIoTICS core ontology. This approach also allowed us to add new elements that are specific for the application in the context model (new observation type), such as the *VehicleCollisionDetectedObservation*, which represents an observation specialization of vehicle collision processed by the mobile app (*MyDriving-LD*). Each application-specific observation type can be modelled in the context model, bringing some advantages, as the explicit representation of an observation type, facilitating the situation identification mechanism to detect the pattern where the new element is used.

The translation mechanism for *Observation* objects is illustrated as method in Code snippet 7.1, which creates observation instances according to the measurement property. For example, if the property is the vehicle collision computed by the ECG device or the mobile, then the translation method creates a *VehicleCollisionDetectedObservation* instance, adding this translated observation to the ordered list of observations (according to the observation timestamp) that are sent as events to the CEP processor. After executing the translations, the *Input Handler* sends, in parallel, the non-structured (“raw” JSON-LD) data to the *Context Data Manager*, and the structured data (POCO classes) to the *Situation Identification Manager*, both through microservices.

The *Context Data Manager* is responsible for annotating the data with the core ontology terms, applying heavyweight semantics by adding predicates from the foundational ontologies, and storing the data with two ways: (1) inserting the JSON-LD messages in a document-based (NoSQL) database (*MongoDB*) for historical data storage; and (2) inserting the RDF (from the JSON-LD) in a triplestore (*Virtuoso*) for querying semantic data. Tests of the total transaction time

showed that the approach (1) is faster for inserting but slower and more complex to query, while the approach (2) is slower for inserting but faster and easier to query, in spite of the semantic capabilities of SPARQL over JPath and other JSON query languages.

Code snippet 7.1. Translation of *sosa:Observation* (measurement property)

```

Observation observation = new Observation(measurementId, sensor, messageId);

// sosa:hasResult predicate
observation.hasResult = new Result();
observation.hasResult.hasValue = measurementValue;
observation.hasResult.hasUnit = measurementUnit;

// sosa:observedProperty predicate
string observablePropertyId = measurementProperty + "_" + Guid.NewGuid();
observation.observedProperty = new ObservableProperty(observablePropertyId, measurementProperty);

// sosa:resultTime predicate
observation.resultTime = measDateTime;

// Treat app-specific observations / processed data (e.g. collision detected, heart rate, etc)
switch (measurementProperty)
{
    case vehicleCollisionPropertyECGdevice:
        AddVehicleCollisionDetected(observation, result);
        break;
    case vehicleCollisionPropertySmartphone:
        AddVehicleCollisionDetected(observation, result, true);
        break;
    default:
        AddObservation(observation, result);
        break;
}

```

INTER-IoT-EWS is also aligned to the *MyDriving* architecture in terms of real-data handling (“hot path”) and historical data handling (“cold path”). Both real-time data and historical data are used by the *Situation Identification* component, which was implemented with NESPER event processing server, taking advantage of the SQL-like approach of EPL. Situation types are defined a priori, as rule sets, describing the risky situations of interest based on emergency plans. Each Situation Type is implemented as an EPL statement, which follows the structure of the POCO classes. For example, the code snippet 7.2 (top) illustrates the EPL *UC01_VehicleCollisionDetected_ST01* that represents the situation type of a vehicle collision detected by the mobile app from the ECG device acceleration data, using the *VehicleCollisionDetectedObservation* class. Below, the EPL statement *UC02_HealthEarlyWarningScore_ST02* represents the situation type of tachycardia (ST02) of UC02. The EPL on bottom represents the *UC02_HealthEarlyWarningScore_ST04*, which is an example of a more complex situation type that checks the occurrence of 3 tachycardia events in sequence within 20 seconds of the same trip (thus, same driver). By adopting the model transformation rules from SIMPLE approach (Chapter 2), these EPL statements

are generated automatically from the SML models, which are linked to the context model.

Code snippet 7.2. EPL statements for situation identification

```
// 7.1.1.1 Detected with ECG device accelerometer, computed by smartphone
// Get observation regarding the detected collision processed by the smartphone with high-frequency data from ECG device
// AND the cross axial function processed by the smartphone to detect the collision: join by
string statementName = "UC01_VehicleCollisionDetected_ST01";
var expr = @"
    SELECT o.madeBySensor.Identifier AS sensorId, o.Identifier AS observationId, o.observedProperty,
           o.resultTime AS TriggerEventBegin, o.hasResult.hasValue AS resultValue, o.hasResult.hasUnit AS resultUnit,
           o.madeBySensor.isHostedBy.location.Lat AS latitude, o.madeBySensor.isHostedBy.location.Long AS longitude,
           CrossAxialUsedToDetectCollision.hasResult.hasValue AS ComputedCrossAxialValue,
           o.madeBySensor.isHostedBy.tripId AS TripId
    FROM VehicleCollisionDetectedObservation.win:time(1 second) AS o
    INNER JOIN
        Observation.win:time(1 second) AS CrossAxialUsedToDetectCollision ON o.MessageId = CrossAxialUsedToDetectCollision.MessageId
    WHERE o.Value = true
        AND CrossAxialUsedToDetectCollision.observedProperty.Label = 'https://w3id.org/saref/instances#CrossAxialFunction'
";

// 7.1.2.2 Tachycardia detected with fixed threshold
string statementName = "UC02_HealthEarlyWarningScore_ST02";
var expr = @"
    SELECT o.madeBySensor.Identifier AS sensorId, o.Identifier AS observationId, o.observedProperty,
           o.resultTime AS TriggerEventBegin, o.hasResult.hasValue AS resultValue, o.hasResult.hasUnit AS resultUnit,
           o.madeBySensor.isHostedBy.location.Lat AS latitude, o.madeBySensor.isHostedBy.location.Long AS longitude,
           o.madeBySensor.isHostedBy.tripId AS TripId
    FROM Observation.win:time(1 second) AS o
    WHERE o.observedProperty.Label = 'https://w3id.org/saref/instances#HeartRate'
        AND o.hasResult.hasValue > " + uc02_thresholdTachycardia_query + @"
";

// 7.1.2.4 Multiple occurrences of tachycardia detected with fixed threshold
string statementName = "UC02_HealthEarlyWarningScore_ST04";
var expr = @"
    SELECT o.madeBySensor.Identifier AS sensorId, o.Identifier AS observationId, o.observedProperty,
           o.resultTime AS TriggerEventBegin, o.hasResult.hasValue AS resultValue, o.hasResult.hasUnit AS resultUnit,
           o.madeBySensor.isHostedBy.location.Lat AS latitude, o.madeBySensor.isHostedBy.location.Long AS longitude,
           o.madeBySensor.isHostedBy.tripId AS TripId
    FROM pattern[every o=Observation(hasResult.hasValue > " + uc02_thresholdTachycardia_query + @",
                                   observedProperty.Label = 'https://w3id.org/saref/instances#HeartRate')
        ->
        (
            (Observation(observedProperty.Label = 'https://w3id.org/saref/instances#HeartRate',
                         madeBySensor.isHostedBy.tripId=o.madeBySensor.isHostedBy.tripId,
                         hasResult.hasValue > " + uc02_thresholdTachycardia_query + @"))
        ->
        (
            (Observation(observedProperty.Label = 'https://w3id.org/saref/instances#HeartRate',
                         madeBySensor.isHostedBy.tripId=o.madeBySensor.isHostedBy.tripId,
                         hasResult.hasValue > " + uc02_thresholdTachycardia_query + @"))
        )
    ]
    where timer:within(20 seconds)
";
```

Each situation type is linked to a response process, i.e., the specific workflow to be executed once a situation is identified. The *Situation Identification Manager* embed a CEP server that is responsible for executing all statements, which check the input data patterns and when the pattern is matched, the CEP server triggers the event handler. The Code snippet 7.3 shows the event handler method responsible for the integration between the *Situation Identification Manager* and the *Situation Reaction Manager*, which is responsible for processing the workflow that includes the activities for brokering messages. The *Situation Reaction Manager* checks the information requirements of each alert target, passing this information to the output handler when a brokering message activity (from BPMN) is used. The *Output Handler* is responsible for transforming the data to semantically

compliant EDXL messages. Therefore, the output handler enables the brokering of notifications of the situations detected, following the JSON-LD syntax and the EDXL data model, which is able to link to the used semantics. A web UI application shows each alert sent by the EWS with its severity and urgency, and other information, including the targets that received the notification and the message sent to each target.

Code snippet 7.3. Situation Reaction Manager event handler

```
private void defaultUpdateEventHandler(object sender, UpdateEventArgs e)
{
    var attributes = (e.NewEvents.FirstOrDefault().Underlying as Dictionary<String, object>);
    try
    {
        if (sender is StatementResultServiceImpl)
        {
            StatementResultServiceImpl result = (StatementResultServiceImpl)sender;
            SituationType situationTypeIdentified = new SituationType(result.StatementName);
            Situation situationIdentified = new SituationIdentification.Situation(situationTypeIdentified, true, attributes);
            situationIdentified.Type.SituationReactionProcess.Execute(situationIdentified);
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Trace.TraceError("[SituationIdentificationManager] defaultUpdateEventHandler Error on situation res
ex.Message + Environment.NewLine + ex.StackTrace + Environment.NewLine + "InnerException:" +
((ex.InnerException != null)? ex.InnerException.Message: "NULL"));
    }
}
```

7.2.4 Role of Ontologies and Semantic Translations

The SAREF ontology was chosen to represent the IoT data at the device level because it fits better to the *MyDriving-LD* solution, which is industry-oriented. Moreover, SAREF provides the capability for tracking devices' energy consumption, used for IoT energy efficient solutions. While the semantic field gateway provides data following the SAREF ontology, INTER-IoT-EWS *Context Model* (core ontology) is aligned to the SSN/SOSA ontology, as described in Chapter 6. The semantic translations between SAREF and SSN/SOSA were implemented as SPARQL queries (*Setup 1*) and as IPSM language (*Setups 2 and 3*).

Code snippet 7.4 illustrates the SPARQL queries used for the semantic translations within *Setup 1*. The first query searches for devices specified in the input message, using SAREF, as well as the each device type and location (through the common approach of WGS84 ontology). The second query looks for all measurements provided in the message, including their value, time, measurement unit and related property.

The SPARQL queries used in *Setup 1* are used to reflect the five mappings between SSN/SOSA and SAREF, introduced in Chapter 6. As described in Chapter 3, the IPSM provides a template-based approach for semantic translations, similar to model transformations based on templates, such as XSLT for XML. For validation purposes, the five mappings were also described with IPSM language to

be used through a POST method from the translator requestor to the IPSM REST API, described in Chapter 9.

Code snippet 7.4. Semantic translations from SAREF to SSN/SOSA (SPARQL)

```
// Query for all geo:location (Device or Sensor) and map to hasLocation (Platform or Sensor)
string sparqlQuery = @"
PREFIX saref: <https://w3id.org/saref#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?devicesubclass ?device ?location ?lat ?long ?label
WHERE
{
    ?device geo:location ?location.
    ?device rdf:type ?devicesubclass.
    ?device rdfs:label ?label.
    ?location geo:lat ?lat.
    ?location geo:long ?long
}

";

// Query for all sensor measurements (SAREF) -> observations (SOSA)
// Order by timestamp (SAREF) -> Observation.resultTime
sparqlQuery = @"
PREFIX saref: <https://w3id.org/saref#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?sensor ?measurement ?measValue ?measUnit ?measProp ?measTime ?sensorLabel
WHERE
{
    ?sensor saref:makesMeasurement ?measurement.
    ?measurement saref:hasValue ?measValue.
    ?measurement saref:hasTimestamp ?measTime.
    ?measurement saref:isMeasuredIn ?measUnit.
    ?measurement saref:relatesToProperty ?measProp.
    ?sensor rdfs:label ?sensorLabel.
}
ORDER BY ?measTime

";
```

Code snippet 7.5 illustrates the sensor properties mapping (M01) with the IPSM language, which is a RDF serialized as a *Turtle* file. The *align:Cell* element represents one mapping, having the source entity characterized through the *align:entity1* property and the destination entity characterized through the *align:entity2*. A variable is used to store each triple pattern found in the source and is used to populate the destination. For example, the variable *var:CTA* is used to store the triples that contains the pattern *?sensor a saref:Sensor*, i.e., all elements with type *saref:Sensor*. This variable is used in *align:entity2* to populate the triple *sosa:Observation sosa:madeBySensor ?sensor*. Behind the scenes, the IPSM approach maps each alignment cell to an automatic generated SPARQL, executing each one and returning the results as a translated message.

In the beginning of the research, health data – both raw ECG and processed heart rate data – were represented as native SAREF. However, this approach showed to be quite verbose due to the lack of an element to represent time series, impacting data transferring and cloud infrastructure costs. SSN/SOSA also presented this limitation, which forced us to look for health ontologies that are adequate for this type of IoT scenario. Chapter 8 discusses this research cycle as part of the SEMIoTICS validation.

Code snippet 7.5. Semantic translations with IPSM (mapping M01)

```
<align:map>
  <align:Cell rdf:about="http://www.inter-iot.eu/sripas#Cell1">
    <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      var:CTA a saref:Sensor ;
      saref:measuresProperty var:CTD ;
      saref:makesMeasurement var:CTB .
      var:CTB saref:hasValue var:CTV ;
      saref:hasTimestamp var:CTT ;
      saref:relatesToProperty var:CTC ;
      saref:isMeasuredIn var:CTU .
    </align:entity1>
    <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      var:CTB a sosa:Observation ;
      sosa:madeBySensor var:CTA ;
      sosa:hasResult [
        a sosa:Result ;
        iiot:hasValue var:CTV ;
        iiot:hasUnit var:CTU
      ] ;
      sosa:observedProperty var:CTC ;
      sosa:phenomenonTime [
        a time:Instant ;
        time:inTimePosition [
          a time:TimePosition ;
          time:numericPosition var:CTT
        ]
      ] .
      var:CTA a sosa:Sensor ;
      saref:measuresProperty var:CTD .
    </align:entity2>
    <align:relation>=</align:relation>
  </align:Cell>
</align:map>
```

For logistics data, experiments with the *Logistics Core Ontology* (LogiCO) ontology¹³², along with its extensions *Transport* (LogiTrans)¹³³ and *Logistics Services* (LogiServ)¹³⁴ ontologies, showed that they are adequate to represent logistics data. LogiCO is a logistics core ontology developed by TNO, which provides the main elements for the logistics domain, as well as mappings (through annotations) to the international standards under the *United Nations rules for Electronic Data Interchange for Administration, Commerce and Transport* (UN/EDIFACT). LogiTrans is aligned to LogiCO and “defines the transport order that (...) specifies the user requirements

¹³² <http://ontology.tno.nl/logico/>

¹³³ <http://ontology.tno.nl/transport/>

¹³⁴ <http://ontology.tno.nl/logiserv/>

for transporting a cargo from one place to another”. LogiServ is aligned to both LogiCO and LogiTrans and provides elements to represent logistic services, such as goods transported by a truck linked to a company. These ontologies followed most of the best practices of ontology engineering, including the foundations with DUL [281].

Code snippet 7.6. Message generated by *MyDriving-LD* with logistics data

```

{
  "@id": "LogiTrans:TransportEvent_c38378a3-98d9-4cb6-9fc6-72b70b5c1e7a_0e7f357d-elaf-4b3c-acc5-7e83ecble63e",
  "@type": "LogiTrans:TransportEvent",
  "LogiCO:hasLocation": {
    "@id": "LogiTrans:TransportLocation_39.45477308_-0.32889051",
    "@type": "LogiTrans:TransportLocation",
    "LogiCO:hasID": {
      "@id": "LogiCO:Identifier_Location_39.45477308_-0.32889051",
      "@type": {
        "LogiCO:Identifier",
        "geo:Point"
      },
      "geo:lat": {
        "@type": "http://www.w3.org/2001/XMLSchema#float",
        "@value": 39.45477308
      },
      "geo:long": {
        "@type": "http://www.w3.org/2001/XMLSchema#float",
        "@value": -0.32889051
      }
    }
  },
  "LogiServ:hasTime": {
    "@id": "time:Instant_2018-10-01T12:35:27.1879150Z",
    "@type": "time:Instant",
    "time:inXSDdateTime": "2018-10-01T12:35:27.1879150Z"
  },
  "LogiTrans:hasTransportHandlingUnit": {
    "@id": "LogiCO:Truck_XPT01298",
    "@type": "LogiCO:Truck",
    "LogiCO:contains": {
      "@id": "LogiTrans:Cargo_c38378a3-98d9-4cb6-9fc6-72b70b5c1e7a_CargoId_TransportingGoodsAt_Location_2018-10-01T12:35:27.1880680Z",
      "@type": "LogiServ:Cargo",
      "rdfs:label": "Cargo being transported by a truck, contains goods",
      "LogiCO:isDangerous": false,
      "LogiCO:contains": {
        "@id": "LogiCO:Identifier_Truck_XPT01298",
        "@type": "LogiCO:Identifier",
        "rdfs:label": "XPT01298"
      }
    }
  },
  "LogiCO:hasID": {
    "@id": "LogiCO:Identifier_Truck_XPT01298",
    "@type": "LogiCO:Identifier",
    "rdfs:label": "XPT01298"
  },
  "dul:isComponentOf": {
    "@id": "LogiServInst:Transport_c38378a3-98d9-4cb6-9fc6-72b70b5c1e7a",
    "@type": "LogiServ:Transport",
    "time:hasBeginning": {
      "@id": "time:Instant_2018-10-01T12:31:27.1660840Z",
      "@type": "time:Instant",
      "time:inXSDdateTime": "2018-10-01T12:31:27.1660840Z"
    },
    "LogiCO:hasIDValue": "c38378a3-98d9-4cb6-9fc6-72b70b5c1e7a"
  }
}

```

Code snippet 7.6 illustrates a logistics message in JSON generated by *MyDriving-LD*, representing the mobile device location (*LogiCO:hasLocation*), the instant time of the message (*LogiServ:hasTime*), the vehicle (*LogiCO:Truck*), , the vehicle identifier (*LogiCO:hasID*), the goods being transported (*LogiServ:Cargo* *LogiCO:contains*) and the trip (transport) basic information (*LogiServ:Transport*), such as the identifier (*LogiCO:hasIDValue*) and the beginning time (*time:hasBeginning*). To avoid coupling the messages, the *MyDriving-LD* app creates messages according to SAREF4health, describing the related IoT devices (mobile and ECG device) and health data, and messages according to *LogiCO/LogiTrans/LogiServ*, describing the trip and logistics data.

Each health message is sent to the cloud according to the mobile-to-cloud frequency configuration. For example, if the ECG device is configured for 256Hz and the mobile-to-cloud method is configured to 3 seconds, the *MyDriving-LD*

sends one message with time series representing 768 ECG data measurements in each 3 seconds. The logistics message is sent according to the default configuration of *MyDriving*, i.e., either after 3 seconds or when moved for 10 meters, relying on GPS signal. Both message publishing methods support buffering to deal with Internet network connectivity loss.

7.3 INTER-IoT-EWS Functional Validation

The validation plan of the INTER-IoT-EWS solution shows how SEMIoTICS addresses the challenges listed in Chapter 4 and how the the specific INTER-IoT use cases requirements were realized. The functional validation activities included factory acceptance tests (FAT) and site acceptance tests (SAT) through a pilot in the port of Valencia, where accidents were simulated in accordance with the port emergency exercises. Both FAT and SAT assess whether the system works for the intended risks' detection and warnings. FAT were performed to check completeness and requirements verification, simulating each of the situation types of each use case. The non-functional validation, including the measurement of semantic interoperability and performance analysis of data processing and brokering, are described in Chapters 8 and 9, respectively.

The FAT were performed to check if the system is operational, functional and meets the specification, which took place in the *University of Twente* campus. It included a check of completeness and requirements verification, by simulating each of the situation types of each use case. Each of these test cases was executed, and the acceptance criteria was based on the output messages, i.e., whether the output message of the test case adhered to the expected behavior. For example, the test case UC01_ST01_02 provides several health and logistics messages (input data) provided by the *MyDriving-LD* app, which were generated (recorded) while driving in the campus area, thus, simulating a trip. One of these messages was changed in a way that the *Situation Identification Manager* could detect a collision and classify it as “green” (minor severity and expected urgency).

The test execution is started by an UI simulator, which reads all input data and simulates the *MyDriving-LD* app publishing data to the *IoT Hub* according to a pre-defined frequency (each 3 seconds). Then, the system should issue a warning to the targets, through their pre-configured channels, when the situation is identified. The functional validation covers the three setups described earlier: with “raw” SPARQL (1), with IPSM (2) and with INTER-MW (3).

7.3.1 Test Environment and Setups

To test the functionality of INTER-*IoT*-EWS in combination with the INTER-*IoT* framework, a more realistic test environment was built, comprising hardware, software, tools and platforms. Hooks were created to inject data or to control parts of the system behavior and the probes inserted provide logging capabilities (system feedback), showing that the system is handling the scenario as it should. In summary, the test environment reflected the deployment components illustrated in Figure 7.3, according to INTER-*IoT* framework layers:

- **Device:** the UI simulates two devices for each driver, a smartphone, acting as a the field gateway (*MyDriving-LD* app), and an ECG device (*Shimmer3* ECG unit).
- **IoT platform:** *Azure* IoT platform with an *IoT Hub* instance as *Azure* subscription *Basics 1 (B1)*, which offers the basic services required to run *MyDriving* app, i.e., a mobile app service (app server) and a SQL Server database (DB server).
- **Middleware:** as described before, three setups were considered: (1) without any INTER-*IoT* component; (2) with IPSM for semantic translations; and (3) with INTER-MW to subscribe to the IoT hub instance and receive the messages through a bridge, executing the semantic translations with IPSM and providing the translated data to INTER-*IoT*-EWS by posting to the *Input Handler* REST API. Setup 1 and 2 were tested, while setup 3 could not be tested due to INTER-MW issues found by the INTER-*IoT* consortium.
- **Application:** the INTER-*IoT*-EWS cloud components (*Input handler*, *Context Data*, *Situation Identification* and *Situation Reaction* managers, and *Output Handler*) were developed with .NET/C# 4.5.2 core web API and deployed in different servers under an *Azure* cloud environment (subscription) of North Europe, complying to the GDPR. The *Situation Reaction Manager* implements a workflow that sends emergency notification to two targets, simulating the port authority and the port emergency command center. For the former, all information is added to the message, simulating that the target required all data related to the emergency situation. For the latter, only information about severity, urgency, event code, general description, time and location are made available. The emergency notification messages are always published in the *IoT Hub*. The first target receives the emergency notification through two channels: a pre-defined e-mail address and an UI application with *Google Maps* that is able visualize the trip in real-time.

Test hooks created include data injection to increase the instantaneous acceleration data for the collision detection and to change the processed heart rate to below the bradycardia threshold and above the tachycardia threshold. The data

injection hook also considers the more complex data patterns, for example for UC02_ST04 (multiple tachycardia) and UC03 (temporal relations between UC01 and UC02). Furthermore, a test hook was created to allow the modification of the thresholds at runtime, useful for the SAT when simulating the use cases during the pilot at the port. For example, the simulation of UC02_ST01 (bradycardia) while driving at the port was performed by changing the threshold (e.g., to 80 bpm) during the recorded trip to a value that was above the driver's average heart rate (e.g., to 75 bpm).

Figure 7.5. UI prototype simulating a command centre receiving data at real-time

The UI prototype was developed to support the execution of the tests. It can be either used for input data injection, as well as to subscribe to *IoT Hub* to simulate how the EWS consumes the data, plotting the vehicle route in a map, including the emergency notification messages that the EWS produces. Therefore, this tool is used for both FAT and SAT. This application runs in debug mode (not deployed) with *Visual Studio 2015*, taking advantage of the testing tools provided by the IDE, such as the diagnostics tools for monitoring memory and CPU usage. Figure 7.5 illustrates the UI prototype, with the map on the left and the measurements (acceleration, cross-axial function, heart rate and battery level) on the right. The test probes log the interactions between components within the test environment, shown on the bottom. In particular, the scenario illustrated in Figure 7.5 shows consecutive collision detected as result of the emergency exercises performed in the port of Valencia.

7.3.2 Execution and Result Analysis

Each situation type of each use case was validated in FAT. Test cases were created for each situation type following the severity/urgency classifications, thus, each situation type has five test cases to reflect a normal operation situation and each range of severity/urgency. For example, the test case UC01_ST01_01 represents the normal operation without any emergency warnings, while UC01_ST01_05 represents an extreme and moderate collision (highest level). Each test case has a set of input files, named TX.Y.json (input and output folders), and the expected output file(s). The acceptance criteria is to check whether the expected output files are similar to the real output files and check if the respective e-mail notification has been issued. All test cases are system testing using scripted data and each test case is performed in both TS_01 and TS_02.

Figure 7.6.
Identification of
UC02_ST03_02
when third
bradycardia is
detected

In total, 44 test cases were executed, resulting in 43 successfully executed, thus, 97.7% of test coverage. The only failed test case was UC03_ST02_01, simulating a bradycardia followed by a collision detected. Although the EPL statements for UC01_ST01 and UC02_ST01 executed correctly, the EPL statement for UC03_ST02 does not trigger the situation identification. A possible explanation for this issue is an open issue of the NESPER component (CEP server) related to C# implementation of dictionaries with data measurements¹³⁵. Figure 7.6 (top) illustrates the input data of the test case UC02_ST03_02 executed within the FAT and the messages received (bottom). The situation shows that when the third bradycardia is detected – the third decrease peak of heart rate (left/bottom chart) – the EWS issues the emergency notification.

¹³⁵ <https://github.com/espertechinc/nesper/issues/20>

The SAT execution occurred on October 1st, 2018, at the port of Valencia. At first, test data were collected by using the ECG device and *MyDriving-LD* within the port area. These test data were used for the execution of all test cases, following FAT procedures. Then, the pilot was executed with a volunteer wearing the ECG device while driving at the port streets. UC01 was simulated by changing the acceleration thresholds and stopping abruptly with the vehicle. Specifically, the threshold was set to 1.6G (instead of 3G), which is an acceleration that reflects an abrupt break, enabling the EWS to detect a lightweight collision successfully (“green” color, according to Table 7.1). The emergency message was issued and the notification email was sent to the pre-defined targets. Similarly, UC02 (arrhythmia) was simulated by changing the bradycardia and tachycardia thresholds. UC04 (dangerous goods) was simulated by selecting the dangerous goods classification in the *MyDriving-LD* app at the beginning of the recording session, repeating the simulation of UC01 and UC02. The pilot could not include the UC03 simulation because of its complexity, which involves multiple occurrences of different events within a time window, e.g., a collision followed by three bradycardia. Figure 7.5 is a snapshot of the SAT execution at the Valencia port in which consecutive vehicle collisions are detected.

The fulfilment of all INTER-IoT-EWS requirements was validated. For IIOT-FR1 requirement, the solution enabled an IoT platform (*Azure IoT*) to be able to coordinate with emergency systems through the most appropriate interoperability standards for input data (SSN/SOSA and SAREF) and for emergency notification (EDXL-CAP). In addition, the test cases addressed all envisioned use cases. IIOT-FR2 requirement was covered by the IoT haulier platform providing cardiac data of drivers in real-time, both raw ECG and processed heart rate data being available through a context broker (*IoT Hub*). The identification of cardiac issues were covered by the arrhythmia situation types, processed by the *Situation Identification Manager* and made available to emergency systems through an emergency interoperability standard (EDXL-CAP). IIOT-NFR1 requirement was covered by addressing both syntactic and semantic interoperability in the INTER-IoT-EWS component. Semantic interoperability was improved by the adoption of the ontologies and, in particular, the development of SAREF4health, which is presented in Chapter 8. Syntactic interoperability was covered by the adoption of JSON-LD, while semantic translations were tested using two approaches, the “raw” SPARQL (internal to INTER-IoT-EWS) and IPSM (external to INTER-IoT-EWS). Both JSON-LD and translation performance are analyzed in Chapter 9. IIOT-NFR2 requirement was also covered by the integration of the health (SAREF4health) and the logistics (LogiCO/LogiServ/LogiTran) ontologies at the application level (Input Handler). Finally, IIOT-NFR3 was also covered, having the battery level of the ECG device provided (in real-time) to the EWS. However, none of the situation types make use of the battery level.

7.4 Chapter Conclusions

In this chapter, the functional validation of SEMIoTICS is presented through use cases at the port of Valencia within the H2020 INTER-IoT project. The SEMIoTICS framework guided the development of INTER-IoT-EWS, having the solution architecture derived from the SEMIoTICS reference architecture. Furthermore, SEMIoTICS methodology supported the analysis of the involved technologies and guided the integration of the components, considering the requirements of the INTER-IoT project. In total, 44 test cases were performed, of which 43 have been successful, thus, this validation presents 97.7% of test coverage. Although the use cases are not so complex, they represent realistic use cases that allowed us to be concrete about how the IoT and semantic technologies are applied.

Besides covering the INTER-IoT requirements, this validation showed that SEMIoTICS is able to tackle the challenge of data analysis (**C3**, defined in Chapter 4). INTER-IoT-EWS illustrates how to use the modelling capabilities for temporal relations among events and situations (**Req7**) by implementing the *after* operator (*Allen*) through the *pattern[every...]* statement of EPL, used in the *UC02_HealthEarlyWarningScore_ST02* statement for consecutive heart rate observations above the tachycardia threshold. This statement also illustrates the representation of complex rules over context data (**Req8**) through the time window operator of EPL (*timer:within*), where the consecutive tachycardia observations should happen within 20 seconds. INTER-IoT-EWS solution also demonstrated how it issues warnings to multiple targets (**Req9**) by adopting the EDXL-CAP semantic model, where each property is mapped to an information requirement of each target. Moreover, this validation showed how SEMIoTICS addresses the semantic integration challenge (**C1**) in terms of **Req2** requirement, i.e., processing data from different sensor types, such as the ECG and mobile devices, described with different semantic models, such as SAREF, SSN/SOSA and LogiCO/LogiTran ontologies.

The main lessons learned with this validation regards the use of IoT platform components. On one hand, the deployment and configuration of a single component, such as the *IoT Hub* or a microservice, has been quite straightforward and easy. On the other hand, the integration of all components to make the solution work as intended was hard and time-consuming, especially because of their asynchronous characteristics. In particular, the *Azure* IoT platform provides a portal to monitor the components separately, which is common in other industry-oriented IoT platforms (e.g., AWS), but this portal fails when monitoring the integration behavior of the solution, making a big manual effort or the development of a monitoring tool necessary.

In addition, the *Azure* IoT platform provides alternatives to route the messages published in the *IoT Hub* through a query-based language and *Event Hubs*, which should simplify the IoT solution development¹³⁶. *Azure* also provides a time series analysis feature¹³⁷ that can be useful for ECG data analysis. These specific capabilities are interesting to be exploited, but the decision to use each capability should be based on specific requirements. For example, the stream analytics approach of *Azure* IoT seems to be a better approach for the situation identification mechanism, but after tests we decided to use NESPER because the EPL language provides higher expressiveness in terms of pattern and time predicates. For example, while the EPL provides the *pattern* element, which facilitates writing the EPL statement, writing a similar statement in stream analytics requires multiple joins, resulting in a more verbose and less maintainable statement.

Although SEMIoTICS guides to develop the *Situation Identification Manager* as a REST microservice deployable in the cloud and running a CEP server, the INTER-IoT-EWS solution showed that this mechanism can be employed to some extent at the device level. Therefore, a hybrid approach for situation identification in both device level (e.g., UC01_ST01) and cloud level (e.g., UC04_ST03) made it possible to balance data processing. Processing at the IoT device level should be used for the simplest situation types, especially when they are independent of integration with external data. For example, in UC01_ST01, the collision detection through cross-axial function calculation is a simple function and only relies on the instantaneous tri-axial acceleration from the device. On the other hand, complex situation types that require event processing features, such as temporal relations and patterns, and depends on the integration with other data, should be processed at the cloud level, which enables processing scalability. For example, UC04_ST03 is a complex situation type, since it detects a sequence of three bradycardia within a time window of 20 seconds, followed by a collision while the truck delivers a dangerous good. The decision of where to implement the situation mechanism should also consider maintainability issues, since a cloud application service is centralized and its deployment is dependent on the EWS administrator, while the deployment of mobile app version updates depends on the end user.

¹³⁶<https://azure.microsoft.com/en-us/blog/azure-iot-hub-message-routing-enhances-device-telemetry-and-optimizes-iot-infrastructure-resources/>
<https://docs.microsoft.com/en-us/azure/iot-hub/tutorial-routing>

¹³⁷ <https://docs.microsoft.com/en-us/azure/time-series-insights/time-series-quickstart>

SAREF4health: IoT e-Health Ontology

This Chapter presents the second part of the validation of SEMIoTICS, describing how SEMIoTICS produces a well-founded ontology that is aligned to existing data representations, considering the restrictions of IoT scenarios (**C1**, **Req1** requirement). Therefore, this Chapter describes how a standardized ontology (SAREF) was extended to support the requirements of INTER-IoT-EWS, particularly the monitoring of cardiac behavior of truck drivers (IIOT-FR2), which required a new research cycle of ontology engineering. Section 8.1 presents a review of semantic models for health data, focusing on models that can represent ECG time series. Section 8.2 describes the comparison study with the semantic models selected for this context, according to four characteristics: quality, message size (payload), IoT-orientation and standardization. We discuss the verbosity problem of SAREF (or SSN/SOSA) messages when serializing frequency-based real-time ECG time series, which affects the payload aspect. Section 8.3 introduces the SAREF4health ontology, a vertical extension of SAREF that addresses the verbosity problem, considering the other three aspects. Section 8.4 presents the SAREF4health validation in twofold: first through the conventional competency questions approach and, second, through the semantic gateway developed within INTER-IoT-EWS solution. Finally, Section 8.5 discusses the lessons learned, including the trade-offs regarding the impact of design decisions and ontology quality, and our contributions.

8.1 Semantic Interoperability for Healthcare

Semantic technologies have been applied in healthcare in many developments, in which a number of semantic models have been produced. For example, the Open Biological and Biomedical Ontology (OBO) Foundry¹³⁸ is a popular ontology-

¹³⁸ <http://www.obofoundry.org/>

driven initiative in the biomedical informatics field, which provides a number of domain ontologies leveraged by BFO (see Chapter 2). SNOMED CT¹³⁹ and LOINC¹⁴⁰ are other relevant ontology-based approaches for clinical health terminologies. EDAM¹⁴¹ is also another relevant ontology, which describes a number of concepts with preferred terms and synonyms. The semantic healthcare lifecycle [297] provides guidelines on how to enrich health-related data by reusing ontologies that can be retrieved from ontology repositories, such as the BioPortal¹⁴², which is a comprehensive web repository of biomedical ontologies, and the US National Center for Biomedical Ontology, the BioOntology¹⁴³.

In the past years, a number of ontology-driven e-health solutions supported with IoT technologies were proposed [19, 122, 196, 259, 293]. IoT healthcare systems use medical wearables to monitor the patient vital signs, where emergency alerts are configured according to a pre-defined set of rules and urgency levels, such as the early warning score approach used by INTER-IoT-EWS. In this context, the cardiac behavior is one of the main features to identify the emergency level, which is a requirement from the case study presented in Chapter 7. Therefore, in this chapter, we focus on the semantic representation of ECG IoT solutions.

8.1.1 Semantic Models for ECG Data

Several solutions for mobile cardiac telemetry are leveraged by IoT technologies for real time monitoring of heart behavior [259, 292, 293]. Modern ECG units can transmit the records of the electrical signals measured from the skin through wireless communications (e.g., *Bluetooth*), as demonstrated through INTER-IoT-EWS with the adoption of *Shimmer3* solution. Although syntactic interoperability of an ECG system is covered by the OASIS standard for emergency services through EDXL-TEP, only few semantic models provide terms related to ECG, such as leads (unipolar, bipolar), measurement time series and recording session.

For example, OBO provides the *Eagle-I Resource Ontology*¹⁴⁴, which includes several terms for health-related instruments and protocols, including ECG monitoring device¹⁴⁵. Both SNOMED CT and LOINC define some terms related to ECG, such as ECG voltage and ECG abnormal finding. However, none of these represent ECG time series, which is the most precise way to monitor the cardiac behavior.

¹³⁹ <http://biportal.bioontology.org/ontologies/SNOMEDCT>

¹⁴⁰ <http://biportal.bioontology.org/ontologies/LOINC>

¹⁴¹ <http://edamontology.org/page>

¹⁴² <http://biportal.bioontology.org/>

¹⁴³ <https://www.bioontology.org/>

¹⁴⁴ <http://purl.obolibrary.org/obo/ero.owl>

¹⁴⁵ http://purl.obolibrary.org/obo/ERO_0000835

The use of HL7 v3 along with IEEE1451 in embedded devices to achieve end to end semantic interoperability on health systems is exploited in [122]. The HL7 aECG [298] is one of these standards, which was chosen by the US Food and Drug Administration (FDA) for clinical trials and implemented as a common lexicon approach, i.e., using XML schemas, running nowadays in several hospital information systems.

A review of ECG storage formats can be found in [299], which also includes the most used standards for ECG data serialization, the SCP-ECG, which stands for “Standard Communications Protocol for computer assisted electrocardiography” [300], and the DICOM, which stands for “Digital Imaging and Communications in Medicine”. HL7 aECG, SCP-ECG and DICOM comply to the international standard for medical device communication (EN ISO 11073) [301].

The review of ECG storage formats concludes that aECG inherits the verbosity of XML, since it produces large messages (25 times larger than a compressed approach), so that it may only be used in clinical drug trials and presents conceptual design issues that hinder its instantiation [299].

For emergency patient tracking, the EDXL-TEP presents only one predicate related to ECG (*twelveLeadECGInterpretation*), meaning “the interpretation of the patient's heart rhythm by the ECG device”, which is a limited support to represent ECG data. Other relevant standard for electronic health records (EHRs) is the OpenEHR¹⁴⁶, which is an open specification that aims at providing interoperability among EHR systems. OpenEHR supports the representation of clinical workflows and leverages on the concept of archetype, which was inherited from the Electronic Health Record Communication (EN 13606) standard.

The HL7 Fast Healthcare Interoperability Resources (FHIR) is an emerging open standard that is considered as the evolution of the HL7 standards, including aECG, for the new generation of health data exchange. HL7 FHIR defines a collection of “resources” (data model elements) that can be mixed and adapted for particular clinical contexts through the “profiling” process, which is quite similar to the OpenEHR archetypes. Along with W3C semantic web community, HL7 developed an RDF representation of the FHIR data model, which can be described and validated with *Shape Expressions* (ShEx) [59].

The central FHIR element is the *Observation* resource¹⁴⁷, which is used to support diagnosis and monitor progress, and is recommended for monitoring vital signs and device measurements. In particular, the “EKG example using Sampled Data”¹⁴⁸ demonstrates how to serialize ECG data using the *Observation* resource, which offers the property *valueSampledData*, which can be used to represent the

¹⁴⁶ www.openehr.org

¹⁴⁷ www.hl7.org/fhir/observation.html

¹⁴⁸ www.hl7.org/fhir/observation-example-sample-data.json.html

ECG sample sequence as a series of measurements of the heart electrical activity through the *SampledData* element, similarly to HL7 aECG. A *SampledData* provides a concise way to handle the data produced by devices that sample a particular physical state at a high frequency. A typical use for this element is to represent the output of an ECG device. The data type includes a series of raw decimal values, which are mostly simple integers, along with adjustments for scale and factor.

A well-founded ECG ontology (UFO ECG) [302] was designed with support of an ontological analysis of existing health standards and ontology-driven conceptual modelling with UFO. The main goal of UFO ECG is to serve as a reference “unified Electronic Health Record (EHR) model”, providing mappings to the most common standards that support the representation of ECG data, including HL7 aECG, SCP-ECG, DICOM and AHA/MIT-BIH [146].

Figure 8.1. Main elements of UFO-ECG ontology [302]

Figure 8.1 shows the main elements of UFO ECG. A *Recording Device* (the ECG device) is “used to acquire (to record) an ECG from a given Patient by means of electrodes” and is defined as a *Kind*. The *Recording Session* is a *Complex Event*, thus, inherits the *start* and *end* time properties, representing the “service in which the Patient is subject of ECG recording by some Recording Device”. The concept of time series can be represented with two elements:

(1) the *Observation Series* as a “series of observations evenly spaced in time”, is a *Complex Event* composed by (a *part-of* relationship) *Observations*, which are carried out in a *Recording session*;

(2) the *Sample Sequence* as an “ordered sequence of samples”, is a *Collective*, i.e., an *ultimate sortal* that has parts that play the same functional role in the whole, represented as an array of data units that results from an *Observation Series*.

In practice, the former represents the composition of the real world atomic events (*Observation*), while the later represents the digital signs of the *Observation Series* (through the *sample-sequence-of* relationship). The *Recording Session* coincides temporally, i.e., has equal *start* and *end time* instants (*Allen’s* interval relation) with the *Observation Series*. A *Sample Sequence* provides an array of numbers (to represent measurements, in which each measurement is quantified in millivolts (mV), i.e., the potential differences (p.d) of the observations evenly spaced in time according to a frequency (*Sample Rate*) measured in Hertz (Hz).

8.2 Limitations of ECG Representations for the IoT

Allowing the exchange of lightweight messages among medical wearables, gateways and cloud infrastructure is an important requirement of IoT solutions when processing big data. One of the main problems is that the verbosity of messages impacts the data exchange performance and cloud infrastructure costs. For example, the costs of the *Azure* cloud gateway (*IoT Hub*) varies according to the message payload size¹⁴⁹. A common approach in IoT to tackle this problem is to aggregate measurement data at the gateway level according to a certain frequency, transmitting time series from the local gateway to the cloud from time to time [296]. The drawback of this approach is that when aggregating a series of measurements of one element, metadata about each specific measurement is lost, affecting the ontological expressiveness of the messages.

Figure 8.2.
Serialization of
ECG time series in
FHIR and SAREF

¹⁴⁹ docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-messages-construct

UFO ECG has direct equivalences to FHIR. For example, while UFO ECG represents time series (series of measurements) with the *Sample Sequence* element, FHIR represents this concept through the *Sampled Data* element. A direct equivalence can be found also with OpenEHR. OpenEHR emphasizes the time series representation by including the concept of “history of events, i.e., as a time series, allowing all software to access data in a uniform way”. The root object (*History* class) provides a list of events as attribute, similar to UFO ECG and FHIR approaches.

SAREF does not define a term for the time series concept, only allowing the representation of each granular measurement of a time series through the *Measurement* element. Similarly, SSN/SOSA only provides the *Observation* element, which has the same meaning as *Measurement* in SAREF, and follows a similar structure. Although this approach improves the expressiveness of the data representation, and thus ontology quality, it produces an overhead to the message size when serialized. Figure 8.2 illustrates this effect with FHIR and SAREF. The time series data serialized as an array addresses the verbosity problem, while with each specific measurement serialized with its unit of measurement and related property enriches the ontological expressiveness of the message but results in verbose messages.

Therefore, more verbose messages are generated when instantiating ECG data with SAREF or SSN/SOSA, when compared to the very same data represented with FHIR RDF, UFO ECG or OpenEHR. For example, our experiments on ECG data serialization, when comparing a JSON-LD message that instantiates SAREF individual *Measurements* against the similar message in FHIR RDF, but with *SampledData* element, we found out that the SAREF message size is fifty times bigger (more verbose) than the FHIR message. We generated sample data using these semantic models, simulating messages representing that an ECG device configured for 256Hz frequency provides data to the semantic gateway, which accumulates data each 5 seconds before sending the message to the cloud, thus with 1280 (5 x 256) measurements per message. While the equivalent SAREF message size has around 5Mb, the FHIR message size has around 100Kb. The discrepancy between SAREF and FHIR messages is non-linear, i.e., the difference of message size grows exponentially according to the number of measurements.

While in FHIR, similarly to UFO ECG or OpenEHR, a measurement is only a number added to the *data* element (property of *SampledData*) representing data of the time series, in SAREF a *Measurement* is a data structure that implements a number of properties (e.g., *isMeasuredIn* and *relatesToProperty*). Different from SAREF, SSN/SOSA alleviates this issue by introducing the *hasSimpleResult* property class axiom of the *Observation* element, which enables the direct link to a number (literal), but still requires the representation of other properties (e.g., *observedProperty*) for each observation. Recently, an on-going effort to extend

SSN/SOSA¹⁵⁰, led by W3C and OGC, proposed the creation of the *sosa:ObservationCollection* element, aiming at representing homogeneous collections of observations that share the same object property values. However, it does not include an object property to represent the sequence of values. Therefore, due to this verbosity issue, we conclude that default SAREF and SSN/SOSA are not suitable for exchanging IoT-based ECG time series data. Some other developments address the ontological representation of time series [303, 304], but, to the best of our knowledge, none of them solve the verbosity problem for IoT scenarios.

Table 8.1. Comparison of semantic models for IoT-based ECG monitoring systems

Characteristic	UFO ECG	FHIR RDF	SAREF	SSN/SOSA
Quality	+	-	+	+
Message size (payload)	+	+	-	-
IoT-oriented	-	+-	+	+
Standardized	-	HL7	ETSI	W3C

We studied alternatives, identifying their benefits and drawbacks. Table 8.1 compares the semantic models considered in this study. The quality of the semantic models is the first characteristic that we compared, following the best practices of ontology engineering for the IoT, which is described in Chapter 3, more specifically the best practices #8 and #12, concerning common ontology pitfalls and ontology reuse, respectively. While UFO ECG, SAREF and SSN/SOSA are built upon common conceptualization mechanisms (DOLCE) and reuse existing ontologies, FHIR RDF has a poor quality level because it is a (semi) automatic serialization of FHIR lexicon standard. Thus, FHIR RDF is a straightforward serialization of the FHIR standard data model, which causes an overload of object properties, e.g., it has numerous properties that represent the “description” concept (as *fhir:Evidence.description*, *fhir:ElementDefinition.slicing.description*, *fhir:CarePlan.activity.detail.description*), and lacks reusing common terms, e.g., instead of having all these “description” terms, it could reuse *dc:description*.

The message size (payload) indicates whether the semantic model allows the representation of time series, as in UFO ECG and FHIR RDF. We also checked whether the semantic model is appropriate to the IoT context, i.e., whether it provides grammatical constructs for the main concepts of the IoT domain, e.g., sensor, actuator, protocol, observation, unit of measurement, measurement properties, among others. Specific characteristics of the ECG domain were considered in this analysis, such as the mereology of an ECG device, the elements participating in an ECG recording session, the ECG leads responsible for measuring frequency-based sample sequences (time series). Since our goal is the development of an EWS for the detection of risks of truck accidents, we also considered how to

¹⁵⁰ <https://w3c.github.io/sdw/proposals/ssn-extensions/#sosa:ObservationCollection>

represent and retrieve multiple sensors data, such as acceleration data to calculate tri-axial function (impact) and ECG data to calculate ECG waveform features. SSN/SOSA and SAREF are the most appropriate IoT reference ontologies, as mentioned earlier, while some FHIR RDF resources can be used to represent some of the IoT constructs. Finally, we evaluated whether the semantic model is standardized by a standardization body, such as HL7, ETSI and W3C for FHIR, SAREF and SSN/SOSA (respectively).

From this comparison study, we concluded that there is no standardized IoT ontology that provides adequate balance of quality and payload representing time series for ECG data. Therefore, the problem addressed in this paper is how to achieve these four characteristics in a single ontology, which led us to the development of SAREF4health as a SAREF extension with emphasis on representing ECG data.

8.3 Ontology Development

The main information requirements of this ontology regard the ECG and the IoT domains, described as competency questions (Table 8.2). While UFO ECG provides a high quality and deeper ontological analysis of the ECG domain, with rich descriptions following foundational categories, SAREF can be used as an IoT reference model. Therefore, a set of competency questions was defined according to the main elements of these reference ontologies. The scope of these questions was narrowed with the information requirements of the use case validation of INTER-IoT-EWS to detect truck accidents, described in the prior chapter.

Table 8.2. Competency questions to be responded by SAREF4health

ID	Textual description
CQ01	What is an ECG device and how it is composed (mereology)?
CQ02	What are the elements participating in an ECG recording session?
CQ03	What is an ECG lead, what are the types of ECG leads, what type of property an ECG lead measures and what type of measurement an ECG lead can measure?
CQ04	What is an ECG sample sequence?
CQ05	What is a time series of measurements?
CQ06	What is the frequency (rate) measurement of an ECG sample sequence?
CQ07	How to represent tri-axial acceleration data from accelerometers of an ECG device?
CQ08	How to integrate measurements from multiple sensors (e.g., ECG leads, accelerometer and battery monitor) of an ECG device for near real-time (frequency-based) monitoring?

8.3.1 Methodology

The SAREF4health development was leveraged by the interactive and iterative approach experienced in the other standardized extensions of SAREF. The

extension was created in a transparent manner to allow stakeholders to provide input and follow the evolution of the work. The first step comprised the requirements' collection to guide the implementation and validation of the ontology, described in Chapter 7.

An in-depth theory review was performed to understand the cardiac behavior domain and how it can be monitored by sensors. Numerous information sources were analyzed, as specifications, datasets, standards, APIs and data formats, as well as domain expert opinions and existing initiatives in the healthcare domain. The second step comprised the use case collection, defined in natural language. The third step comprised the definition of the purpose and scope of the ontology for monitoring the cardiac behavior of drivers in a way that abnormal situations, e.g., arrhythmia (bradycardia and tachycardia), could be detected by INTER-IoT-EWS, which are reflected in the competency questions.

For the design of SAREF4health, we applied ontological analysis of the ECG domain. The UFO ECG ontology played the role of a unified well-founded EHR reference model representing the ECG domain, covering the main elements required for the INTER-IoT-EWS use cases. A side effect of this approach is that UFO ECG has mappings to HL7 aECG, which FHIR is based upon, thus enabling straightforward alignments to FHIR RDF.

The SAREF4health conceptual model built with OntoUML was used as context model, as guided by SEMIoTICS. It was designed according to decisions about the definition of terms harmonized from the semantic models analyzed and the alignment to existing ontologies. Then, the UFO cyclic semantic enhancement process was performed, aiming at increasing the SAREF4health ontology quality through model assessment based on formal lightweight verification and validation activities. Therefore, by extensively creating random instances we have been able to improve the quality of our ontology.

8.3.2 Design

The main definitions used to support the SAREF4health elements are described in Table 8.3. Although UFO ECG and FHIR standard were carefully designed with the support of healthcare experts, we decided to review the terminology used by them. For example, why UFO ECG uses the term *Sample Sequence* while FHIR uses *Sampled Data* to describe the very same concept? Either way, the terms reflect the *Time Series* collective concept, i.e., a sequence of data units in successive equally spaced points in time: each sampled data unit plays the same role in the series (whole).

In SAREF4health, we termed the element *Measurement Time Series* since it refers to a time series of a sequence of measurements made by a device and it is a term

often used in the measurement science (metrology)¹⁵¹. We avoided prefixing this term with *ECG* because it can be applied to other types of measurements. An *ECG Sample Sequence* is a *Measurement Time Series* that is measured in *Electric Potential* units (an array) and relates to the *Heart Electrical Activity* property. A *Sample* (UFO ECG) can be interpreted as a *Measurement* (SAREF), so we classify *Measurement* as a kind.

Table 8.3. Definitions related to time series in UFO ECG and standards

Term	Source(s)	Textual definition
<i>Sample sequence</i>	UFO ECG	Collective: “ordered sequence of samples resulting from an <i>Observation series</i> ” (ecgOnto:095).
<i>Sample</i>	UFO ECG	Kind: “Every observation produces an electrical potential <i>Sample</i> (in the scale of millivolts). The sample values are projections into the voltage (p.d.) quality structure within the set of the real numbers. Every sample is, in turn, a grain of a <i>Sample sequence</i> (...) with typical sample rates of 256 or 300 Hz”
<i>Observation series</i>	UFO ECG	Complex event: “Series of observations evenly spaced in time carried out in an ECG Recording session” (ecgOnto:093).
<i>Sampled data</i> <small>Observation.component.valueSampledData</small>	HL7 FHIR	“Data that comes from a series of measurements taken by a device, which may have upper and lower limits”.
<i>Time Series Observation</i>	O&M (ISO 19156)	“observation whose result is a time-series”.
<i>Series</i>	HL7 aECG	“Contains one or more sequence sets sharing a common frame of reference”.
<i>Series</i> General Series Module	DICOM	A property of General ECG that “specifies the attributes that identify and describe general information about the <i>Series</i> within a <i>Study</i> ”. A <i>Series</i> is as a sequence of data sharing a common frame of reference.
<i>History</i>	OpenEHR	“Root object of a linear history, i.e., time series structure. For a periodic series of events, period will be set, and the time of each Event in the History must correspond”
<i>Event</i>	OpenEHR	“Defines the abstract notion of a single event in a series. This class is generic, allowing types to be generated which are locked to particular spatial types”
<i>Recording device</i>	UFO ECG	Kind: “Device used to acquire (to record) an ECG from a given Patient by means of electrodes. Also called electrocardiograph” (ecgOnto:087).
<i>Recording device as recorder</i>	UFO ECG	Role: “Recording device as it plays the role of an ECG recorder” (ecgOnto:088).
<i>Recording session</i>	UFO ECG	Complex event: “Medical service in which the Patient is subject of ECG recording by some <i>Recording device</i> . The <i>Recording session</i> (event) can be said to temporally coincide, albeit in a different level of abstraction, with the <i>Observation series</i> (event). In other words, these two events have the same time boundaries”.
<i>Lead</i>	UFO ECG	Kind: “Viewpoint of the heart activity that emerges from an <i>Observation series</i> of the p.d. between two electrode placements on specific regions of the surface of the patient’s body” (ecgOnto:096)
<i>Lead</i>	HL7 aECG	“A vector along which the heart’s electrical activity is recorded as a waveform”
<i>Patient</i>	UFO ECG	Role: “Person as he/she plays the role of being subject of care, i.e., scheduled to receive, receiving, or having received a healthcare service (based on ISO/TC 18308:2003)”

¹⁵¹ <https://plato.stanford.edu/entries/measurement-science/>

Although the conceptual model was ontological consistent, for the implementation of the ontology we considered the introduction of a semantic issue when classifying *Measurement Time Series* as *Measurement* in SAREF, for two reasons:

- (i) this representation adheres to the definition of *Measurement*, i.e., measured value (*Electric Potential* units) of a property (*Heart Electrical Activity*);
- (ii) we reused the same structure of SAREF regarding class axioms of object properties, e.g., *hasTimestamp*, *isMeasuredIn* and *relatesToProperty*.

The main implications of this choice are twofold:

- (1) ontologically this specialization is incorrect, since a collective cannot specialize a kind;
- (2) the *hasValue* property limits the value domain of a *Measurement* to exactly one float number.

The *hasValues* property was added to overcome this issue, in which a *Measurement Time Series* can instantiate this property for an array of float numbers. In the conceptual model, instead of adding this specialization, the relation *makesMeasurement* between an *ECG Lead* and an *ECG Sample Sequence* makes this design decision explicit.

Finally, following the same approach of UFO ECG with *Sample Rate* data type and FHIR with *period* data property of *SampledData*, a class axiom was added in *Measurement Time Series* to relate it to a frequency (rate). For the definition of frequency, we reused SAREF4envi, importing the object property *has Frequency Measurement*.

An *ECG device* is usually referred as an ECG unit that plays the role of a *recorder* in the complex event (action) of *ECG Recording Session*. In SAREF, we can classify this complex action as a *Task* that an ECG device *accomplishes*. Therefore, *accomplishes* plays the role of the inverse of the *hasParticipant* relationship (same as *isAccomplishedBy*). The *hasParticipant* relationship between *ECG Recording Session* and *Person Under ECG Monitoring* is implemented with *dc:author* (Dublin Core). In addition, we aligned UFO ECG with SSN/SOSA by representing *ecg:RecordingDevice* as the role of a *sosa:Sensor* related to the recording function.

In UFO, the *event* stereotype provides the relations *start* and *end* to limit the event temporal boundaries, as used in UFO ECG. SAREF provides the *hasTime* property, a “relationship to associate time information to an entity”, thus, we specialized this relation as *hasStart* and *hasEnd*, adding them as class axioms of *Task*. In UFO ECG, the other participant in this event is a *Patient*, but in SAREF4health we adopted a different ontological commitment. We argue that someone does not need to be a patient (person under medical treatment) to participate in an *ECG Recording Session*. Thus, we introduced the *Person Under ECG Monitoring* element, i.e., a role of a *Living Person*, which is a phase exploited several times in the UFO research [142]. This ontological commitment is motivated by the INTER-IoT-EWS use case, since a driver is not necessarily a patient. In SAREF4health, a *Living Person*

is a *Person*, which is an element imported from DUL and is aligned to schema.org. We only consider ECG recording sessions applied to humans.

Usually, the frequency of an *ECG device* can be set through an API. This reflects the frequency of each *ECG Sample Sequence* measured during a *Recording Session*. Therefore, we added a class axiom to the *ECG device* element, the *has Frequency Measurement* property ranging on only *Frequency Measurements* (from SAREF4envi). Although it seems redundant, this approach is required to differentiate the current frequency of a device from a frequency used in prior sample sequences. For example, the device Shimmer3 ECG can be set for sampling frequency (rate) of 512Hz, which is recommended for clinical grade ECG, i.e., 512 data samples per second or an interval of 0.002 second between two consecutive data samples, from 0.05Hz to 8000Hz range. After collecting some sample sequences and before sending the message to the gateway, the frequency is set to 256Hz and new sample sequences are collected. With our approach the message describes the current frequency of the device (256Hz) and the frequencies used in each collected sample sequence (512Hz and 256Hz).

An *ECG Device* registers the *Heart Electrical Activity* with support of electrodes attached to different places of the body of a *Living Person* (assuming that the heart is beating in this living person). Two electrodes enable the measuring of an *ECG lead*, which is an electrical vector characterized by the depolarization of the heart resulted by the electrical signal between the atria and the ventricles. Manufacturers commonly characterize an *ECG device* by its number of *ECG leads*. An ECG device is composed by extremity electrodes, which must be attached close to the left arm (LA), right arm (RA), left leg (LL) and the right leg (RL); and chest (precordial) electrodes, which can vary from one unit to six units (V₁₋₆). By convention, lead I measures the electrical activity from RA to LA, lead II measures the electrical activity from RA to LL, lead III measures the electrical activity from LA to LL. The rule $\text{lead I} + \text{lead III} = \text{lead II}$ enables to derive a lead based on the other two. Leads I, II, III are known as *Bipolar Limb*. *Unipolar* leads measure the electrical activity from the Wilson's central terminal (negative pole) to each of the chest electrodes (positive poles).

For the sake of simplicity and to follow the industry terminology, we decided to represent an *ECG device* according to its *ECG leads*, classifying an *ECG lead* as a *Sensor*, since a sensor “detects and responds to events or changes in the physical environment” (SAREF). A lead can be either bipolar or unipolar, and an *ECG device* consists of at least one *Bipolar Limb* and one *Unipolar* lead. An *ECG lead* measures the *Heart Electrical Activity* property and *makes measurements* of *ECG Sample Sequence*. Figure 8.3 represents these main elements in an OntoUML model.

Figure 8.3.
SAREF4health
main elements
(green) with
instantiated
examples (blue)

8.4 Ontology Validation

The first phase of the SAREF4health validation is the consistency check, by using an on-line reasoner called *Ontology Pitfall Scanner* (OOPS)¹⁵². To do so, SAREF4health was implemented as RDF, i.e., the realization of the context model with the RDF language, being able to be serialized as XML, Turtle or JSON-LD formats, which was supported by the *Menthor* and *Protegé* tools. Then, the XML serialization was used as input to OOPS, which validated the ontology consistency.

The second phase of the validation checked the suitability of SAREF4health for the use case. Instances of SAREF4health were created as examples, and SPARQL queries were executed to answer each competency question. Figure 8.2 shows SAREF4health instances representing the *Shimmer3* ECG device and the ECG time series data aggregated at the mobile level before being submitted to the cloud, thus, following the INTER-IoT-EWS solution. Table 8.4 summarizes the results, whereas the “Result” column presents the resulted triples and class axioms, the “Description” column provides a natural language analysis of the result. We removed textual properties (e.g., labels and comments) from the results for the sake of conciseness. The following SPARQL query template has been used:

```
SELECT * WHERE { {?s1 ?p1 [Name of the term]} UNION {[Name of the term] ?p2 ?o2} . }
```

All competency questions could be answered, assessing the suitability of SAREF4health and, therefore, its semantic validity. Our ontological commitments were based on the definitions listed in Table 8.3 and were assessed by analyzing the SPARQL results. For example, *Time Series* element (CQ05) is equivalent to *Sample Sequence* (UFO ECG) and *Sampled Data* (FHIR). The SPARQL result shows that the structure allows the representation of an ordered frequency-based sequence of float data. The semantic validity provided a basic quality level for SAREF4health, which was leveraged by the application of our methodology based on the best practices for ontology engineering.

Table 8.4. Responding the competency questions with SPARQL queries

ID	Result	Description
CQ01	<ul style="list-style-type: none"> > rdfs:subClassOf saref:consistsOf min 1 ECGLeadUnipolar > rdfs:subClassOf saref:consistsOf min 1 ECGLeadBipolarLimb > rdfs:subClassOf saref:accomplishes only ECGRecordingSession > rdfs:subClassOf saref4envi:hasFrequencyMeasurement only saref4envi:FrequencyMeasurement > rdfs:subClassOf saref:Device < sarefInst:Shimmer3ECG_unit_T9JRN42 rdf:type 	An ECG device is a device composed by at least one unipolar and one bipolar limb leads. It accomplishes the task of ECG recording session and has a specific frequency measurement.
CQ02	<ul style="list-style-type: none"> > rdfs:subClassOf dc:author only LivingPerson > rdfs:subClassOf saref:Task > ECGDevice saref:accomplishes only ECGRecordingSession < sarefInst:RecordingECGSession_01 rdf:type 	An ECG recording session is a task in which a living person participates and is accomplished by an ECG device.

¹⁵² <http://oops.linkeddata.es/response.jsp>

CQ03	<ul style="list-style-type: none"> > rdfs:subClassOf saref:measuresProperty only HeartElectricalActivity > rdfs:subClassOf saref:makesMeasurement only ECGSampleSequence > rdfs:subClassOf saref:Sensor < ECGLeadBipolarLimb rdfs:subClassOf < ECGLeadUnipolar rdfs:subClassOf 	An ECG lead is a sensor that can be either bipolar limb or unipolar, which is able to measure a heart electrical activity, making measurements of the type ECG sample sequence.
CQ04	<ul style="list-style-type: none"> > rdfs:subClassOf saref:relatesToProperty only HeartElectricalActivity > rdfs:subClassOf saref:isMeasuredIn only ElectricPotential > rdfs:subClassOf TimeSeriesMeasurements < ECGLead saref:makesMeasurement only ECGSampleSequence < saref:inst:ECGMeasurementsSeries_Example001 rdf:type 	An ECG sample sequence is a measurement time series that relates to the property of heart electrical activity, measures in an electric potential unit. ECG sample sequences are measured by ECG leads.
CQ05	<ul style="list-style-type: none"> > rdfs:subClassOf saref4envi:hasFrequencyMeasurement only saref4envi:FrequencyMeasurement > rdfs:subClassOf hasValues only xsd:float > rdfs:subClassOf saref:Measurement < ECGSampleSequence rdfs:subClassOf 	A measurement time series is a measurement that has a frequency and a set of float values. Currently, the only type of measurement time series available is the ECG sample sequence.
CQ06	<ul style="list-style-type: none"> > saref4envi:FrequencyMeasurement > rdfs:subClassOf saref:Measurement > rdfs:subClassOf saref:isMeasuredIn exactly 1 saref4envi:FrequencyUnit > rdfs:subClassOf saref:relatesToProperty value saref4envi:Frequency 	The frequency measurement type imported from SAREF4envi is a measurement which is measured in a frequency unit and relates to the frequency property.
CQ07	<ul style="list-style-type: none"> > saref:inst:Shimmer3ECG_unit_T9JRN42 rdf:type :ECGDevice ; saref:consistsOf saref:inst:AccelerometerX_ECGDevice ; saref:consistsOf saref:inst:AccelerometerY_ECGDevice ; saref:consistsOf saref:inst:AccelerometerZ_ECGDevice ; > saref:inst:AccelerometerX_ECGDevice rdf:type saref:Sensor ; saref:makesMeasurement saref:inst:Measurement_AccelerationX_001 ; saref:measuresProperty dim:Acceleration ; > saref:inst:Measurement_AccelerationX_001 rdf:type saref:Measurement ; saref:hasTimestamp "2018-04-22T22:15:30"^^xsd:dateTime ; saref:hasValue "100"^^xsd:float ; saref:isMeasuredIn unit:metrePerSecondSquared ; saref:relatesToProperty quantity:acceleration ; 	Tri-axial acceleration data are represented according to each accelerometer sensor (X,Y,Z) of the ECG device. Each accelerometer sensor measures the property acceleration and makes acceleration measurements. An acceleration measurement is measured in metre per second squared, has a value and a timestamp.
CQ08	<ul style="list-style-type: none"> > saref:inst:Shimmer3ECG_unit_T9JRN42 rdf:type :ECGDevice ; saref4envi:hasFrequencyMeasurement saref:inst:FrequencyOf256Hertz ; saref:accomplishes saref:inst:RecordingECGSession_01 ; saref:consistsOf saref:inst:AccelerometerX_ECGDevice ; saref:consistsOf saref:inst:AccelerometerY_ECGDevice ; saref:consistsOf saref:inst:AccelerometerZ_ECGDevice ; saref:consistsOf saref:inst:ECGLead_III_code131389 ; saref:consistsOf saref:inst:ECGLead_II_code131330 ; saref:consistsOf saref:inst:ECGLead_I_code131329 ; saref:consistsOf saref:inst:ECGLead_Vx_RL_code131389 ; saref:consistsOf saref:inst:Shimmer3BatteryLevelSensor_T9JRN42 ; saref:hasManufacturer "Shimmer" ; saref:hasTypicalConsumption saref:inst:Shimmer3ECGBattery ; 	The ECG device mereology is responsible for representing the device sensors through the <i>consists of</i> property. This structure allows (near) real-time monitoring by accumulating the measurements (made by the sensors). This includes the measurement time series (e.g., made by ECG leads) and isolated measurements (e.g., acceleration and battery level).

The third phase of the validation focused on providing evidence about the semantic interoperability improvement in INTER-IoT-EWS solution, connecting the *Shimmer3* ECG device to the field gateway (*MyDriving* mobile app) to the cloud broker (*MS Azure IoT Hub*). The prototype developed incorporates the ECG device manufacturer's Android app (*Shimmer Xamarin Capture*¹⁵³) into the *MyDriving* mobile app, porting the ECG monitoring capability. The data received from the ECG device are aggregated and translated to SAREF4health, which enables different rates of data exchange between the ECG device and the smartphone and

¹⁵³<https://github.com/ShimmerEngineering/Shimmer-C-API/tree/master/ShimmerCaptureXamarin>

between the smartphone and the cloud, a common network bandwidth optimization requirement. This scenario also allowed us to exploit the SAREF4health approach for time series.

Code Snippet 8.1 shows the method that generates the ECG device messages as JSON-LD documents according to SAREF4health within *MyDriving-LD*, which is also responsible for publishing the data in the cloud gateway (*IoT Hub*). This method performs the serialization of the device mereology (through *saref:consistsOf*) as well as the relation to the *Recording Session* complex event. These characteristics show the benefits of the use of a foundational ontology during conceptual modelling, as an evidence of the semantic enhancement of this approach.

Code Snippet 8.1. Ontology-driven health IoT data exchange of an ECG device

```
public JObject GetECGDeviceJSON_SAREF4health(List<JObject> listDevicesOfDevice, JObject recordingECGSession)
{
    string deviceId = "sarefInst:Shimmer3ECG_unit_T9JRN42_DeviceId";
    JObject eCGDeviceJSON = JObject.FromObject(new
    {
        comment = "Shimmer3 ECG unit (T9J-RN42): INTER-IoT-EWS project",
        label = "Shimmer3 ECG unit T9J-RN42",
        seeAlso = "http://www.shimmersensing.com/products/ecg-development-kit#specifications-tab"
    });
    eCGDeviceJSON.Add("@id", deviceId);
    eCGDeviceJSON.Add("@type", JObject.Parse("[ 'saref4health:ECGDevice', 'saref:Device' ]"));
    eCGDeviceJSON.Add("saref4envi:hasFrequencyMeasurement", hasFrequencyMeasurement);
    eCGDeviceJSON.Add("saref:accomplishes", recordingECGSession);
    eCGDeviceJSON.Add("saref:hasManufacturer", "Shimmer");
    eCGDeviceJSON.Add("saref:hasTypicalConsumption", hasTypicalConsumption);
    eCGDeviceJSON.Add("saref:consistsOf", JToken.FromObject(listDevicesOfDevice));

    return eCGDeviceJSON;
}
```

The prototype was validated through functional tests regarding accident simulations with truck drivers transiting in a port area, discussed in Chapter 7, and the results show that SAREF4health is an adequate data representation considering performance issues of IoT solutions. In particular, the message size (payload) was measured and indicates an even smaller size to equivalent FHIR messages when representing time series. On average, a SAREF4health message representing the ECG time series of a 256Hz frequency, aggregated within the interval of 3 seconds, along with measurements representing average acceleration, processed heart rate and collision detection, has 35Kb, whilst a similar FHIR RDF message representing the same information has an average of 100Kb. The main reason for such difference is the FHIR RDF low quality regarding the representation of object properties, since the RDF is automatically generated from the FHIR lexicon standard. Therefore, SAREF4health is a better ontology for IoT-based ECG systems.

8.5 Chapter Conclusions

SAREF4health has not been published as an ETSI standard because some activities of the ETSI procedure (described in Chapter 3) need to be performed. A specialists' task force is ongoing to enable interactive validation with domain experts and other stakeholders, creating new validation scenarios. This has been performed through the STF566¹⁵⁴, which aims at reusing much of SAREF4health for the standardization of a SAREF extension for e-Health and aging well (SAREF4ehaw). Furthermore, the scope of the current version of SAREF4health may need to be extended to include new healthcare use cases besides ECG monitoring, which will require a new ontology engineering cycle. The SAREF4health maintenance is under discussion, depending whether SAREF4ehaw will replace SAREF4health or not.

Besides the SAREF4health ontology, an important contribution of this Chapter is the validation of the SEMIoTICS methodology, balancing conceptual modelling and RDF implementation concerns, where design decisions must be clearly exposed during conceptual modelling to understand the discrepancies between the conceptual model and the final ontology. The validation showed that a trade-off between ontology quality and lightweight data serialization was a crucial aspect on the design of the SAREF4health ontology.

The use of an ECG reference ontology grounded in UFO theory played a major role to provide ontology quality. Furthermore, the reuse of standardized ontologies showed to be essential to understand both IoT and healthcare domains, as well as implementation constraints. Therefore, we argue that SAREF4health has a high quality because of the ontological foundations inherited from UFO balanced with practical matters, such as making SAREF4health as dereferenceable as possible and checking and reusing popular and common ontologies. SAREF4health addressed the performance requirement regarding the impact of measurement time series in the message size (payload), i.e., the verbosity problem, validated through the INTER-IoT-EWS solution.

SAREF4health has higher ontological expressiveness because of its completeness for the ECG domain, since “the notion of completeness at the level of individual specifications is related to the notion of ontological expressiveness” [142]. Moreover, we claim that construct redundancy [305] was eliminated from SAREF4health regarding the elements about *Observation* from UFO ECG. Although these terms are ontologically correct, they bring additional complexity that are implicitly represented in SAREF through the *saref:makeMeasurement* property and the *saref:Measurement* class. While in SAREF a device makes a measurement, which has a timestamp, a value and is measured in a specific unit; in UFO ECG a device

¹⁵⁴ <https://portal.etsi.org/STF/STFs/STFHomePages/STF566>

carries an observation, which has a timestamp and this observation produces a sample, which has a value and is measured in a specific unit.

Therefore, SAREF4health validated SEMIoTICS w.r.t the semantic integration challenge (C1), particularly by enabling the development of a well-founded ontology that considered existing data representations, such as UFO ECG and FHIR, coping with performance restrictions (the SAREF verbosity problem). Therefore, we addressed **Req1** by tackling the verbosity problem, which also impacts on the processing challenge (C2), since the size of messages have a direct effect on the overall performance of the EWS (**Req4**). However, this validation was not enough to completely show how this performance requirement is addressed by SEMIoTICS, which is further discussed in Chapter 9.

As a side effect of this work, clear mappings between UFO ECG to FHIR were identified, as (UFO) *Observation series* to (FHIR) *Observation*, (UFO) *Sample sequence* to (FHIR) *Sampled Data*, (UFO) *Sample sequence* $>$ *sample-sequence-of* is the inverse of (FHIR) *Observation* $>$ *valueSampledData*. Other mappings can be easily extracted from the results of our ontological analysis. The SAREF4health ontology (in Turtle) and the prototype are available for download¹⁵⁵, and can be used in other e-Health scenarios supported by wearables, such as constant remote monitoring for assisted living, fitness, preventive medicine and aging well, such as in the VICINITY [306] and ACTIVAGE [307] scenarios.

A limitation of SAREF4health is that it lacks a concept to represent the physician(s) administering an ECG recording session, which is common in other standards (e.g., DICOM). To overcome this issue, we suggest to add a class axiom to *ECG Recording Session* with the *dc:creator*, similarly to the approach with *dc:author*.

Future work also includes a formal empirical validation to compare SAREF4health with other ontologies studied here, especially HL7 FHIR RDF. Semantic translations between SAREF4health and FHIR RDF can be useful to address the requirements of integrating different healthcare IoT platforms. Finally, after this full validation, SAREF4health will be submitted to the ETSI standardization process. Independent of whether SAREF4health will ever become a standard, we believe that this work provides an important contribution towards a higher-quality FHIR ontology, rather than a simple RDF representation of the FHIR lexicon data model. Although HL7 FHIR lexicon standard is widely adopted and used in industry (e.g., by Google and Apple), its semantic model is still in early stages.

¹⁵⁵ <https://github.com/jonimoreira/INTER-IoT-EWS>

Performance Validation

Chapters 7 and 8 described the validation of SEMIoTICS regarding semantic integration (C1) and data analysis for effective responses (C3) challenges. This chapter discusses how the challenge of processing safety-critical applications (C2) is addressed by SEMIoTICS. Section 9.1 compares the performance of JSON and JSON-LD, in terms of verbosity and impact on total transaction time, through a statistical difference-making experiment. Section 9.2 analyses the performance of the semantic translations between SSN/SOSA and SAREF, in terms of completeness, efficiency, accuracy and maintainability. Two translation mechanisms are compared, namely the SPARQL approach and the approach using IPSM (INTER-IoT component). Section 9.3 discusses the overall performance analysis of INTER-IoT-EWS, including a comparison between multi-cluster semantic brokering and the *Semantic IoT EWS* approach. Section 9.4 presents a cost analysis about the *Azure* IoT platform resources, including its cloud premises, that are required for INTER-IoT-EWS. Finally, Section 9.5 presents the validation conclusions, listing the contributions along with the threats to validity and limitations.

9.1 Comparison of JSON versus JSON-LD for IoT

Although JSON-LD has been designed to be lightweight, it causes an increase of the size of messages if compared to native JSON, due to the additional metadata. This section presents an empirical performance analysis of JSON-LD compared to JSON in common IoT scenarios. Event data produced by a sensor are serialized as JSON and JSON-LD, and the total transaction time is compared in request/reply and publish/subscribe architectures. Our goal has been to check the impact of the migration of IoT approaches from JSON to JSON-LD, and also to analyze the impact on performance related to JSON-LD (de)serialization, using a popular library (*JSON-LD processor JavaScript API* [236]).

9.1.1 Case Study Design

This performance analysis followed the Experimentation in Software Engineering methodology [65] and considered two common event-driven architectures [308], illustrated in Figure 9.1:

- Scenario 1 (S1): request/response
- Scenario 2 (S2): publish/subscribe message broker

Figure 9.1.
Common IoT
event-driven
patterns used in
this experiment

Our validation targets the performance of real time observations described with SAREF for environment (SAREF4envi). This is a standardized extension of SAREF that enables the representation of data about light pollution, exemplified with a photometer device that monitors light magnitude, sending the measurements to a server (see Chapter 3). This extension was chosen for this experiment rather than SAREF4health because SAREF4envi is already an ETSI standard and, thus can be considered a completely unbiased ontology. The sensor observation of light magnitude is used as message baseline serialized in JSON-LD (Code snippet 9.1), while a very simple structure is used as message baseline serialized in “pure” JSON, which only contains the raw data and no associated metadata, i.e., light magnitude equals to 0.1 at 2016-10-05T08:15:30.

The metric considered when formulating the hypotheses was the adequacy of JSON-LD for IoT solutions, which balances the benefits of semantics and performance. Performance is measured in terms of total transaction time of a sensor data producer interacting with a consumer, which includes data transfer and processing at both sides. The null and alternative hypotheses are, respectively:

(H0) The JSON-LD serialization is not adequate to be used in IoT solutions, i.e., the performance loss is too high compared to the semantics benefit.

(H1) The JSON-LD serialization provides the same performance of JSON or presents an irrelevant difference of less than 1%, i.e., the JSON approach is up to 1% faster than JSON-LD.

The first variable is the payload, i.e., the message size in bytes. The assumption is that small messages containing, for example, only one device measurement, yield

a greater delay in JSON-LD. This is due to the `@context` element of JSON-LD, which does not grow linearly with the message size, since the `@context` element only increases when new ontologies are used and/or when new classes and properties are used.

Three distinct SAREF4envi message were classified according to the number of measurements within the message and its respective size (in bytes), represented as (1) small: 1 measurement, JSON with 309 bytes and JSON-LD with 3389 bytes; (2) medium: 10 measurements, JSON with 4880 bytes and JSON-LD with 7567 bytes; and (3) large: 100 measurements, JSON with 42211 bytes and JSON-LD with 45264 bytes. Therefore, the overheads of the message serialized in JSON-LD compared to JSON are 996% for small (almost ten times), 55% for medium and 7% for large. The Code snippet 9.1 illustrates the small SAREF4envi message serialized as JSON-LD.

Code snippet 9.1. JSON-LD message using SAREF4Envi used in the experiment

```
"@id" : "example1TESS005-UCM",
"@type" : [ "s4envi:TESS", "owl:NamedIndividual" ],
"location" : "example1LocationTESS005-UCM",
"hasManufacturer" : "Universidad Complutense de Madrid",
"hasName" : "stars5",
"exposes" : "example1PowerProfileTESS005-UCM",
"hasFrequencyMeasurement" : "example1FrequencyMeasurementTESS005-UCM",
"hasTransmissionPeriod" : "example1TransmissionPeriodTESS005-UCM",
"hasVersion" : "v1",
"makesMeasurement" : [
  {
    "@id" : "example1Measurement2016-10-05T08:15:30TESS005-UCM",
    "@type" : [ "owl:NamedIndividual", "s4envi:Measurement" ],
    "label" : {
      "@language" : "en",
      "@value" : "Measurement 2016-10-05T08:15:30 TESS005-UCM"
    },
    "hasValue" : "0.1",
    "isMeasuredIn" : "example1mgPerarcsec2",
    "hasTimestamp" : "2016-10-05T08:15:30",
    "relatesToProperty" : "s4envi:LightMagnitude"
  }
],
"measuresProperty" : "s4envi:LightMagnitude",
"usesCommunicationInterface" : [ "example1Bluetoothv2-1",
"example1Wi-Fi-v802-11ah",
"example1RS232v24" ],
"usesCommunicationProtocol" : "example1MQTT-Broker",
"@context" : [ ... ]
```

Besides the message size, some simple processing was added in both sender and receiver sides, a simple *if* clause comparing a measurement with a constant, logging the result in the terminal (a simple I/O command). The second variable is the type of scenario, namely (S1) asynchronous request/reply; and (S2) publish/subscribe, which are illustrated in Figure 9.1. S1 includes two sensor data providers following the RESTful pattern, where one provides data serialized in JSON while the other provides the same data serialized in JSON-LD. The consumer sends 100

asynchronous requests to the provider in each transaction. An application computes the average of the total transaction time by executing the consumer 1000 times (synchronously). So, in total 1000 x 100 requests are forwarded to the provider.

S2 includes a lightweight message broker, where five topics are created in the broker to simulate five devices publishing event data. One subscriber is created, which listens to all topics. An application sends a synchronous request to the subscriber informing the beginning of the execution, so the subscriber is able to identify the first message and computes the total time on the arrival of the 100th message. The application simulates 20 asynchronous events for each topic. Each scenario was executed 1000 times, according to the configuration of the variables. The applications running these scenarios were implemented with *NodeJS*, which is commonly used in IoT solutions [206] and allows the implementation of REST/JSON services with a few lines of code¹⁵⁶.

In S1, the total transaction time is collected from the application for each request to the customer to avoid noise in both sides. Anomalous values were discarded, i.e., values greater than the mean plus the standard deviation. In S2, a popular open source broker (*Mosquitto*¹⁵⁷) was used. An application executed 5 publishers in parallel, so the subscriber computes 5x100 messages on each size (1, 2, 3).

Both scenarios were deployed in the cloud (using the *AWS EC2* service), having the provider/publisher deployed in a server in Brazil and the consumer/subscriber in a server in Australia. For S2, the MQTT broker was deployed in a third server in Australia, simulating a broker gateway near the device (provider/publisher). Brazil and Australia were chosen because they have the slowest link for data transfer within AWS regions. The intention was to perform the analysis in an environment that presents a relatively big difference in terms of data propagation, where the message size can affect the total transaction time.

The main issue regarding the migration of IoT approaches from JSON to JSON-LD is the impact on the message size, especially when few measurements are represented in the message. For example, a SAREF4envi message with one measurement is almost ten times bigger when represented with JSON-LD (~30Kb) in comparison to JSON (~4Kb). The increase of data volume impacts several cloud services, such as storage and brokering, and, therefore, their costs. For example, the *Azure IoT Hub* pricing relies on message size (4Kb as default), as mentioned in SAREF4health problem analysis (Chapter 8), so that the price of the standard tier *S1* is €21.09 (unit/month), where the total number of messages/day/unit is 400.000, thus, 4.6 messages/second or 18.51 Kb/second

¹⁵⁶ <https://github.com/jonimoreira/JSON-LD>

¹⁵⁷ <https://mosquitto.org/>

(message size of 4Kb). Considering an IoT solution operating on this threshold, for JSON messages containing 1 measurement, the very same solution with JSON-LD would require a *S2* tier, which costs €210.83 (unit/month), thus, multiplying the broker costs by ten. This issue is alleviated when the solution uses more measurements per message, thus, similar to the performance analysis results. If the message includes 100 measurements, then the costs are almost the same, since the message size difference is negligible.

9.1.2 Result Analysis

Scenario *S1* showed different execution performance with the different message sizes, where JSON is on average 0.15% faster than JSON-LD. Figure 9.2 shows the comparison analysis as a function, where the horizontal axis represents the experiments of each message size (1, 2, 3) in each scenario, having *S1* as green bar and *S2* as blue bar. The vertical axis represents the message size (on the left) and the speed difference (in percentage) comparing JSON over JSON-LD (on the right). For size 1, the difference of total time average was very small (less than 1 second), meaning that JSON is 0.19% (*S1*) and 0.17% (*S2*) faster than JSON-LD. For size 2, JSON is 0.12% (*S1*) and 0.11% (*S2*) faster than JSON-LD. For size 3, JSON is 0.05% (*S1*) and 0.07% (*S2*) faster than JSON-LD.

Figure 9.2.
Comparison
analysis of the
performance of
JSON and JSON-
LD

As the messages get larger, the difference between JSON and JSON-LD decreases. This behavior was expected because the message growth, in terms of number of measurements represented, is not linear with respect to the size in bytes. In JSON the size 1 is eleven times smaller than in JSON-LD, whereas in JSON the size 3 is only 1.08 times smaller than in JSON-LD. We conclude that if the device transmits more than 100 measurements, then JSON and JSON-LD perform almost identically. *S2* confirmed our findings from *S1*, since it provides similar results. The only difference observed in *S1* is a small increase on the total transaction time (less than 0.01%), which was expected because of the broker. This also indicates that the choice for *Mosquitto* as a lightweight broker was appropriate, causing minimal impact with a few publishers sending data to a topic.

Therefore, this validation showed that **Req4** requirement is fulfilled, since the IoT solution using JSON-LD provides adequate performance in terms of the total processing time. This validation also demonstrates the fulfillment of **Req2** requirement, since JSON-LD allows the IoT EWS to be able to process data from different sensor types described with different semantics, thus, providing syntactic and semantic interoperability with sensor systems.

9.2 Semantic Translation Analysis

Two analysis were performed to assess the performance of the semantic translations between SAREF and SSN/SOSA: (1) we used a popular example of SSN 1.0, the representation of the *WM30 Wind Sensor*, to verify the completeness of the mappings introduced in Chapter 6; (2) we assessed the accuracy and efficiency of these translations by using the INTER-IoT-EWS case study to analyze (a) if the messages generated from each semantic translation mechanism (SPARQL and IPSM) represent the expected output; and (2) the impact of the total transaction time of each of these approaches.

9.2.1 WM30 Wind Sensor: W3C SSN/SOSA to ETSI SAREF

The W3C SSN Incubator Group created the *Vaisala WM30 wind sensor* represented with SSN (version 1.0)¹⁵⁸, which describes the measurement capabilities, power supply and operating and survival properties based on the WM30 technical specification of wind direction and speed sensors. This example was initially reported in [309], with a precise description of the WM30 sensor. We updated these axioms with SSN/SOSA (version 2.0), shown in Code snippet 9.2. The axioms represent the Vaisala WM30 as a *sosa:Sensor* (a *ssn:System*), composed by (*ssn:hasSubSystem*) WM30 particular sensors for wind direction and wind speed, being able to measure (*sosa:observes*) wind direction (*WM30_WindDirection*) and speed (*WM30_WindSpeed*).

```

WM30:Vaisala_WM30  $\sqsubseteq$  sosa:Sensor
WM30:Vaisala_WM30  $\sqsubseteq$ 
   $\exists$  ssn:hasSubSystem WM30:WM30_WindDirection
WM30:Vaisala_WM30  $\sqsubseteq$ 
   $\exists$  ssn:hasSubSystem WM30:WM30_WindSpeed
WM30:Vaisala_WM30  $\sqsubseteq$ 

```

¹⁵⁸ <https://www.w3.org/2005/Incubator/ssn/ssnx/meteo/>

$\exists \text{ sosa:observes WM30:WindDirection}$

$\text{WM30:Vaisala WM30} \subseteq$

$\exists \text{ sosa:observes WM30:WindSpeed}$

Code snippet 9.2. Axioms representing WM30 wind sensor with SSN/SOSA

This validation simulates the unidirectional translation from WM30 ontology (SSN/SOSA) to SAREF. Therefore, the translation results must represent the WM30 wind sensor according to SAREF ontology, keeping as much as possible the semantics of the original. The goal of this validation is to check the *semantic completeness* of the SAREF ontology according to the SSN/SOSA semantics used in the WM30 scenario.

Table 9.1. Competency questions to be answered by the translated ontology (SAREF)

ID	Textual description
CQ01	What are the characteristics of the WM30 sensor?
CQ02	What is the composition of this sensor, i.e., what are the parts of WM30?
CQ03	What measurement properties this sensor performs?
CQ04	What are the accuracy, delay distance, starting threshold and damping ratio of wind direction sensors?
CQ05	What measurement range constraints differentiate these types of wind direction sensors?

We used competency questions to measure this semantic completeness, where each question was answered by navigating the elements of both ontologies. The competency questions are listed in Table 9.1. The answers are compared to verify whether the semantics are maintained after the simulation. Intentionally, the competency questions were conceived according to the expressiveness of the WM30 original ontology, targeting its main elements, as the different measurement capabilities described in the technical specifications¹⁵⁹, illustrated in Figure 9.3 (top). For example, the accuracy of the WM30 wind speed sensor (within a range from 0.4 to 60 m/s) is ± 0.3 m/s for wind speed lower than 10 m/s and $\pm 2\%$ of variance for wind speed higher than 10 m/s. The formalization of the WM30 capabilities with SSN is illustrated in Figure 9.3 (bottom).

First we answered each question based on the original ontology instance (in SSN) and, then, we answered the same question based on the generated ontology instance (SAREF). Second, we compare whether the answer has similar semantics, i.e., if the semantics are lost or maintained. The execution of the mappings was simulated, resulting in the WM30 ontology represented with SAREF, which is available online¹⁶⁰. The answers of the competency questions were produced by navigating the generated ontology with support of an ontology editor.

¹⁵⁹<http://www.vaisala.fi/Vaisala%20Documents/Brochures%20and%20Datasheets/WM30-Datasheet-B210384EN-B-LoRes.pdf>

¹⁶⁰ <https://github.com/jonimoreira/SSN-SAREF>

Figure 9.3.
WM30 device
capabilities (top)
and formalization
(bottom)

Wind speed		General	
Measurement range	0.5 ... 60 m/s	Supply voltage	3 ... 15 VDC
Starting threshold	< 0.4 m/s	Electrical connections	5-pin male with 12 mm threads
Distance constant	2 m	Recommended connector at cable end	BINDER 99 1436 814 05
Transducer output	1 Hz ~ 0.7 m/s	Operating temperature	-40 ... +55 °C (-40 ... +131 °F)
Accuracy (within range 0.4 ... 60 m/s)		Storage temperature	-60 ... +65 °C (-76 ... +149 °F)
wind speed < 10 m/s	± 0.3 m/s	Material	
wind speed > 10 m/s	± 2%	housing	AlMgSi, gray anodized
Characteristic transfer function	$U = -0.24 + 0.699 \times F$	cups	PA, reinforced with carbon fibre, black
(where U = wind speed [m/s], F = output frequency [Hz])		vane	PA, reinforced with fiberglass, white
Wind direction		Dimensions	265 (h) × 360 (Ø) mm
Measurement range		Weight	360 g
WMS301 with 1-wiper potentiometer	0 ... 355°	Accessories	
WMS302 with 2-wiper potentiometer	0 ... 360°	Mounting adapter	WMS30KIT
Starting threshold	< 1.0 m/s	Sensor connector	19370
Damping ratio	0.3	Sensor connector and cable, 10 m	19904
Overshoot ratio	0.4	Test compliance	
Delay distance	0.6 m	Wind tunnel tests	ASTM standard method D5366-93 ASTM standard method D5096-90
Accuracy	better than ±3°		

```

ssn:forProperty only WM30:WindDirection
ssn:hasMeasurementProperty some (WM30:DampingRatio
and (ssn:hasValue some (DUL:Amount
and (DUL:hasRegionDataValue value 0.3)))
ssn:hasMeasurementProperty some (WM30:DelayDistance
and (ssn:hasValue some (DUL:Amount
and (WM30:unitOfMeasure some {unit:metre}))
and (DUL:hasRegionDataValue value 0.6)))
ssn:hasMeasurementProperty some (WM30:OverShootRatio
and (ssn:hasValue some (DUL:Amount
and (DUL:hasRegionDataValue value 0.4)))
ssn:hasMeasurementProperty some (WM30:StartingThreshold
and (ssn:hasValue some (WM30:ValueRange
and (WM30:unitOfMeasure some {unit:metrePerSecond}))
and (WM30:hasRangeMaxValueExclusive value 1.0)))
ssn:hasMeasurementProperty some (ssn:Accuracy
and (ssn:hasValue some (WM30:ValueRange
and (WM30:unitOfMeasure some {unit:degreeUnitOfAngle}))
and (WM30:hasRangeMaxValueExclusive value "3.0"^^xsd:float)
and (WM30:hasRangeMinValueExclusive value "-3.0"^^xsd:float)))
ssn:ofFeature only WM30:WindFeature

```

Each competency question is responded as follows:

CQ01. In the original ontology, the main characteristics of the WM30 sensor can be extracted by navigating the *WM30:Vaisala_WM30* element, which is a *sosa:Sensor*, which inherits the properties from *ssn:System* and inherits the composition pattern. The sensor is composed by two sensors (*WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed*), one for wind direction and another for wind speed, both *sosa:Sensor*. Moreover, WM30 sensor can measure (*sosa:observes*) the types (properties) *WM30:WindDirection* and *WM30:WindSpeed* (both *ssn:Property*). Figure 9.4 illustrates these properties. In the generated ontology instance (SAREF), according to M01, *WM30:Vaisala_WM30* element is represented as a *saref:Sensor*. A *saref:SensingFunction* is created and the *WM30:Vaisala_WM30* element linked to it through *saref:hasFunction* property. According to M02, the composition of the sensor *WM30:Vaisala_WM30* is derived

from the *ssn:hasSubSystem* properties, i.e., *WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed*. M03 produced the measurement properties of the sensor from the *sosa:observes* element, i.e., *saref:measuresProperty* to *WM30:WindDirection* and *WM30:WindSpeed*. Figure 9.4 illustrates the result *WM30:Vaisala_WM30* as a *saref:Sensor*. Therefore, by navigating to *WM30:Vaisala_WM30*, a *saref:Sensor*, it is possible to answer this competency question in the same way as in the original instance, i.e., the semantics is completely maintained.

CQ02. In the original ontology, the composition of WM30 sensor can be extracted by navigating from the *WM30:Vaisala_WM30* element to the *ssn:hasSubSystem* properties, i.e., *WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed* (*sosa:Sensor*). The WM30 wind direction sensor (*WM30:WM30_WindDirection*) can have one wiper (*WMS301*) or two (*WMS302*). A corresponding structure is generated in SAREF, resulted from M02, having *WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed* created as *saref:Device*, linked through the object property *saref:consistsOf*. It is possible to obtain the same structure of *WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed*, including the specialization to one or two wipers, by navigating from *WM30:Vaisala_WM30* (*saref:Sensor*) through *saref:consistsOf*. This competency question is answered in the same way as in the original instance (semantics maintained).

CQ03. In the original ontology, the measurement properties of *Vaisala WM30* sensor can be extracted by navigating from the *WM30:Vaisala_WM30* element through the *sosa:observes* properties, i.e., *WM30:WindDirection* and *WM30:WindSpeed*, both *ssn:Property*. WM30 original example also uses an ontology of Quantity Kinds¹⁶¹ through the element *qu:QuantityKind* as *ssn:Property*. This element provides a taxonomy of quality dimensions, making use of *dim:Angle* for *WM30:WindDirection* and *dim:VelocityOrSpeed* for *WM30:WindSpeed*. The same structure is generated in SAREF, resulted from M03, as *saref:Property*. Therefore, similarly to CQ01 and CQ02, semantics is completely maintained.

CQ04. The specification of the WMS30 wind direction sensor describes the accuracy, delay distance, starting threshold and damping ratio of this sensor. For example, accuracy can vary from -3.0 to 3.0 degrees, while the delay distance is 0.6 meters and the starting threshold is lower than 1.0 m/s, represented in the axioms of Figure 9.3. In the original ontology, the measurement capabilities of each wind direction and speed components of Vaisala WM30 sensor can be extracted by navigating from the *WM30:Vaisala_WM30* element through the *ssn:hasSubSystem* properties, i.e., *WM30:WM30_WindDirection* and *WM30:WM30_WindSpeed*, both *sosa:Sensor*, as described in CQ02. The WM30 wind direction sensor with one wiper (*WMS301*) has measurement capability a

¹⁶¹ <http://purl.oclc.org/NET/ssnx/qu>

WM30:WM30_WindDirection_MeasurementCapability_WMS301 (as *WM_S302*). A *WM30_WindDirection_MeasurementCapability_WMS301* is a *WM30_WindDirection_MeasurementCapability*, which describes the ranges supported (restrictions) for accuracy, delay distance, starting threshold and damping ratio. *ssn-system:SystemCapability* is a *ssn:Property*.

Regarding the accuracy, SSN provides naively the *ssn-system:Accuracy* element which represents the accuracy of all involved sensors, extracted with the SPARQL query:

```
SELECT * WHERE { ?a ?b ssn:Accuracy. }
```

In the generated ontology (SAREF), these measurement capabilities were lost because there is no similar structure of *ssn-system:SystemCapability* in SAREF, thus, no mappings were added to consider the *ssn-system:hasSystemCapability* object property of *sosa:Sensor*. This question could not be responded with the generated ontology (semantics was lost).

CQ05. In the original ontology, the measurement range constraints differentiating 301 and 302 wind direction sensors can be extracted by analyzing the restrictions of *WM30:WM30_WindDirection_MeasurementCapability_WMS301* and *WM30:WM30_WindDirection_MeasurementCapability_WMS302* regarding the object property *ssn-system:hasSystemProperty*. The first restricts the measurement range from 0 to 355 degrees, while the second restricts from 0 to 360 degrees. This question could not be answered with the generated ontology because of the same reason described in CQ04, i.e., the absence of *ssn-system:SystemCapability* in SAREF and no mappings considering the *ssn-system:hasSystemCapability* object property.

The validation demonstrated that the *semantic completeness* of SAREF to SSN/SOSA translations is 60%, i.e., from 5 competency questions only 3 (60%) could be answered with the mappings introduced in this thesis. The main issue identified is the lack of a native element in SAREF to describe the measurement (system) capabilities of a sensor, which SSN/SOSA enables through the *ssn-system:hasSystemCapability* object property. To cope with this issue we suggest that a new mapping is created to align the structure from SSN/SOSA, i.e., reuse the object property *ssn-system:hasSystemCapability* applied to *saref:Sensor* with the restriction of only *ssn-system:SystemCapability*. In addition, the mapping must consider to align both *ssn-system:hasSystemCapability* and *ssn-system:hasSystemProperty* as (is-a) *saref:Property*.

Figure 9.4.
Composition
pattern: SAREF
on top and
SSN/SOSA on
bottom

●	saref:WindSensor
⊖	saref:consistsOf some WM30:WM30_WindDirection
⊖	saref:consistsOf some WM30:WM30_WindSpeed
⊖	saref:hasFunction some WM30:WindSensingFunction
⊖	saref:measuresProperty some WM30:WindDirection
⊖	saref:measuresProperty some WM30:WindSpeed
<hr/>	
●	ssn:SensingDevice
⊖	ssn:hasSubSystem some WM30:WM30_WindDirection
⊖	ssn:hasSubSystem some WM30:WM30_WindSpeed
⊖	ssn:observes some WM30:WindDirection
⊖	ssn:observes some WM30:WindSpeed

A practical issue when mapping from SSN/SOSA to SAREF is to consider extending the taxonomy of sensor types by creating a new element when the type does not exist in SAREF. For example, smoke and temperature sensors are classified as *saref:SmokeSensor* and *saref:TemperatureSensor*, respectively, having function (*saref:hasFunction*) and measure property (*saref:measuresProperty*), linking the type of the sensor with its functions and the type of property it measures (*saref:Smoke* and *saref:Temperature*). Thus, in our example, the appropriate implementation for the wind sensor is to create the subclass *saref:WindSensor*, with function sensing and measuring the properties of wind direction and wind speed, which is guaranteed by M01. A small issue was identified in WM30 ontology universal quantifier on *ssn:forProperty*, which is incorrect.

9.2.2 INTER-IoT-EWS: ETSI SAREF to W3C SSN/SOSA

Semantic translations between SAREF and SSN/SOSA for the INTER-IoT-EWS case study have been discussed in Chapter 7, which describes how *MyDriving-LD* mobile app provides data enriched with SAREF4health and logistics ontologies and the EWS *Input Handler* is responsible to execute the semantic translations, either through SPARQL or using the IPSM approach. The goal of this validation is to measure the translation accuracy, efficiency and maintainability, following the validation approach similar to [58]. However, different from this approach, we only considered as input test data the messages generated by the *MyDriving-LD* mobile app during the functional tests, both in the campus and in the port area. This means that the input test data only considered describable semantic messages.

We also only considered three types of result cases: (a) semantic error; (b) semantic distorted and (c) semantically equivalent. Therefore, it does not include the cases of untranslatable and syntax errors because the *MyDriving-LD* generates data as JSON-LD natively and the *Input Handler* relies on well-formed RDF (serialized as JSON-LD), through the use of a parser from the *dotNetRDF* library. During the functional tests, no parser errors were found in the semantic translation

method, thus, this means that all messages were syntactically correct. The formula to calculate the overall matching results was the following:

$$\frac{\text{No. of matched to expected translation output}}{\text{No. of test messages}}$$

Each input message M_{SAREF} , was translated to SSN/SOSA by using the transformation function $T_{\text{SAREF} \rightarrow \text{SSN/SOSA}}$. Then, we developed the transformation function $T_{\text{SSN/SOSA} \rightarrow \text{SAREF}}$ in a validation tool (the UI prototype described in Chapter 7), enabling the inverse transformation (from SSN/SOSA to SAREF), using the mappings presented in Chapter 6 for the opposite direction. The transformation $T_{\text{SSN/SOSA} \rightarrow \text{SAREF}}$ was executed on each $M_{\text{SSN/SOSA}}$ message, generating M'_{SAREF} message. Then, the validation tool compared M'_{SAREF} with M_{SAREF} by checking whether they have the same semantics.

Code snippet 9.2. SPARQL query for M03 and M04 mappings (SAREF)

```
sparqlQuery = @"
PREFIX saref: <https://w3id.org/saref#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?sensor ?measurement ?measValue ?measUnit ?measProp ?measTime ?sensorLabel
WHERE
{
    ?sensor saref:makesMeasurement ?measurement.
    ?measurement saref:hasValue ?measValue.
    ?measurement saref:hasTimestamp ?measTime.
    ?measurement saref:isMeasuredIn ?measUnit.
    ?measurement saref:relatesToProperty ?measProp.
    ?sensor rdfs:label ?sensorLabel.
}
ORDER BY ?measTime

";
```

This comparison was performed by first querying M_{SAREF} and loading in memory the main data representations that are related to the mappings. For example, the SPARQL query illustrated in Code snippet 9.2 selects the sensor measurements along with their measurement unity and property, as well as the value and timestamp, thus, to all predicates related to M03 and M04 mappings. The very same query is applied to M'_{SAREF} , enabling the comparison of the value used in each predicate. The output classification was performed automatically, where a semantic error was established if no data matched, while semantically equivalent represented the case that all data matched, and semantic distortion was established otherwise.

The results showed that, from 90 input messages, none messages were classified as semantic error nor semantic distortion, thus, providing 100% accuracy.

Although this result is quite expressive, we should also take into account that the accuracy measured regards only the predicates used in the mappings, lacking a number of SAREF predicates. For example, predicates as *saref:hasManufacturer*, *saref:hasTypicalConsumption* and *saref:accomplishes* were lost after the consecutive (round trip) execution of the semantic translations. However, if the semantic translation mechanism considers to simply include into the output message the predicates that do not participate on the mapping rules, then these predicates would remain after the round trip translations. This is the case of IPSM, which requires that the user specifies when a predicate should not be propagated to the output. This approach should be avoided because by propagating the source predicates to the target, the restrictions from the source are indirectly brought to the target, which can compromise the consistency of the target message.

Concerning the data size overhead produced by the semantic translations, i.e., the size difference between M_{SAREF} and $M_{\text{SSN/SOSA}}$, we found out that the $M_{\text{SSN/SOSA}}$ (around 300Kb) is ten times bigger than M_{SAREF} (around 30Kb) when IPSM is used. This difference was expected because the mappings did not include the time series predicate developed within SAREF4health context and IPSM copied the time series structure from the original message, but serializes the JSON-LD as flattened data structure, which adds an identity code to each specific measurement of the time series. This reflects exactly the time series problem tackled by SAREF4health. When using SPARQL, the time series data are not included in the SSN/SOSA output message, thus, a smaller message is generated.

Translation efficiency was measured according to the translation delay, i.e., the total time to translate by the mechanism. The messages of one test case from the functional tests, e.g., UC01_ST01_05 (strong collision), were used as input data. Then, the UI prototype simulated the execution of the test case two times (separately), one configuration for the SPARQL approach and for IPSM. The threat validity for this comparison regards network issues (noises) during each simulation. For example, if the Internet throughput is much different, the propagation time can influence the results. To tackle this issue, we performed the tests 100 consecutive times for each mechanism, computing the average and removing the anomalies, similar to the performance comparison of JSON-and JSON in Section 9.1.

Figure 9.5 shows how SPARQL compares with IPSM in terms of the total transaction time. As expected, the total transaction time for the “raw” SPARQL approach is lower than IPSM. The X axis represents the execution number regarding the efficiency test case, while the Y axis presents the average translation time in milliseconds. Each efficiency test case execution was performed in a different date (varying the time of the day) and each scenario was executed 100 times for each setup, computing the average of message translation time.

Figure 9.5.
Comparison of
SPARQL and
IPSM: transaction
time

The SPARQL approach is faster because it is simpler, having queries executed directly on the input data by the *Input Handler*, while the IPSM approach requires a REST post method for each message. However, IPSM is a more elaborated solution that, internally, also executes the transformation of the alignment definition from its language to a set of SPARQL queries, where each query is translated from a cell defined in the alignment. Therefore, while the first approach benefits from a manual query tuning, making the SPARQL query more efficient, the IPSM approach automatically generates the SPARQL queries for each message with lower efficiency.

The results of this comparison resembles the traditional approaches to implement SQL queries, where manually written SQL statements lead to more efficient queries if compared to queries that are automatically generated. On the other hand, the advantage of automatic generation is the better code maintainability, i.e., it is easier to maintain the queries over the time since they follow the patterns generated from the code generation transformation rules. Therefore, coding the semantic translations manually, such as the SPARQL approach, can lead to poor maintainability, but higher query efficiency. On the other hand, the IPSM model-driven approach abstracts the SPARQL code through its mapping language, enabling non-SPARQL specialists to create and change the semantic translations. These approaches should be selected according to the non-functional requirements and restrictions of the solution. For example, if the solution requires that domain experts should be able to create the semantic translations without requiring much SPARQL expertise, then the IPSM approach

is recommended. Both approaches are appropriated for the INTER-IoT-EWS solution, since maintainability and query efficiency were not part of the solution requirements.

The validations discussed in this section showed that by performing an ontological analysis on existing IoT standards (SSN/SOSA and SAREF), SEMIoTICS fulfils the **Req1** requirement, since this approach used the extension of the EWS core ontology with SSN/SOSA alignments. Furthermore, this validation demonstrated that SEMIoTICS fulfils the **Req2** requirement, allowing semantic interoperability between two sensor systems that use different IoT ontology standards according to the EWS requirements. The semantic translations efficiency achieved through the raw SPARQL approach showed that SEMIoTICS is able to address processing time restrictions of safety-critical applications, thus, SEMIoTICS fulfils **Req4** requirement.

The completeness of the translations can impact the semantic interoperability of IoT solutions that may use it, requiring to analyze the ontology elements that should be used by both source and target ontologies. The creation of new mappings is necessary when such elements are required. Usually, the mapping specifications reveal the possible gaps in the involved standards. For example, we noticed that SSN/SOSA does not provide a predicate to represent a device manufacturer, which is represented by *saref:hasManufacturer*. On the other hand, we noticed that SAREF does not provide a predicate to describe the device capability, which is represented by *ssn-system:SystemCapability*.

9.3 Performance Analysis

We analyzed the SEMIoTICS performance by measuring the total transaction time for the whole EWS dataflow, from data acquisition of the measurement (through the *Input Handler*), which is responsible for triggering a situation, to information dissemination (through the *Output Handler*), when the emergency notification is issued. The validation methodology is similar to the *Semantic IoT EWS* approach for testing scalability and resilience, in addition to an analysis of data storage and query.

The analysis has been performed in three steps: (1) we assessed the upstream data acquisition and situation identification, measuring the total transaction time from the *Input Handler* to the *Situation Identification Manager* for each test setup; (2) we assessed the data storage and query performed by the *Context Data Manager*, discussing the pros and cons of using a document-based NOSQL database against a triplestore graph database; (3) we assessed the multi-brokering multi-cluster approach by discussing how an industry-oriented cloud infrastructure, such as *Azure*, is able to multiply the throughput of 700 msg/sec, which is the threshold that the Semantic IoT EWS approach achieved in the TRIDEC project [10].

9.3.1 Response Time

We performed similar testes of the first step of the *Semantic IoT EWS* validation approach of “upstream lightweight sensor data and metadata acquisition and exchange” [14]. Therefore, we monitored sensor data acquired through the *Input Handler*, the execution of semantic translations followed by the execution of the *Context Data Manager* up to the situation detection performed by the *Situation Identification Manager*. Therefore, we measured the total transaction time from the upstraem data acquisition to the downstream information dissemination.

We observed that the semantic translations mechanism has the biggest effect on the total transaction time. This is an expected behavior since the main data processing activity resides on the semantic and syntactical translations. The difference (performance overhead) is minimal, less than 3% of the total time, and only reflects the propagation time between the *Input Handler* and the two REST microservices. Therefore, we can conclude that only 3% from the total response time of a situation triggered by the IoT EWS is spent by the translation mechanism, which is one of the main differential of the SEMIoTICS approach when compared to the *Semantic IoT EWS* approach. So, the performance overhead of using SEMIoTICS is quite small when using semantic translations.

9.3.2 Storing and Querying Data

In this validation we consider the transaction time and memory consumption of the the resources involved in INTER-IoT-EWS deployment, focusing on data storage and query performance of the *Context Data Manager* and the CEP patterns identification performed by the *Situation Identification Manager*. The *Context Data Manager* stores data in both *MongoDB*, for long-time storage of JSON-LD messages (historical raw data), and *Virtuoso*, to support querying with SPARQL. We have chosen to use both of them because initial tests with JSON-LD documents stored in *MongoDB*, querying data with *JQuery*, showed to be complex and inefficient, if compared to semantic querying capabilities (SPARQL), especially when the JSON-LD is flattened. Since data are already formatted as RDF (JSON-LD), storing data in a *Virtuoso* server allows the use of SPARQL. In contrast with *MongoDB*, the triplestore was slower for data insertion but more efficient for semantic querying. Therefore, for long-time historical data storage purposes, such as backup for audit, *MongoDB* showed to be the best option. On the other hand, *Virtuoso* showed to be useful to perform data analytics with ontology inference, supporting the so called “cold path” flow of *Azure IoT* platform.

We could not test the integration of real-time with historical data for situation identification (**Req3** requirement) in realistic settings because the INTER-IoT-EWS solution did not require this behavior. In INTER-IoT-EWS, there are only situation types that requires integration of event occurrences within 20 seconds of

time window, at most. If a situation type requires, for example, to calculate a dynamic threshold for tachycardia based on the last year ECG data of the driver, then the *Situation Identification Manager* would require historical data to be accessed. In this case, a service from the *Situation Identification Manager* would constantly run in background, calculating the threshold and inserting these data into the NESPER working memory, so that an EPL statement would be able to identify the pattern. Therefore, although SEMIoTICS is prepared to provide this capability, we could not validate the fulfillment of **Req3** requirement.

9.3.3 Broker Performance

Concerning the efficiency of publishing semantically rich sensor data with a multi-broker approach of the *Semantic IoT EWS*, we conclude that the threshold of 700 msg/sec/unit, with message size of 6Kb, thus, 4.200 Kb/sec/unit, is easily reached with a commercial cloud infrastructure such as *Azure*. According to the *IoT Hub* quotas and throttling¹⁶², described in Table 9.2, the throughput of publishing sensor data (termed “device-to-cloud”) can reach 6.000 msg/sec/unit, with message size of 4Kb¹⁶³, thus, 24.000 Kb/sec/unit. Therefore, the *Azure* cloud infrastructure can scale up to five times the broker throughput of *Semantic IoT EWS* approach.

Furthermore, *Azure* allows a maximum of 50 brokers (*IoT Hub* instances) to be deployed per subscription, thus, enabling a multi-broker approach deployed either in a single cluster (an *Azure* “resource group” deployment) or multiple clusters, as in a federated cluster deployment. Therefore, an equivalent configuration of the *Semantic IoT EWS* multi-cluster deployment can be achieved, with two clusters and five brokers per cluster, and load balancing. In the multi-broker multi-cluster case, latency issues should be avoided. *Microsoft* recommends to deploy the cloud broker in the closest region to the devices. If there is the situation of a device moving to another region, the load balance should be configured with rules to switch the publisher target to the closest region. Some cloud vendors provide tools to perform latency-sensitive operations at both field and cloud gateways, such as the *Azure IoT Edge*.

The SEMIoTICS approach is even more efficient than the *Semantic IoT EWS* approach because the *Semantic IoT EWS* approach splits data and metadata, separating the sensor data as key-pair values, serialized as XML messages, from the semantics (metadata), which is incorporated only on the server side, at the cloud level. Therefore, SEMIoTICS is more efficient and enriches the semantics of sensor data at the field gateway level, which decouples the solution to the specific IoT platform and cloud infrastructures. However, this high efficiency has a financial

¹⁶² <https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-quotas-throttling>

¹⁶³ In *Azure*, the default message size is 4Kb but can be up to 256 Kb.

cost, which is the main drawback of the SEMIoTICS framework when the IoT EWS infrastructure maintenance has budget restrictions.

9.4 Cost Analysis

The maximum throughput rate of *Azure IoT Hub* is 24.000 Kb/sec/unit, as discussed in the previous section. To achieve this throughput, the *B3* or *S3 IoT Hub* subscription is required (see Table 9.2). The *B3* tier price (in 2018) is €421,65 per unit per month. This cost would be unfeasible for the INTER-IoT-EWS business model, which proposes that the solution is financed by the haulier company along with the employees' health insurance. In *Azure*, a similar throughput of the *Semantic IoT EWS* approach of 4.200 Kb/sec/unit can only be achieved with *B3* or *S3* instances because the levels immediately below *B3*, i.e., *B2* and *S2*, allow a maximum of 480 Kb/sec/unit throughput, costing €42,17 (unit/month).

Table 9.2. *Azure IoT Hub* subscription costs and their throttle limits¹⁶⁴

Subscription	Cost / month	Throttle (device-to-cloud)	Nr. of msgs/day/hub
<i>B1</i> and <i>S1</i>	€8,43 - €21,09	12 messages/sec/unit	400.000
<i>B2</i> and <i>S2</i>	€42,17 - €210,83	120 messages/sec/unit	6.000.000
<i>B3</i> and <i>S3</i>	€421,65 - €2.108,25	6.000 messages/sec/unit	300.000.000

A crucial characteristic that impacts the performance and costs is the operating time requirements, i.e., the period that the system will be used. The operational frequency of the solution is impacted by the *IoT Hub* tier. While *B2* tier allows a maximum of 6.000.000 messages per day (message size of 4Kb), the *B2* limits to 300.000.000 messages per day. In our test deployments we used the subscription tier *B1*, which limits to 400.000 messages per day, thus, 1.600.000 Kb/day (~1.5Gb/day), a throughput of 100 msg/sec/unit and costs €8,43 (unit/month). Considering that the *MyDriving-LD* app publishes health messages (~30Kb) and logistics messages (~6Kb) each 3 seconds, thus, 12 Kb/sec; and we estimate each driver using the solution for 8 hours/day, then each driver consumes ~42 Mb/h (337.5 Mb/day) and one broker *B1* tier would support up to four drivers. Thus, the *IoT Hub* cost for a haulier company would be €8,43 per driver per month. The total cost for the company should also consider the cloud infrastructure of other resources, such as application and database servers, as well as the cost to buy (and maintain) each ECG device.

Table 9.3. Cloud resources for INTER-IoT-EWS validation and other involved components

¹⁶⁴ <https://azure.microsoft.com/en-us/pricing/details/iot-hub/>

Component	Subscription	Cost	Observation
<i>ECG device (Shimmer3)</i>	NA	~500€ per unit at once	Maintenance cost should be added, i.e., need to replace malfunctioning devices
<i>IoT Hub</i>	<i>IoT Hub B1</i>	~8.4€ per month/unit	
<i>Input Data Handler REST</i>	App service plan B3	~95€ per month	This application server (B3) is available through a virtual machine D2 v3/D2s v3 in EU North region, with 3.5 GB memory and 200 total ACU
<i>Context Data Manager REST</i>	App service plan B3	0	
<i>Situation Identification Manager REST</i>	App service plan B3	~188€ per month	
<i>Situation Reaction Manager REST</i>	App service plan B3	0	
<i>Output Handler REST</i>	App service plan B3	0	
<i>MongoDB service</i>	DB service	~20€ per month	The estimated cost includes the scale up tier and SSD-backed storage
<i>Virtuoso service</i>	DB service	~20€ per month	
<i>MyDriving REST</i>	App service plan D1	~8€ per month	Required to support the basic functions of <i>MyDriving</i> – app microservices for trips management
<i>MyDriving DB (SQL Server)</i>	S1 DTUs - Single Standard	~8€ per month	Required to support the basic functions of <i>MyDriving</i> – SQL Server database of trips management
<i>4G Subscription (mobile internet)</i>	6,5Gb/month	~20€ per month	Considering 20 days/month transmitting data 8h/day

Table 9.3 describes the estimated costs of each component used in INTER-IoT-EWS validation. The application and database server deployment tiers were chosen according to the CPU and memory consumption (quotas) required to process 4 drivers 8h/day simultaneously. These resources reflect the worst-case scenario and can serve as a baseline if more drivers (devices) are required to be added to the solution. We conclude that the total monthly cost of the solution per driver is €84,75 for cloud infrastructure (divided by 4), €8,44 for IoT Hub and €20,00 for 4G internet, thus, €113,19/month. This is not a fixed cost, this cost was calculated when using the “pay-as-you-go” subscription type, which is a dynamic pricing schema based on the resources used. The price of the solution can be reduced by optimizing the resources, using a mixed approach with application service subscription B2. Since that application service subscription and the associated virtual machines represent more than 60% of the costs, in-house datacenter solutions might be an opportunity to reduce de costs.

9.5 Chapter Conclusions

The performance validation of of data acquisition using semantic technologies fullfils **Req2** and **Req4**, through the analysis of JSON-LD with different

ontologies. We compared the verbosity when using schemaless simple JSON messages with semantic rich data serialized as JSON-LD, assessing the total transaction time for IoT applications. The variables included the message size and data processing in two typical IoT scenarios. The results show that a small message (with 1 measurement) has the greater impact when porting from JSON to JSON-LD, representing 0,19% of the total transaction time. Therefore, migrating from JSON to JSON-LD should not impact the performance of most IoT solutions, but may impact applications that have strict time response constraints. This impact decreases according to the increase of measurements in a message. In average, the message with 100 or more measurements has almost the same size serialized as simple JSON or rich JSON-LD. Therefore, the adoption of JSON-LD allowed the integration of sensors described with different semantics (**Req2**) and has a small impact on the performance of the solution (**Req4**).

Regarding the semantic translations between SSN/SOSA and SAREF, the validation shows that the *semantic completeness* of the mappings is 60%, i.e., the translations provide a coverage of 60% of semantics maintained from the original ontology (SSN/SOSA) to the generated (SAREF), for the WM30 wind sensor case. The main issue identified is the lack of system capabilities in SAREF to represent the collection of system properties of a sensor component. The representation of these characteristics in SAREF should be addressed in future versions. Another issue identified with the semantic translations validation is that the methodology proposed for the development of semantic translations (Chapter 6) may not be scalable for ontologies with thousands classes because the method is manual and time-consuming, although it produces high quality ontology alignments. Future work include balancing this method with an automated approach. This validation demonstrated that SEMIoTICS is able to address **Req1** and **Req2** requirements by adopting such translations, which are leveraged by the application of a well-founded ontology aligned to SSN/SOSA (SEMIoTICS core ontology) and the executions of mappings with SAREF4health extension.

The performance analysis showed that **Req4** is addressed by the framework. The analysis covers the total transaction time for the whole EWS dataflow, scalability, resilience and data storing and querying. The *Semantic IoT EWS* approach was used as baseline, showing how message throughputs can be leveraged by the adoption of an IoT platform along with cloud resources. In the case of INTER-IoT-EWS, with *Azure IoT Hub* as a cloud broker, the throughput can reach five times the throughput of *Semantic IoT EWS* approach. Furthermore, this validation showed that SEMIoTICS can address the scalability requirement (**Req5**) because of the RESTful microservice architecture along with the publish/subscribe multi-broker deployed in the cloud.

The analysis of data acquisition showed that the translations mechanism plays the major role in the total transaction time, having both SPARQL and IPSM

approaches feasible for the INTER-IoT-EWS requirements. Therefore, the validation showed that SEMIoTICS is appropriate for addressing data acquisition with semantic integration (**Req2** requirement).

Achieving such an efficient and scalable IoT EWS has a high cost, which may be unfeasible for some business models. For example, in the case of INTER-IoT-EWS, if the maximum throughput is required (24.000 Kb/sec/unit) then the solution would cost more than €1,000 per driver per month, considering the operating time of 8h/day. However, for the INTER-IoT-EWS solution, the deployed resources require less throughput (4.200 Kb/sec/unit) for message brokering and, along with other cloud resources (e.g., application server), can provide an adequate environment for 4 drivers simultaneously using the solution during 8h/day. The estimated cost for this setup, in a “pay-as-you-go” style.

The lack of test cases for the integration of real-time with historical data for situation identification is the main limitation of SEMIoTICS validation, affecting the validation of **Req3** requirement. The framework covers a solution to address this requirement, through a service of the *Situation Identification Manager* that is able to fetch and aggregate historical data into the CEP working memory (described in Chapter 6), but it could not be tested.

Conclusions

This chapter concludes this work, presenting the lessons learned, contributions, limitations and recommendation for future work. Section 10.1 outlines this thesis, discussing the lessons learned when tackling each challenge described in Chapter 4. Section 10.2 elaborates on the research contributions of this thesis, describing the importance of SEMIoTICS framework for interoperable smart emergency services with IoT EWSs. Finally, Section 10.3 discusses the limitations and future work.

10.1 Challenges and Lessons Learned

This thesis follows the Design Science Methodology for the engineering of interoperable IoT EWSs, applying the research design cycle of (1) problem investigation, (2) treatment design and (3) validation. Problem investigation is described in Chapters 2, 3 and 4. Treatment design is described in Chapters 5 and 6. Treatment validation is described in Chapters 7, 8 and 9.

Chapter 2 motivates the research on interoperability for DRR and the role of EWSs, describing approaches to improve the semantic interoperability of information systems, in particular, the situation-aware applications. Chapter 3 describes the concept of IoT platform, presenting its enabling technologies and how semantic technologies are used to improve the interoperability among IoT platforms. Chapter 4 describes a systematic literature review about interoperability of EWSs, which was leveraged by the background knowledge, detailing the challenges along with the requirements to be addressed by this thesis.

Chapter 5 introduces the SEMIoTICS framework, describing how the reference architecture components interact, the ontology-driven MDE methodology adopted and the role of the FAIR data principles, illustrating the framework usage through an example case. Chapter 6 describes the improvements performed in

each component of the SEMIoTICS framework, namely the semantic translations for the IoT, the situation foundations for context modelling and the situation modelling for IoT EWS.

Chapter 7 presents the first part of SEMIoTICS validation, describing the INTER-IoT-EWS solution, discussing its functional tests. Chapter 8 presents the second part of the validation, describing how a standardized ontology (ETSI SAREF) was extended to support the requirements of the IoT EWS developed for INTER-IoT-EWS with the SEMIoTICS methodology. Finally, Chapter 9 presents the last part of the SEMIoTICS validation, describing the analysis of the impact of semantic data on the overall performance of an IoT EWS developed with SEMIoTICS.

The SEMIoTICS framework addresses the challenges and requirements as follows:

C01: Semantic Integration of a variety of data sources

(Req1) Enable the development of an EWS core context model as a well-founded ontology, taking into account existing data representations and possible restrictions from the EWS requirements.

- The SEMIoTICS methodology applies ODCM for the design of context models as well-founded ontologies, while considering RDF best practices for implementation, which is reinforced by the FAIR principle *I1* (use formal language for knowledge representation). Furthermore, reusing data models is addressed by following *R1.3* (adopt domain-relevant standards) and *I3* (adopt and/or align with semantic standards).
- We showed that the harmonization of terms related to situation awareness in UFO can leverage the ODCM practice. We experienced that deciding whether to design a concept in the foundational layer or in the core layer or in domain-specific layers is a challenging task. For example, in the SAREF extension for e-Health (Chapter 8), the *Time Series* concept may suit better to the core ontology (SAREF) rather than the domain-specific (SAREF4health).
- The development of the SAREF4health ontology followed the SEMIoTICS guidelines, which supported the development of a well-founded, standard-oriented and efficient semantic model (RDF). The SAREF4health development showed that a trade-off between ontology quality and lightweight data serialization is a relevant aspect to deal with IoT restrictions, such as message payload size. The use of a reference ontology grounded in UFO theory (the UFO ECG) improved the conceptual modelling quality of SAREF4health because of its explicit semantic foundations, resulting in the improvement of the semantics of the RDF

implementation. Moreover, the reuse of standardized ontologies showed to be important because they usually provide the viewpoint of a domain from a community perspective, in particular the IoT, healthcare and emergency domains.

(Req2) Enable the development of an EWS data acquisition component to be able to process data from different sensor types described with different semantic models, i.e., provide syntactic and semantic interoperability.

- The data acquisition component of SEMIoTICS (*Input Handler*) uses a semantic translation mechanism, which allows the IoT EWS to integrate sensor data described with different semantics and, therefore, to improve semantic interoperability. We showed that this mechanism can be implemented with standard semantic querying technology (SPARQL) or with existing translation systems, such as IPSM, which provides a language to describe the mappings. The main tradeoff between these two approaches is the choice of performance tuning or maintainability, albeit both approaches are suitable for improved semantic interoperability.
- The development of semantic translations between ETSI SAREF and W3C SSN/SOSA enabled the integration of sensor data that are described with these standards, improving the semantic interoperability for the IoT domain. However, analyzing the different representations for same concepts with detailed aspects showed to be a difficult and time-consuming task. We conclude that the main different representations of these ontologies are merely a matter of distinct ways to represent the same conceptualization, where SAREF seems to be more straightforward to understand while SSN/SOSA seems to be more complex. This is a characteristic sought by the SAREF task force when creating the ontology along with industry partners, emphasizing on the practical point of view from industry initiatives (e.g., proprietary data models of existing solutions).
- The development of such translations illustrated the application of the SEMIoTICS guidelines for ontology alignments among domain-specific ontologies. The validation showed that this approach addresses the functional requirements of an IoT EWS, i.e., information elements required for the INTER-IoT-EWS use case (Chapter 7), and non-functional requirements, i.e., completeness and total time processing performance (Chapter 9).
- Having the ontologies implemented as RDF and serialized as JSON-LD addresses syntactic translations and, therefore, improves syntactic interoperability.

(Req3) Enable a data fusion mechanism for EWS, allowing it to combine incoming real-time data with historical data, both for situation identification and for downstream information dissemination.

- The *Context Data Manager* fulfils this requirement because it allows the EWS to store data in different databases, such as triplestores and document-oriented databases. In particular, triplestores are used to store semantic data, including aggregated data, which are inserted in the working memory of the CEP engine, under the *Situation Identification Manager*. Therefore, data fusion of historical data with real-time data is enabled by the integration of the *Context Data* and *Situation Identification Manager* components.

C02: Processing in time- and safety-critical applications

(Req4) Enable the EWS to provide adequate performance, in terms of total processing time, which depends on the specific safety-critical restrictions regarding minimal response time. If the EWS requirements explicitly restrict the minimum time thresholds, the EWS should be able to process the data within this threshold.

- The adoption of an IoT platform with cloud infrastructure supports the realization of this requirement. This is evidenced by the adoption of data acquisition with publish/subscribe brokers that provide high operation throttles, demonstrated through the performance validation of *Azure IoT Hub* (Chapter 9). Furthermore, the event-driven middleware underlying the brokers run high performative protocols for IoT (AMQP and MQTT) and situation identification with a CEP engine provides fast integration of real-time incoming data. The validation showed that, on one hand, the deployment and configuration of a single IoT platform component (e.g. *IoT Hub*) is quite straightforward, while the integration of all components to make the IoT solution fully functional (as intended) was complex, especially because of the distributed nature of an IoT solution that requires to deal with asynchronous events among the components.
- The adoption of JSON-LD instead of XML also supported addressing this requirement, since JSON-LD yields lower verbosity than XML and allows the use of fast JSON processors. JSON-LD is a lightweight syntax and the impact on performance is minimal when migrating from IoT applications using JSON, as demonstrated by the empiric comparison study described in Chapter 9.
- The SEMIoTICS guidelines for ontology engineering also support the production of leaner JSON-LD messages, which was demonstrated

through SAREF4health (Chapter 8) development, considering the verbosity of messages generated by similar ontologies.

- Comprising syntactic and semantic translations performance (Chapter 9), two approaches (SPARQL and IPSM) were compared in terms of processing performance and maintainability, showing a tradeoff: while the SPARQL approach provides better performance, the IPSM approach provides better maintainability. The validation showed that both approaches could properly support the INTER-IoT-EWS use case.

(Req5) Enable the EWS to be scalable according to the data input volume and speed, i.e., be able to allocate resources according to the number of data sources, their input frequency and message size.

- Similar to **Req4**, this requirement was addressed by adopting IoT platform technologies. Brokers can scale horizontally by adding resources to a server and vertically (multi-cluster), which allows both data acquisition and downstream information dissemination to be scalable. The elasticity of operation throttles of IoT platforms, such as *Azure IoT*, also enables scalability w.r.t. the variations of input data frequency and message size. Furthermore, RESTful microservices handles efficiently a large number of requests at the same time, and this number can be increased by using mechanisms such as load balancing. Storage scalability of IoT EWS can be optimized by sharding database approaches.

(Req6) Enable a dynamic and adaptive mechanism to modify the EWS at runtime, minimizing the effect on the running instances.

- This requirement is fulfilled by the *Context Data Manager* component by means of RDF adoption, which provides extensibility capabilities for the context model and, therefore, allows the extension of the core ontology while the EWS is executing.
- The *Situation Identification Manager* component also fulfills this requirement, since the complex rules represented as *Situation Types* can be deployed at runtime with minimal impact, such as the query-based approach of SIMPLE (with EPL) and the rule-based approach of SCENE (with DRL). For example, we used EPL statements that could be inserted in CEP engines at runtime with minimal interference in the running instances.
- Finally, the *Situation Reaction Manager* component also fulfills this requirement because it employs workflow systems for managing emergency plans as business processes, which allows response messages to

be configured at runtime according to the targets information requirements.

C03: Data analysis for effective responses

(Req7) Offer modelling capabilities for the representation of temporal relations among different types of events and situations. In particular, it must enable the EWS to use interval relationships, such as *Allen's* operators, to relate events for situation identification.

- This requirement is fulfilled by using CEP engines that support pre-defined patterns of object relationships and temporal relations of events and other situations described as *Situation Models* for emergency detection rules. This capability was demonstrated by using complex temporal rules in the INTER-IoT-EWS use case, such as the *patterns* statement of EPL for consecutive events. Although the *stream analytics* approach of *Azure IoT* seems to be suitable for situation identification, it provides lower expressiveness when compared to the EPL. SML leverages on this approach by allowing the EWS designer to use temporal reasoning (*Allen's* operators among STs) during the specification phase.
- The INTER-IoT-EWS validation showed that the situation identification mechanism can be employed, to some extent, at device level. Therefore, a hybrid approach for situation identification should be employed considering data processing at both device and cloud levels. Situation identification processing in the IoT device level should be performed when data about the situation participants are available, such as vehicle collision detection based on the vehicle acceleration. For more complex situations that rely on external sources, such as a possible tuberculosis outbreak, the situation identification should be processed in the cloud.

(Req8) Enable the representation of complex rules over context data, such as multivariate functions (risk, odds, rate and prevalence) and temporal existential rules (sliding time windows).

- A CEP engine such as ESPER allows the use of aggregation functions for multivariate functions and sliding time windows, such as the EPL *timer:within* statement. This capability was demonstrated by the use of sliding time windows in the INTER-IoT-EWS use case. Similarly to **Req7**, SML also fulfills this requirement by allowing the EWS designer to use aggregated functions and temporal existential rules (SML mutability) during the specification phase.

(Req9) Enable the EWS to issue warnings, as semantic enriched messages, to multiple targets through multiple channels (e.g., broker, e-mail, SMS). In particular, enable the modification of the information requirements of the targets at runtime through an UI.

- This requirement is fulfilled by using RDF and JSON-LD in the *Output Handler* component. In particular, this is enforced by the semantic model of emergency alerts (EDXL-CAP). The INTER-IoT-EWS validation showed that warnings can be issued using semantic messages following the EDXL-CAP lexicon standard structure.
- The *Output Handler* enables message brokering to multiple targets, which can pre-configure the information elements that should be included in the warning messages, aligned to the EDXL-CAP elements. The INTER-IoT-EWS validation also demonstrated how an EWS developed with SEMIoTICS can issue warnings to multiple targets.

10.2 Main Contributions

Current proposals for IoT-based EWS only partially address the semantic integration of a variety of data sources along with processing in time-critical applications and data analysis for effective responses.

10.2.1 Holistic Approach for Semantic IoT EWS

The SEMIoTICS framework offers a holistic solution to address this problem by applying different modelling languages, ontologies and IoT technologies towards interoperability within and between IoT EWSs. The SEMIoTICS framework has been designed to be domain-agnostic, so it is applicable for diverse types of disaster. Furthermore, the framework follows the FAIR data principles and, therefore, is able to produce FAIR data. Our validation with the INTER-IoT-EWS use case highlighted this characteristic properly, by showing how to develop an IoT EWS that is able to integrate IoT solutions from two different domains (in health and logistics). The SEMIoTICS framework has a sound conceptual foundation, reflected by the languages to be used in the application specification phase, emphasizing the improvement of semantic interoperability from the conceptual modelling phase to the implementation phase of the EWS development lifecycle.

10.2.2 Improved IoT Semantic Interoperability

The SEMIoTICS validation showed that the trade-off between ontology quality and lightweight data serialization was crucial in the design of the SAREF4health ontology due to the performance requirement regarding the impact of

measurement time series in the message payload, i.e., the verbosity problem. Currently, SAREF4health is being used by the ETSI task force (STF566) that is developing the standardized extension of SAREF for e-Health and aging well (SAREF4ehaw). This activity includes more general use cases rather than specific ECG monitoring and involves more general requirements from different stakeholders. However, independent of whether SAREF4health will ever be fully incorporated in SAREF4ehaw, we believe that this work provides an important contribution towards a higher-quality FHIR ontology, rather than a simple RDF representation of the FHIR lexicon data model. This same comment is applicable to the EDXL-CAP ontology produced under the RESCUER project, which is only a direct RDF representation (semantic model).

The validation of the SEMIoTICS methodology, which is based on conceptual modelling and RDF implementation, also represents a contribution of this thesis. Furthermore, the mappings between SAREF and SSN/SOSA are a contribution of this thesis, which introduce fundamental translation rules to enable the semantic integration of IoT platforms relying on these ontologies, emphasizing their core elements, such as *Sensor*, *Device*, *Function*, *Measurement* and *Observation*. In terms of performance, the empirical study on migrating IoT solutions from JSON to JSON-LD is also a relevant contribution of this thesis, which showed the feasibility of transforming IoT solutions to semantic IoT solutions.

In addition, the foundational aspect of SEMIoTICS can also be considered as a contribution. The modelling languages adopted in SEMIoTICS are expressive enough to improve understanding of context data. This is enforced by adopting a mature foundational ontology and its ontological language (UFO/OntoUML), which was extended with concepts related to situations, such as *situoid*, *event type* and *infor*. Furthermore, the extension of *OntoEmerge* as the well-founded SEMIoTICS core ontology is a relevant contribution, in which we incorporated terms from multiple ontologies, leveraged by the ODCM practice with multiple-alignments to SSN/SOSA and to EDXL-CAP. We have shown that IoT EWS semantic interoperability can be improved with a clear distinction between the concepts of *situation* and *event*. This distinction is represented explicitly in EDXL messages as a result of our approach based on model transformations (MDE). Furthermore, the ontological analysis of EDXL was useful for the development of more accurate model transformations.

10.2.3 Improved Situation Identification for IoT EWS

The development of a new version of SML (2.0) is also a contribution of this thesis. We analyzed the gaps of the first version of SML, regarding both the IoT EWS requirements and the impact of the *Situoid* harmonization in UFO along with the relations to the ontological language OntoUML. The use of SML 2.0 was

exemplified with an epidemiological surveillance EWS for *Zika* outbreaks scenario. The SML concrete syntax indicates that the rules can be now better understood due to the increased expressivity of the language. Moreover, although we used CEP technology to support the SML redesign, SML 2.0 is independent of implementation technology and is domain-independent. Therefore, we gave emphasis to the description of SML semantics and the impact on the abstract and concrete syntaxes, grounded on the theories of situation-awareness and situation logics and a foundational ontology (UFO).

10.2.4 Interoperability Reference for Emergency Services

By achieving the research goals of this thesis through the SEMIoTICS framework, we claim to have advanced the state-of-the-art on semantic interoperability of emergency services research, grounding our research in two pillars: improvements in ontology engineering theory, and demonstration of the usefulness of this contribution in practical emergency scenarios where EWS interoperability is required. In addition, we recommend that the SEMIoTICS framework should be considered by the UN agenda for DRR, contributing to the “integrated and holistic approach to early warnings for multiple hazards and risks tailored to user needs across sectors”, as described by the Sendai framework [29].

10.3 Limitations and Future Work

Relevant limitations of this thesis include (1) the validation on using machine learning for situation identification, using social media for data acquisition and brokering, and lack of cybersecurity research; (2) open issues in SAREF4health and in the semantic translations between SSN/SOSA and SAREF; (3) lack of updating model transformations from SML 2.0 to EPL; (4) lack of formalization of situation foundations; and (5) the high costs of SEMIoTICS with commercial IoT platforms with longer operating times.

10.3.1 Machine Learning, Social Media and Cybersecurity

Although the SEMIoTICS methodology prescribes the use of machine learning for situation identification by considering explainable AI (Chapter 5), the validation lacked on demonstrating this capability. Similarly, the use of social media is also prescribed by SEMIoTICS, such as *Twitter* and *Facebook*, as data sources and channels for early warnings (a popular approach). However, we did not develop an IoT EWS integrated to social media. To achieve it, a semantic wrapper is required to execute semantic translations between the platforms’ API (data interfaces) and the core

ontology, playing the role of a proxy, similar to the semantic proxy developed for EUDAT B2Share data repository (Annex 1).

Although cybersecurity is a crucial aspect of an IoT EWS, this thesis did not emphasize the importance of end-to-end security for smart behavior. Therefore, this thesis does not address specific issues on securing messaging between embedded systems, as well as security issues of the related standards/ontologies. Rather than guiding specific cybersecurity methods and technologies, the SEMIoTICS framework relies on existing guidelines provided by the IoT platforms, especially the commercial ones like *Azure*. As described in Chapter 3, *Microsoft* provides IoT data compliance policies for privacy, security and ethics, and guidelines for the development of threat models. The SEMIoTICS framework also lacks a mechanism to cope with quality of information (QoI) at the network level, such as a *Grubbs'* test for outlier detection, and a statistical algorithm that can classify anomalous or invalid sensor values. These are also interesting topics for future work.

10.3.2 Semantic Interoperability for IoT Solutions

Regarding the semantic translations between SAREF and SSN/SOSA, although the focus was to address the main IoT elements, some relevant concepts were not mapped, such as *saref:Actuator* with *sosa:Actuator*. Therefore, further research to map the entire ontologies is necessary, at least covering the required elements of each ontology. Furthermore, the wind sensor example showed that only 60% of semantics are maintained from the original ontology (in SSN/SOSA) to the generated (in SAREF), which is a relevantly low semantic completeness. The main issue identified is the lack of system capabilities in SAREF to represent the collection of system properties of a sensor component, which should be addressed in future versions. We recommend to extend SAREF with the *ssn-system:hasSystemCapability* object property (used by *saref:Sensor*) and importing *ssn-system:SystemCapability* and *ssn-system:hasSystemProperty* (as *saref:Property*).

Another issue identified with the semantic translations validation is that the methodology proposed for the development of semantic translations (Chapter 6) may not be scalable for many TBox elements, being time-consuming, although this methodology can produce more semantically assertive ontology alignments. Future work to balance this quality improvement with a more automated approach should be performed. Furthermore, the comparison study between SPARQL approach and IPSM approach can be extended to other approaches, such as W3C *Shape Expressions* (ShEx)¹⁶⁵ and W3C *Shapes Constraint Language* (SHACL)¹⁶⁶.

¹⁶⁵ <https://www.w3.org/2001/sw/wiki/ShEx>

¹⁶⁶ <https://www.w3.org/TR/shacl/>

Regarding the SAREF4health ontology, future work includes a formal empirical validation to compare it with other ontologies, such as the HL7 FHIR RDF. Semantic translations between SAREF4health and FHIR RDF are crucial to achieve semantic interoperability for e-Health. A limitation of SAREF4health is that it still lacks on a concept to represent the physicians who administer an ECG recording session, which is present in other standards (e.g., in DICOM). To overcome this issue, we suggest including a class axiom to *ECG Recording Session* with the *dc:creator*, similarly to the approach with *dc:author*.

10.3.3 SML Model Transformations

Regarding SML, future work comprises an extended validation of SML to evaluate the completeness and performance of the transformations from SML to EPL. Although the SEMIoTICS framework prescribes the use of model transformations from SML to generate the CEP code, we have only reused the existing transformations of the SIMPLE approach, also considering the patterns adopted in the SCENE approach. Although SML 2.0, described in Chapter 6, includes aggregation functions and Boolean operators for composing comparative relations, the model transformations were not extended to support them. Furthermore, to enable SEMIoTICS to work with different CEP technologies at the implementation level, the model transformations may be integrated to other existing MDE approaches for CEP engines, such as the *Model4CEP*, and for conventional approaches such as *Decision Tables*. An ontology can be developed to describe rules to share the situation identification rules, such as it is done for the lexicon approach of *RuleML* [310].

10.3.4 Formalization of Situation Foundations

Concerning foundational ontologies and the harmonization of concepts related to situations in UFO along with the impact on OntoUML (Chapter 6), although we have elaborated on the reuse of axioms from the *Situoid* theory and their impact on the axioms of UFO-B, we did not completely formalized these axioms and ignored the impact on UFO consistency. Therefore, future work includes this formalization with first order logic and a formal representation of causation relationship between situation and event, which can be leveraged by the epidemiological causality model of *Bradford Hill Criteria*. These improvements should consider the ongoing approach for scene representations in UFO. Although UFO and DOLCE formed the conceptual basis of this thesis (UFO for SEMIoTICS core ontology and standardized ontologies based on DOLCE) we did not consider the integration or convergence of DOLCE and UFO. We recommend that these foundational ontologies should be combined for a single universal foundational ontology, or at least proper ontology alignments between these ontologies should be provided. Besides

ontology alignments, understanding ontologies is challenging because they are sometimes developed in different times with different perspectives, relying on a huge analytical effort. More research is required in order to systematize the analysis and reuse of so many existing ontologies.

10.3.5 Costs for IoT EWS with Commercial IoT Platform

One of the main limitations of SEMIoTICS is its potential high operational costs, which may be unfeasible in some business models. For example, if the INTER-IoT-EWS solution uses the maximum throughput (24.000 Kb/sec/unit), it would cost more than €1,000 per driver per month. A relevant limitation of the validation through the INTER-IoT-EWS is the lack of test cases for the integration of real-time with historical data for situation identification, so that **Req3** requirement could not be validated. Another threat for the validity of SEMIoTICS is the lack of test cases that cover changing the behavior at runtime with minimal effects on the running instances, i.e., to demonstrate **Req6**. Therefore, 7 from 9 requirements were successfully validated, and if we consider that all requirements have the same importance and complexity, the SEMIoTICS validation presents a coverage of 78% (approximately). However, since these requirements are simpler to address and have lower importance than others (e.g., **Req1** and **Req2**), we conclude that the SEMIoTICS validation is quite thorough and the SEMIoTICS framework presents satisfactory advances for the state-of-art.

Appendix A

Author Publications

The relevant publications from the author during the development of this thesis are listed in chronological order:

1. Ferreira, M. I. G. B., Moreira, J., Campos, M. L. M., Braga, B. F. B., Sales, T. P., de Cordeiro, K. F., & Borges, M. R. S. (2015). *OntoEmergePlan: variability of emergency plans supported by a domain ontology*. In *12th International Conference on Information Systems for Crisis Response and Management (ISCRAM 2015)*.
2. *Developing situation-aware applications for disaster management with a distributed rule-based platform*. In *Proceedings of the RuleML 2015 Challenge, Special Track on Rule-based Recommender Systems for the Web of Data, Special Industry Track, and RuleML 2015 Doctoral Consortium*. CEUR Workshop Proceedings; Vol. 1417.
3. Moreira, J., Ferreira Pires, L., van Sinderen, M. J., & Dockhorn Costa, P. (2015). *Towards ontology-driven situation-aware disaster management*. *Journal of Applied ontology*, 10(3-4), 339-353. DOI: 10.3233/AO-150155.
4. Moreira, J., van Sinderen, M. J., Ferreira Pires, L., & Dockhorn Costa, P. (2016). *Improving semantic interoperability of big data for epidemiological surveillance*. *Enterprise Interoperability IX: I-ESA proceedings*. (BD4EI workshop 2016).
5. Moreira, J.; Sales, T. P.; Guerson, J.; Braga, B. F. B.; Brasileiro, F.; Sobral, V., Menthor editor: *An ontology-driven conceptual modeling platform*. *9th International Conference on Formal Ontology in Information Systems (FOIS 2016) - Demonstration Track*, vol. 1660, 2016.
6. Poster & PhD symposium ISCRAM 2016: "A situation-aware framework for early warning systems".

7. Moreira, J. L., Ferreira Pires, L., van Sinderen, M. J., & Dockhorn Costa, P. (2017). *Ontology-Driven Conceptual Modeling for Early Warning Systems: Redesigning the Situation Modeling Language*. *MODELSWARD 2017: proceedings of the 5th international conference on Model-Driven Engineering and Software Development* (pp. 467-477). Sétubal: SCITEPRESS. DOI: 10.5220/0006208904670477.
8. Moreira, J. L., Daniele, L. M., Ferreira Pires, L., van Sinderen, M. J., Wasielewska, K., Szejeja, P., ... Paprzycki, M. (2017). *Towards IoT platforms' integration: Semantic Translations between W3C SSN and ETSI SAREF*. Paper presented at SEMANTiCS conference 2017, International Workshop on Semantic Interoperability and Standardization in the IoT (SIS-IoT) workshop, Amsterdam, Netherlands.
9. Poster in DI4R 2017: "B2SHARE-FAIR: Implementing B2SHARE as a FAIR Data Point".
10. Moreira, J.; Pires, L. F.; Sinderen, M. V.; Wieringa, R.; Singh, P.; Costa, P. D.; Llop, M., "Improving the semantic interoperability of IoT Early Warning Systems: the Port of Valencia use case". *Enterprise Interoperability IX: I-ESA proceedings*, 2018.
11. Moreira, J.; Pires, L. F.; Sinderen, M. V., "Semantic interoperability for the IoT: Analysis of JSON for Linked Data". *Enterprise Interoperability IX: I-ESA proceedings. BD4EI workshop*, 2018.
12. Moreira, J.; Pires, L. F.; Sinderen, M. V., "Semantic interoperability of Early Warning Systems: a systematic literature review". *Enterprise Interoperability IX: I-ESA proceedings. ICRIM workshop*, 2018.
13. Moreira, J. L., Ferreira Pires, L., van Sinderen, M. J., Daniele, L. M. (2018). *SAREF4health: IoT standard-based ontology-driven healthcare systems*. 10th International Conference on Formal Ontology in Information Systems (FOIS 2018). Best paper award.
14. Moreira, J. L., Ferreira Pires, L., van Sinderen, M. J. (2019). *SEMIoTICS: Semantic Model-Driven Development for IoT Interoperability of Emergency Services*. 16th International Conference on Information Systems for Crisis Response and Management (ISCRAM 2019).

Appendix B

Stakeholders in EWS interoperability

The list of main stakeholders is given in Table B.1 below.

Table B.1. Main involved stakeholders and their goals

Name	Type	Goal
UNISDR	International body	UN Office for Disaster Risk Reduction “to serve as the focal point in the United Nations system for the coordination of disaster reduction and to ensure synergies among the disaster reduction activities of the United Nations system and regional organizations and activities in socio-economic and humanitarian fields” (...) “facilitate the implementation of the International Strategy for Disaster Reduction (ISDR)”
WHO	International body	World Health Organisation “Our primary role is to direct and coordinate international health within the United Nations’ system.” Provides and maintains the ICD (versions 9, 10 and 11): “International Classification of Diseases is the foundation for the identification of health trends and statistics globally, and the international standard for reporting diseases and health conditions. It is the diagnostic classification standard for all clinical and research purposes. ICD defines the universe of diseases, disorders, injuries and other related health conditions, listed in a comprehensive, hierarchical fashion that allows for: - easy storage, retrieval and analysis of health information for evidenced-based decision-making; -sharing and comparing health information between hospitals, regions, settings and countries; and - data comparisons in the same location across different time periods.”
CDC	International body	Centers for Disease Control & Prevention “CDC works 24/7 to protect America from health, safety and security threats, both foreign and in the U.S. (...) CDC fights disease and supports communities and citizens to do the same. (...) As the nation’s health protection agency, CDC saves lives and protects people from health threats. (...) CDC conducts critical science and provides health information that protects our nation against expensive and dangerous health threats” (America)
ECDC	International body	European Centre for Disease Prevention & Control “EU agency with aim to strengthen Europe’s defences against infectious diseases. (...) ECDC’s mission is to identify, assess and communicate current and emerging threats to human health posed by infectious diseases.” (Europe)
OMG	Standards institute	Object Management Group (OMG) “international, open membership, not-for-profit technology standards consortium (...). OMG standards are driven by vendors, end-users, academic institutions and government agencies. OMG Task Forces develop enterprise integration standards for a wide range of technologies and an even wider range of industries.” Information Exchange Framework™ (IEF™) Information Exchange Packaging Policy Vocabulary™ (IEPPV™)

OASIS	Standards institute	“a nonprofit consortium that drives the development, convergence and adoption of open standards for the global information society.(...) produces worldwide standards for security, Internet of Things, cloud computing, energy, content technologies, emergency management, and other areas. OASIS open standards offer the potential to lower cost, stimulate innovation, grow global markets, and protect the right of free choice of technology.”Emergency Data Exchange Language (EDXL)
W3C	Standards institute	“international community where Member organizations, a full-time staff, and the public work together to develop Web standards. (...) The W3C mission is to lead the World Wide Web to its full potential by developing protocols and guidelines that ensure the long-term growth of the Web.” Emergency community (https://www.w3.org/community/emergency/)
OGC	Standards institute	“The OGC (Open Geospatial Consortium) is an international not for profit organization committed to making quality open standards for the global geospatial community. These standards are made through a consensus process and are freely available for anyone to use to improve sharing of the world's geospatial data. OGC standards are used in a wide variety of domains including Environment, Defense, Health, Agriculture, Meteorology, Sustainable Development and many more.” http://www.opengeospatial.org/ogc/markets-technologies/swe
CEN	Standards institute	CEN (European Committee for Standardization) “(...) is an association that brings together the National Standardization Bodies of 33 European countries. CEN is one of three European Standardization Organizations (together with CENELEC and ETSI) that have been officially recognized by the European Union and by the European Free Trade Association (EFTA) as being responsible for developing and defining voluntary standards at European level.” http://www.en13606.org/the-ceniso-en13606-standard
ETSI	Standards institute	“ETSI, the European Telecommunications Standards Institute, produces globally-applicable standards for Information and Communications Technologies (ICT), including fixed, mobile, radio, converged, broadcast and Internet technologies. Our standards enable the technologies on which business and society rely. For example, our standards for GSM™, DECT™, Smart Cards and electronic signatures have helped to revolutionize modern life all over the world.” TETRA and National Standards Organizations (NSOs) https://portal.etsi.org/TBSiteMap/NSO/Home.aspx
ISO	Standards institute	“International Organization for Standardization. We develop and publish International Standards.” “ISO/TC 184/SC 5 - Interoperability, integration, and architectures for enterprise systems and automation applications”
HL7	Standards institute	“a not-for-profit, ANSI-accredited standards developing organization dedicated to providing a comprehensive framework and related standards for the exchange, integration, sharing, and retrieval of electronic health information that supports clinical practice and the management, delivery and evaluation of health services. HL7 is supported by more than 1,600 members from over 50 countries, including 500+ corporate members representing healthcare providers, government stakeholders, payers, pharmaceutical companies, vendors/suppliers, and consulting firms.” Reference Information Model (RIM) CDA® R2 Implementation Guide: Emergency Medical Services; Patient Care Report Emergency Care committee Public Health and Emergency Response committee http://www.hl7.org/special/Committees/emergencycare/index.cfm
NIEM	Standards institute	National Information Exchange Model (NIEM) “a community-driven, standards-based approach to exchanging information.” “The NIEM Emergency Management (EM) domain works to improve decision-making by organizations that share information relating to emergency response and emergency management and to increase the ability to prepare for, respond to, and recover from emergency situations.” Emergency management https://niem.gov/communities/em/Pages/about-em.aspx

IETF	Standards institute	<p>“Internet Engineering Task Force (IETF) is the premier Internet standards body, developing open standards through open processes.</p> <p>The IETF is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. The technical work of the IETF is done in Working Groups, which are organized by topic into several Areas. Much of the work is handled via mailing lists. The IETF holds meetings three times per year.”</p>
DHS	National body	<p>Department of Homeland Security.</p> <p>“SAFECOM has created key documents such as the Interoperability Continuum, the Statement of Requirements (SoR) for baseline communications and interoperability standards, the Statewide Communication Interoperability Plan (SCIP) Methodology, and the National Emergency Communications Plan (NECP) to assist emergency responders nationwide with improving communications and interoperability.”</p> <p>https://www.dhs.gov/safecom</p>
EuroRec	Standards institute	<p>“The EUROREC Institute (EuroRec) is an independent not-for-profit organisation, promoting in Europe the use of high quality Electronic Health Record systems (EHRs). One of its main missions is to support, as the European authorised certification body, EHRs certification development, testing and assessment by defining functional and other criteria.”</p> <p>http://www.eurorec.org/index.cfm</p>
IHTSDO	Standards institute	<p>“The International Health Terminology Standards Development Organisation (IHTSDO) determines global standards for health terms, an essential part of improving the health of humankind.</p> <p>We are committed to maintaining and growing our leadership as the global experts in healthcare terminology, ensuring that SNOMED CT, our world-leading product, is accepted as the global common language for health terms.” SNOMED CT (1965) Systematized Nomenclature of Pathology (SNOP) to describe morphology and anatomy.</p> <p>(1975) expanded SNOP to create the Systematized Nomenclature of Medicine (SNOMED). The most widely adopted version of SNOMED was SNOMED II (1979).</p> <p>http://browser.ihtsdotools.org/</p>
IBM	IT solution provider	<p>IBM Emergency Management Center</p> <p>“Increase situational awareness: Managing an emergency situation involves many agencies, and it can be difficult for governmental systems and processes to transition from normal to extreme operations. For faster and more coordinated emergency responses, managers need information and insights to adapt to rapidly changing situations. The IBM Emergency Management Center solution offers near-real-time situational awareness to help improve decision-making speed, accuracy and effectiveness.”</p> <p>http://www.ibm.com/smarterplanet/us/en/smarter_cities/solutions/solution/planning_mgt_solutions/R321271K60502K40.html</p>
MIR3	IT solution provider	<p>Emergency Notification Solution For Crisis Management</p> <p>“Intelligent Notification can instantly launch an emergency notification via Web, email or phone to populations of any size across any geographical area. Reach thousands of people in just minutes by a variety of means, including email, mobile phone, SMS message, TTY, BlackBerry Messenger, pager, landline and more, in multiple languages or by a custom method. It reliably delivers an emergency notification to hundreds of thousands of people all over the world simultaneously.”</p> <p>“Intelligent Notification solutions for business continuity and disaster recovery, business operations, emergency notification, IT alerting, and CRM notification.”</p>
GE	IT solution provider	<p>GE Health Cloud. “The GE Health Cloud is designed to be a scalable, secure, connected and interoperable platform, delivering the largest application ecosystem for the healthcare industry.</p> <p>Designed to integrate into clinical workflows while managing the volume, velocity, and variety of healthcare data, the GE Health Cloud will be capable of connecting to more than 500,000 GE medical imaging machines and more than 1.5 million imaging machines worldwide. Additionally, the GE Health Cloud will be capable of linking to</p>

		millions of other healthcare devices including patient monitoring, diagnostics, anesthesia delivery, ultrasound, mammography and various data sources.” http://www3.gehealthcare.com/en/products/categories/ge_health_cloud
Intermedix	IT solution provider	Clinical Data Exchange (CDX) “(…) technology-enabled solutions for global health and emergency response. Our solutions support and connect health care providers and emergency preparedness & response personnel.” “The Clinical Data Exchange application is a state NEMSIS repository for collecting, analyzing and reporting pre-hospital patient care data submitted by EMS agencies. With this solution, agencies, regional EMS officials and state administrators can view and analyze a variety of reports within the Web interface for benchmarking and identifying quality improvement initiatives.” https://www.intermedix.com/solutions/cdx

Appendix C

Translations of SSN/SOSA and SAREF

This appendix provides the complete semantic translations of SAREF and SSN/SOSA from the mappings proposed in this thesis (Chapter 6), which were developed for INTER-IoT-EWS with the IPSM language (Chapter 7).

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE RDF
[
  <ENTITY sripas "http://www.inter-iot.eu/sripas#">
  <ENTITY var "http://www.inter-iot.eu/sripas#node_">
  <ENTITY sosa "http://www.w3.org/ns/sosa/">
  <ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
  <ENTITY iiot "http://inter-iot.eu/GOIoTP#">
  <ENTITY iiotex "http://inter-iot.eu/GOIoTPex#">
  <ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <ENTITY var "http://www.inter-iot.eu/sripas#node_">
  <ENTITY sarefInstances "https://w3id.org/saref/instances#">
  <ENTITY geo-sf "http://www.opengis.net/def/sf/">
  <ENTITY saref "https://w3id.org/saref#">
  <ENTITY time "http://www.w3.org/2006/time#">
  <ENTITY geo "http://www.w3.org/2003/01/geo/wgs84_pos#">
  <ENTITY geosparql "http://www.opengis.net/ont/geosparql#">
  <ENTITY sweet_units "http://sweet.jpl.nasa.gov/2.3/reprSciUnits.owl#">
  <ENTITY sweet_screla "http://sweet.jpl.nasa.gov/2.3/relaSci.owl#">
  <ENTITY sweet_mrela "http://sweet.jpl.nasa.gov/2.3/relaMath.owl#">
  <ENTITY sosaInst "http://www.w3.org/ns/sosa/instances/">
]
>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sripas="http://www.inter-iot.eu/sripas#"
  xmlns:var="http://www.inter-iot.eu/sripas#node_"
  xmlns:align="http://knowledgeweb.semanticweb.org/heterogeneity/alignment#"
  xmlns:exmo="http://exmo.inrialpes.fr/align/ext/1.0/"
  xmlns:sosa="http://www.w3.org/ns/sosa/"
  xmlns:iiot="http://inter-iot.eu/GOIoTP#"
  xmlns:time="http://www.w3.org/2006/time#"
  xmlns:sarefInstances="https://w3id.org/saref/instances#"
  xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
  xmlns:geosparql="http://www.opengis.net/ont/geosparql#"
  xmlns:sweet_units="http://sweet.jpl.nasa.gov/2.3/reprSciUnits.owl#"
  xmlns:sweet_screla="http://sweet.jpl.nasa.gov/2.3/relaSci.owl#"
  xmlns:sweet_mrela="http://sweet.jpl.nasa.gov/2.3/relaMath.owl#"
  xmlns:saref="https://w3id.org/saref#"
>
  <align:Alignment>

    <dcelem:title>SAREF_CO</dcelem:title>
    <exmo:version>0.59</exmo:version>
    <dcelem:creator>Joao Moreira (U.Twente)</dcelem:creator>
    <dcelem:description>ETSI SAREF to GOIoTP (W3C SSN/SOSA)</dcelem:description>

    <align:xml>yes</align:xml>
    <align:level>2IPSM</align:level>
    <align:type>***</align:type>
    <align:method>manual</align:method>
    <align:time>07-09-2018</align:time>
    <sripas:cellFormat>
      <iiot:DataFormat rdf:about="http://inter-iot.eu/sripas#turtle" />
    </sripas:cellFormat>

  </align:ontol>
```

```

    <align:Ontology rdf:about="https://w3id.org/saref#">
      <align:formalism>
        <align:Formalism align:name="OWL2.0" align:uri="http://www.w3.org/2002/07/owl#" />
      </align:formalism>
    </align:Ontology>
  </align:onto1>
  <align:onto2>
    <align:Ontology rdf:about="http://inter-iot.eu/goiotpex#">
      <align:formalism>
        <align:Formalism align:name="OWL2.0" align:uri="http://www.w3.org/2002/07/owl#" />
      </align:formalism>
    </align:Ontology>
  </align:onto2>

  <sripas:steps rdf:parseType="Collection">
    <sripas:step sripas:order="1" sripas:cell="http://www.inter-iot.eu/sripas#Cell1"/>
    <sripas:step sripas:order="2" sripas:cell="http://www.inter-iot.eu/sripas#Cell2"/>
    <sripas:step sripas:order="3" sripas:cell="http://www.inter-iot.eu/sripas#Cell3"/>
    <sripas:step sripas:order="4" sripas:cell="http://www.inter-iot.eu/sripas#Cell4"/>
    <sripas:step sripas:order="5" sripas:cell="http://www.inter-iot.eu/sripas#Cell5"/>
    <sripas:step sripas:order="6" sripas:cell="http://www.inter-iot.eu/sripas#Cell6"/>
  </sripas:steps>

  <align:map>
<!--
M04. Observations (Measurements) that a Sensor makes.
There is an ontological equivalence between these two patterns:
(1) sosa:Observation sosa:madeBySensor sosa:Sensor
    sosa:hasResult sosa:Result
        sosa:observedProperty ssn:Property
        sosa:phenomenonTime time:Instant
(2) saref:Sensor saref:makesMeasurement saref:Measurement
    saref:measuresProperty saref:Property
-->

    <align:Cell rdf:about="http://www.inter-iot.eu/sripas#Cell1">
      <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTA a saref:Sensor ;
          saref:measuresProperty var:CTD ;
          saref:makesMeasurement var:CTB .
        var:CTB saref:hasValue var:CTV ;
          saref:hasTimestamp var:CTT ;
          saref:relatesToProperty var:CTC ;
          saref:isMeasuredIn var:CTU .
      </align:entity1>
      <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTB a sosa:Observation ;
          sosa:madeBySensor var:CTA ;
          sosa:hasResult [
            a sosa:Result ;
            iiot:hasValue var:CTV ;
            iiot:hasUnit var:CTU
          ] ;
          sosa:observedProperty var:CTC ;
          sosa:phenomenonTime [
            a time:Instant ;
            time:inTimePosition [
              a time:TimePosition ;
              time:numericPosition var:CTT
            ]
          ] .
        var:CTA a sosa:Sensor ;
          saref:measuresProperty var:CTD .
      </align:entity2>
    </align:relation>=</align:relation>
  </align:Cell>
</align:map>

  <align:map>
<!--
M05. Device Geo-Location

```

While in SSN/SOSA the GeoSPARQL approach is recommended to represent geospatial objects, in SAREF the conventional (and simple) approach based on the WGS84 ontology is suggested. In particular, GeoSPARQL provides a sophisticated way to represent latitude and longitude through the `geosparql:asWKT` property, which links to a `geosparql:wktLiteral` with value as the string "Point(#latitude, #longitude)", where #latitude and #longitude are float values. The WGS84 approach simply introduces a class `geo:Point` which represents latitude and longitude through the `geo:lat` and `geo:lon` properties, respectively.

```

-->
    <align:Cell rdf:about="http://www.inter-iot.eu/sripas#Cell2">
      <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTX geo:lat var:x ;
          geo:long var:y .
      </align:entity1>
      <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTX geosparql:asWKT var:z .
      </align:entity2>
    </align:relation>=</align:relation>
    <sripas:transformation rdf:parseType="Literal">
      <sripas:function sripas:about="str">
        <sripas:param sripas:order="1" sripas:about="&var;x"/>
        <sripas:return sripas:about="&var;sx"/>
      </sripas:function>
      <sripas:function sripas:about="str">
        <sripas:param sripas:order="1" sripas:about="&var;y"/>

```

```

        <srripas:return srripas:about="&var;sy"/>
    </srripas:function>
    <srripas:function srripas:about="concat">
        <srripas:param srripas:order="1" srripas:val="Point("/>
        <srripas:param srripas:order="2" srripas:about="&var;sx"/>
        <srripas:param srripas:order="3" srripas:val=" "/>
        <srripas:param srripas:order="4" srripas:about="&var;sy"/>
        <srripas:param srripas:order="5" srripas:val="")"/>
        <srripas:return about="&srripas;node_z"/>
    </srripas:function>
    <srripas:transformation>
    <srripas:filters rdf:parseType="Literal">
        <srripas:filter srripas:about="&var;sx" srripas:datatype="xsd:string"/>
        <srripas:filter srripas:about="&var;sy" srripas:datatype="xsd:string"/>
    </srripas:filters>
    <srripas:typings rdf:parseType="Literal">
        <srripas:typing srripas:about="&var;z" srripas:datatype="&geo-sf;wktLiteral"/>
    </srripas:typings>
    </align:Cell>
</align:map>

<align:map>
<!--
M02. Device Composition

ssn:hasSubSystem and saref:consistsOf are ontologically equivalent.

-->
<align:Cell rdf:about="http://www.inter-iot.eu/sripas#Cell3">
    <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTA a saref:Device ;
        saref:consistsOf var:CTB .
        var:CTB saref:measuresProperty var:CTC .
    </align:entity1>
    <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTA a sosa:Platform ;
        sosa:hosts var:CTB .
        var:CTB sosa:observes var:CTC .
    </align:entity2>
    <align:relation>=</align:relation>
    </align:Cell>
</align:map>

<align:map>
<!--
M03. Property Measured by a Sensor

sosa:observes and saref:measuresProperty are ontologically equivalent.
>> sosa:observedProperty

-->
<align:Cell rdf:about="http://www.inter-iot.eu/sripas#Cell4">
    <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTA sosa:observedProperty saref:instances:BatteryLevel .
    </align:entity1>
    <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTS sosa:observedProperty saref:instances:BatteryLevel .
        saref:instances:BatteryLevel a sosa:ObservableProperty .
    </align:entity2>
    <align:relation>=</align:relation>
    </align:Cell>
</align:map>

<align:map>
<!--
INTER-IoT specific mapping for units of measurement

-->
<align:Cell rdf:about="http://www.inter-iot.eu/sripas#Cell5">
    <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTA iiot:hasUnit saref:instances:ElectricPotential_MilliVolts .
    </align:entity1>
    <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTA iiot:hasUnit saref:instances:ElectricPotential_MilliVolts .
        saref:instances:ElectricPotential_MilliVolts a
        sweet_units:Unit,
        sweet_units:UnitDerivedByScaling ;
        sweet_screla:hasBaseUnit sweet_units:volt ;
        sweet_screla:hasPrefix sweet_units:milli ;
        sweet_mrela:hasScalingNumber 0.001 .
    </align:entity2>
    <align:relation>=</align:relation>
    </align:Cell>
</align:map>

<align:map>
<!--
M05. Device Geo-Location

-->
<align:Cell rdf:about="http://www.inter-iot.eu/sripas#Cell6">
    <align:entity1 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTA geo:location var:CTB .
        var:CTB a geo:Point .
    </align:entity1>
    <align:entity2 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
        var:CTA iiot:hasLocation var:CTB .
    </align:entity2>
    <align:relation>=</align:relation>
    </align:Cell>
</align:map>

```

```
var:CTB a iiot:Location .  
  </align:entity2>  
  <align:relation>=</align:relation>  
  </align:Cell>  
  </align:map>  
  </align:Alignment>  
</rdf:RDF>
```

Appendix D

Meta-model of SML 2.0

This appendix provides the meta-model of the new version of SML (2.0) produced in this thesis. SML 2.0 was developed with the Sparx Enterprise Architect 12.1 following the approach for language definition through profiles. The entire code is available here¹⁶⁷. The code below refers to the definition of the main element, profile.xml:

```
<?xml version="1.0" encoding="windows-1252"?>
<UMLProfile profiletype="uml2">
  <Documentation id="15E88BE3-B" name="SML" version="2.0" alias="SML 2.0" notes="Situation Modeling Language"/>
  <Content>
    <Stereotypes>
      <Stereotype name="AllenLink" metatype="AllenLink" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0">
        <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes" dt:dt="bin.base64">UESDBBQAAAAIABKImEZ0UvU0hAAAAOoAAAAHABEAc3RyLmRhZGVudQAHMXY6VzI2OlWZdjpVjY7BCcJAEEEXfNYI9hOSiLeJkxZgAwajBFYTkVUUUpEl2JzPRRRv8mF2/8ybvxs4UFETyLnQcFRBX9ESVWs3fJg5M0aydGbsJxbsnA7srWddwdVb5EbvRsFS5XS6xklMxF26NrczvZcOPGRG5vsbtqx+MkqmlLN+v/r6x2D9J/W7N/EEUESBAhcLFAAAAgAEoiYRnRS9TSEAAAA6gAAAAACQAAAAAAAAAAAAAHN0ci5kYXRvVAUAB5l2OlVQSwUGAAAAAAEAAQA+AAAAugAAAAAA
        </Image>
        <AppliesTo>
          <Apply type="Association">
            <Property name="direction" value="Unspecified"/>
            <Property name="_lineStyle" value="orthogonalS"/>
            <Property name="_SourceMultiplicity" value="0..*"/>
            <Property name="_TargetMultiplicity" value="0..*"/>
          </Apply>
        </AppliesTo>
        <TaggedValues>
          <Tag name="nature" type="AllenRelationType" description="" unit="" values="" default=""/>
          <Tag name="reflexivity" type="Reflexivity" description="" unit="" values="" default=""/>
          <Tag name="symmetry" type="Symmetry" description="" unit="" values="" default=""/>
          <Tag name="transitivity" type="Transitivity" description="" unit="" values="" default=""/>
          <Tag name="cyclicity" type="Ciclicity" description="" unit="" values="" default=""/>
        </TaggedValues>
      </Stereotype>
      <Stereotype name="AllenLinkAfter" alias="AllenLinkBefore Alias" metatype="AllenLinkAfter" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="AllenLink" baseStereotypes="AllenLink">
        <TaggedValues>
          <Tag name="nature" type="AllenRelationType" description="" unit="" values="" default="after"/>
        </TaggedValues>
      </Stereotype>
    </Stereotypes>
  </Content>
</UMLProfile>
```

¹⁶⁷ <https://github.com/jonimoreira/ea-sml-editor>

```

        <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
        <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymetric,antisymmetric,nonsymmetric" default="assymmetric"/>
        <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
</Stereotype>
<Stereotype name="AllenLinkBefore" alias="AllenLinkBefore Alias"
metatype="AllenLinkBefore" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1"
borderwidth="-1" hideicon="0" generalizes="AllenLink" baseStereotypes="AllenLink">
    <TaggedValues>
        <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
default="before"/>
        <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
        <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymetric,antisymmetric,nonsymmetric" default="assymmetric"/>
        <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
</Stereotype>
<Stereotype name="AllenLinkCoincides" alias="AllenLinkBefore Alias"
metatype="AllenLinkCoincids" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="AllenLink"
baseStereotypes="AllenLink">
    <TaggedValues>
        <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
default="coincids"/>
        <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="reflexive"/>
        <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymetric,antisymmetric,nonsymmetric" default="symmetric"/>
        <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
</Stereotype>
<Stereotype name="AllenLinkDuring" alias="AllenLinkBefore Alias"
metatype="AllenLinkDuring" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1"
borderwidth="-1" hideicon="0" generalizes="AllenLink" baseStereotypes="AllenLink">
    <TaggedValues>
        <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
default="during"/>
        <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
        <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymetric,antisymmetric,nonsymmetric" default="assymmetric"/>
        <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
</Stereotype>
<Stereotype name="AllenLinkFinishedBy" alias="AllenLinkBefore Alias"
metatype="AllenLinkFinishedBy" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="AllenLink"
baseStereotypes="AllenLink">
    <TaggedValues>
        <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
default="finishedBy"/>
        <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
        <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymetric,antisymmetric,nonsymmetric" default="assymmetric"/>
        <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
</Stereotype>
<Stereotype name="AllenLinkFinishes" alias="AllenLinkBefore Alias"
metatype="AllenLinkFinishes" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="AllenLink"
baseStereotypes="AllenLink">
    <TaggedValues>
        <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
default="finishes"/>
        <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
        <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymetric,antisymmetric,nonsymmetric" default="assymmetric"/>
        <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>

```



```

    </TaggedValues>
  </Stereotype>
  <Stereotype name="AllenLinkIncludes" alias="AllenLinkBefore Alias"
    metatype="AllenLinkIncludes" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
    bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="AllenLink"
    baseStereotypes="AllenLink">
    <TaggedValues>
      <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
        default="includes"/>
      <Tag name="reflexivity" type="enumeration" description="" unit=""
        values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
      <Tag name="symmetry" type="enumeration" description="" unit=""
        values="symmetric,assymmetric,antisymmetric,nonsymmetric" default="assymmetric"/>
      <Tag name="transitivity" type="enumeration" description="" unit=""
        values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="AllenLinkMeets" alias="AllenLinkBefore Alias"
    metatype="AllenLinkMeets" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1"
    borderwidth="-1" hideicon="0" generalizes="AllenLink" baseStereotypes="AllenLink">
    <TaggedValues>
      <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
        default="meets"/>
      <Tag name="reflexivity" type="enumeration" description="" unit=""
        values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
      <Tag name="symmetry" type="enumeration" description="" unit=""
        values="symmetric,assymmetric,antisymmetric,nonsymmetric" default="assymmetric"/>
      <Tag name="transitivity" type="enumeration" description="" unit=""
        values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="AllenLinkMetBy" alias="AllenLinkBefore Alias"
    metatype="AllenLinkMetBy" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1"
    borderwidth="-1" hideicon="0" generalizes="AllenLink" baseStereotypes="AllenLink">
    <TaggedValues>
      <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
        default="metBy"/>
      <Tag name="reflexivity" type="enumeration" description="" unit=""
        values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
      <Tag name="symmetry" type="enumeration" description="" unit=""
        values="symmetric,assymmetric,antisymmetric,nonsymmetric" default="assymmetric"/>
      <Tag name="transitivity" type="enumeration" description="" unit=""
        values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="AllenLinkOverlappedBy" alias="AllenLinkBefore Alias"
    metatype="AllenLinkOverlappedBy" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
    bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="AllenLink"
    baseStereotypes="AllenLink">
    <TaggedValues>
      <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
        default="overlappedBy"/>
      <Tag name="reflexivity" type="enumeration" description="" unit=""
        values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
      <Tag name="symmetry" type="enumeration" description="" unit=""
        values="symmetric,assymmetric,antisymmetric,nonsymmetric" default="assymmetric"/>
      <Tag name="transitivity" type="enumeration" description="" unit=""
        values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="AllenLinkOverlaps" alias="AllenLinkBefore Alias"
    metatype="AllenLinkOverlaps" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
    bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="AllenLink"
    baseStereotypes="AllenLink">
    <TaggedValues>
      <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
        default="overlaps"/>
      <Tag name="reflexivity" type="enumeration" description="" unit=""
        values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
      <Tag name="symmetry" type="enumeration" description="" unit=""
        values="symmetric,assymmetric,antisymmetric,nonsymmetric" default="assymmetric"/>
      <Tag name="transitivity" type="enumeration" description="" unit=""
        values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="AllenLinkStartedBy" alias="AllenLinkBefore Alias"
    metatype="AllenLinkStartedBy" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
    bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="AllenLink"
    baseStereotypes="AllenLink">

```

```

    <TaggedValues>
      <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
default="startedBy"/>
      <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
      <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymetric,antisymmetric,nonsymmetric" default="assymetric"/>
      <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="AllenLinkStarts" alias="AllenLinkBefore Alias"
metatype="AllenLinkStarts" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1"
borderwidth="-1" hideicon="0" generalizes="AllenLink" baseStereotypes="AllenLink">
    <TaggedValues>
      <Tag name="nature" type="AllenRelationType" description="" unit="" values=""
default="starts"/>
      <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
      <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymetric,antisymmetric,nonsymmetric" default="assymetric"/>
      <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="AttributeLink" alias="testando alias" metatype="AttributeLink"
notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1"
hideicon="0">
    <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIAHK+VUkSS1gPnAAAAACABAAAHABEAc3RyLmRhdFVUDQAHDacKWA2qClgNqgpY
XY7BCsIwEETffVcF/KD0peGg9iCA9CB68+wVioWZaW9KoiPjvJmLVGE3s8ybtRrP07KgRJRJS
1jMPRGwZSFU0wc+1bmR4HBexy46cNRsOKhdUzkbdc1TKi0+Zf91GjhdrKXT22licHGNRMxZM
e/eklyZC34wmzl2TUTmzYqtXo0DH3WJUpuvIVWJSrQz+EM/u1o/1ftjWuLdn6oXUESBAhCL
FAAAAagAcr5VSRJLWA+cAAAAIAEAAAACACQAAAAAIAAAAAAHN0ci5kYXRvVVAUABw2q
ClhQSwUGAAAAAAEAQA+AAAA0gAAAAAA</Image>
    <AppliesTo>
      <Apply type="Association">
        <Property name="_lineStyle" value="orthogonals"/>
        <Property name="_TargetMultiplicity" value="1"/>
        <Property name="_SourceMultiplicity" value="1"/>
        <Property name="direction" value="Unspecified"/>
      </Apply>
    </AppliesTo>
    <TaggedValues>
      <Tag name="reflexivity" type="Reflexivity" description="" unit="" values=""
default=""/>
      <Tag name="symmetry" type="Symmetry" description="" unit="" values="" default=""/>
      <Tag name="transitivity" type="Transitivity" description="" unit="" values=""
default=""/>
      <Tag name="cyclicity" type="Ciclicity" description="" unit="" values="" default=""/>
      <Tag name="negated" type="boolean" description="" unit="" values="true,false"
default="false"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="AttributeReference" metatype="AttributeReference" notes="An
AttributeReference is a reference to a quality (also can be a reference to an attribute)
defined in the context model.&#xA;" cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1"
borderwidth="-1" hideicon="0" generalizes="ReferableElement"
baseStereotypes="ReferableElement">
    <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIAADgFnEgPBamJ/wAAAC4CAAAHABEAc3RyLmRhdFVUDQAHKFWvhIVcoXCFX
1ZFFS8NAEMkftUK/w5JcWqhoXyNg9aTgzYPeRe2aBtoYmihIyXf3N9OQLHiSYWdnu/Nn5dG
G72qVlTJqghUpalOdNDE781wb/TC+xZEASbohPpPpHwalfFr8nu+6x3//Arq3BteDfYER9kEwi
9cCreNvxbeScYTVVbNoWBKUZrYssV9N3iPpVfWvB9qj5EW8dwj36hIVPafM9cRbeb0mXsUeJYjM9
UKnRM75wdE125zqt/A3T2NgZdf6ci3nWeHOMNYBXMrpEiU6evOCUEsBAhCLFAAAAAGAOAwC
SA8FqYn/AAAAIgIAAAACAAQAAAAAIAAAAAAIAAAAAAHN0ci5kYXRvVVAUABYhclVQdQSwUGAAAA
AAEAQA+AAAAQEAQAAAA</Image>
    </Stereotype>
  <Stereotype name="CharacterizationLink" metatype="CharacterizationLink" notes="" cx="90"
cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0">
    <AppliesTo>
      <Apply type="Association">
        <Property name="_lineStyle" value="orthogonals"/>
        <Property name="_TargetMultiplicity" value="1..*"/>
        <Property name="_SourceMultiplicity" value="1"/>
        <Property name="direction" value="Unspecified"/>
      </Apply>
    </AppliesTo>

```

```

    <TaggedValues>
      <Tag name="reflexivity" type="Reflexivity" description="" unit="" values=""
default="irreflexive"/>
      <Tag name="symmetry" type="Symmetry" description="" unit="" values=""
default="assymmetric"/>
      <Tag name="cyclicity" type="Ciclicity" description="" unit="" values=""
default="acyclic"/>
      <Tag name="transitivity" type="Transitivity" description="" unit="" values=""
default="transitive"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="Class1" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0"/>
  <Stereotype name="ContextFormalLink" metatype="ContextFormalLink" notes="" cx="90"
cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="FormalRelation" baseStereotypes="FormalRelation">
    <TaggedValues>
      <Tag name="parameter" type="string" description="" unit="" values="" default=""/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="EntityParticipant" metatype="EntityParticipant"
notes="EntityParticipant, which are the object classes from the context model and represents
the entities such as a person, a car, a client or a building." cx="90" cy="70" bgcolor="-1"
fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="Participant"
baseStereotypes="Participant">
    <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIAIoFnEgC7Aw4EQEAAGoCAAAHABEAc3RyLmRhdFVUDQAH9VwhV/VcIVf1XCFX
1ZFLS8NQEIxPvsh/EUJNCgWNf1R8rFy4c6F70TSNgTSWJApS+t/9ZhKSC65kuHPnzj1nnp0+
9KadCkXaYlVqdkJ7V0f8FuKt8Zf8hPpjhMrB9+oR7e8bkbs9z+whdbgevDvMAppteBVV64nX
8LfFN3NokR1RrMyebKOU3lskUtdmKPSJ9eMdJbPHYi2mCH/ZyZqCnZDRiO/w915bJtErjkt
zEwrXwsJ8lWxuhytFTJns2py77FhItZlqjNHDjrjDq2Zafrg2k4X7Kj2idmki3ZZIZan7zD
0ru0/p45S8+S6SrIYIyKz1I9eHTC7p0RsXrwWdZsVHbkOWIid3qy/exQKJglsbagws5h2Ca
Qye/UESBAhLFAAAAAgA1gWcSALSDdGRAQAaAgIAAAACACQAAAAAaaaaaCAAAAAAHN0ci5k
YXRvVAUAB/VcIVdGSwUGAAAAAEEAAQA+AAAAARwEAAAAA</Image>
    </Stereotype>
  <Stereotype name="EqualsLink" metatype="EqualsLink" notes="" cx="90" cy="70" bgcolor="-
1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="FormalRelation"
baseStereotypes="FormalRelation">
    <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIAHygkfhbivzTwAAAGQAAAAHABEAc3RyLmRhdFVUDQAH9VwhV/VcIVf1XCFX
y2FIZEHISGIXIVVBygyXIZEGbWhwgP4khn6EECPOBoj1WbWbMXAzVDJxgmpOhgKEIqCMPqEqD
QYlhnNYM:QyFDKVBtDsNuIF+TwRqsrpYBAFBLAQIXCQAAAAIAHygkfhbivzTwAAAGQAAAAH
AAkAAAAAaaaaaAaaaaABZdHiuZGF0VGVQFAACCHVtXUESFBGAAAAAABAEAFGAAAIUAAAA
AA==</Image>
    </Stereotype>
  <Stereotype name="FormalRelation" metatype="FormalRelation" notes="" cx="90" cy="70"
bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0">
    <AppliesTo>
      <Apply type="Association">
        <Property name="direction" value="Source -> Destination"/>
        <Property name="_lineStyle" value="orthogonals"/>
        <Property name="_SourceMultiplicity" value="0..*"/>
        <Property name="_TargetMultiplicity" value="0..*"/>
      </Apply>
    </AppliesTo>
  </TaggedValues>
  <Tag name="reflexivity" type="Reflexivity" description="" unit="" values=""
default=""/>
  <Tag name="symmetry" type="Symmetry" description="" unit="" values="" default=""/>
  <Tag name="transitivity" type="Transitivity" description="" unit="" values=""
default=""/>
  <Tag name="cyclicity" type="Ciclicity" description="" unit="" values="" default=""/>
  <Tag name="negated" type="boolean" description="" unit="" values="true,false"
default="false"/>
</TaggedValues>
</Stereotype>
<Stereotype name="Function" metatype="Function" notes="Formerly references to the so-
called Qualitative Formal Relations, Functions are re- designed as user-defined operations on
the situation type elements. Not every relation between elements in the SML model is
necessarily established in the context model. Take for example the sum of two numbers.
Although in the real world, given two numbers there exists a formal sum relation between them,
usually the ontology designer (in our case the context-model designer) leaves out those
subtleties that may not aggregate much in the description of a domain. This means establishing
a modeling scope so that the model does not grow indefinitely in complexity, as technically
the possibilities of talking about things are infinite. Thus, relations like sum may always be
defined in the context model if wanted, but we also leave open the possibility of defining
them as functions (without linking to any context model elements) in the SML model itself. It
is important to mention, however, that functions must be manually implemented in some
automatic process is to be used (such as simulation in Alloy). Figure 29 depicts an example of

```

```

a High Body Mass Index (BMI) situation using functions.&#xA;" cx="90" cy="70" bgcolor="-1"
fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="ReferableElement"
baseStereotypes="ReferableElement">
  <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIAFQFnEhh7pJvEgEAAFWCAAAHABEAc3RyLmRhdhFVUDQAHYlwhV2JcIVdiXCFX
1ZFLS8RQDIXPVsH/UNPNB0ac8S0+Nj5wS2LciZrXTqTh7YKMvS/+9lwas+4kpCQNOecNLmt
VnrVRk6JlmslaulpVluLoYGu9EKvolvQT3SNp3qHvasjNlSXafv9D6zTElwh/g2G0weV1lsO
vJremm8j5wDboOL/tANbK6d7hWVqwsnT7If2yvTDZzqMi j8ZWfDnCxJgG/4HunBzgVdot2
hTcwj3SmKbi5LkJ2qpNo0r0xSia2a0TalZGoQ80Ca2ZlonOykeVjb9F7G71QZZfyF4rfyGt3
erKtCtvM77TapzZpJv6oXnLlXI8otXomFoYuqe7sdiUv6N/Fs1N0G33Z/sdYEt30s7bgYk4f
Xa/XL1BLAQIXCxAQAAAAIAFQFnEhh7pJvEgEAAFWCAAAHAAKAAAAAAAAAAAAAABzdHIU
ZGF0VQVFAADIXCFXUESFBgAAAAABAAEAPgAAAEgBAAAAAA=</Image>
</Stereotype>
<Stereotype name="FunctionParameterLink" notes="" cx="90" cy="70" bgcolor="-1"
fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0">
  <AppliesTo>
    <Apply type="Association">
      <Property name="direction" value="Source -&gt; Destination"/>
      <Property name="_lineStyle" value="orthogonals"/>
      <Property name="_SourceMultiplicity" value="0..*"/>
      <Property name="_TargetMultiplicity" value="0..*"/>
    </Apply>
  </AppliesTo>
</Stereotype>
<Stereotype name="FunctionResultLink" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-
1" bordercolor="-1" borderwidth="-1" hideicon="0">
  <AppliesTo>
    <Apply type="Association">
      <Property name="direction" value="Source -&gt; Destination"/>
      <Property name="_lineStyle" value="orthogonals"/>
      <Property name="_SourceMultiplicity" value="0..*"/>
      <Property name="_TargetMultiplicity" value="0..*"/>
    </Apply>
  </AppliesTo>
</Stereotype>
<Stereotype name="InstanceOf" alias="testando alias" metatype="InstanceOf" notes=""
cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0">
  <AppliesTo>
    <Apply type="Association">
      <Property name="_lineStyle" value="direct"/>
      <Property name="_TargetMultiplicity" value="1"/>
      <Property name="_SourceMultiplicity" value="1"/>
      <Property name="direction" value="Unspecified"/>
    </Apply>
  </AppliesTo>
  <TaggedValues>
    <Tag name="reflexivity" type="Reflexivity" description="" unit="" values=""
default=""/>
    <Tag name="symmetry" type="Symmetry" description="" unit="" values="" default=""/>
    <Tag name="transitivity" type="Transitivity" description="" unit="" values=""
default=""/>
    <Tag name="cyclicity" type="Ciclicity" description="" unit="" values="" default=""/>
    <Tag name="negated" type="boolean" description="" unit="" values="true,false"
default="false"/>
  </TaggedValues>
</Stereotype>
<Stereotype name="InstantiationLink" metatype="InstantiationLink" notes="" cx="90"
cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="FormalRelation" baseStereotypes="FormalRelation">
  <Stereotype name="Literal" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0" isAbstract="true"
generalizes="SituationTypeElement" baseStereotypes="SituationTypeElement"
strictness="profile">
    <TaggedValues>
      <Tag name="isDerived" type="boolean" description="" unit="" values="true,false"
default=""/>
    </TaggedValues>
  </Stereotype>
<Stereotype name="MediationLink" metatype="MediationLink" notes="" cx="90" cy="70"
bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0">
  <AppliesTo>
    <Apply type="Association">
      <Property name="direction" value="Unspecified"/>
      <Property name="_lineStyle" value="orthogonals"/>
      <Property name="_SourceMultiplicity" value="0..*"/>
      <Property name="_TargetMultiplicity" value="1..*"/>
    </Apply>
  </AppliesTo>
  <TaggedValues>

```

```

    <Tag name="reflexivity" type="Reflexivity" description="" unit="" values=""
default="irreflexive"/>
    <Tag name="symmetry" type="Symmetry" description="" unit="" values="" default=""/>
    <Tag name="transitivity" type="Transitivity" description="" unit="" values=""
default=""/>
    <Tag name="cyclicity" type="Ciclicity" description="" unit="" values="" default=""/>
  </TaggedValues>
  </Stereotype>
  <Stereotype name="ModeReference" metatype="ModeReference" notes="a ModeReference is a
reference to a mode defined in the context model.&#xA;" cx="90" cy="70" bgcolor="-1"
fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="ReferableElement"
baseStereotypes="ReferableElement">
    <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIABQFnEhk0IBysQEAAHIDAAAHABEAc3RyLRhdfVUDQAHglwhVxpcIVcaXCFX
1VLLSsNAFD1bBf8HJJsUBK3YovjY+MCdhXYvasY0kKQ1GGu15N899+Y1Nd2Uy9ybyZxzH2em
xBKEFWMPAQ8avBDnOcIqd+hNc0OawPCnoZ4yWjBAj3DmIDCtmMMyhhjjknbezHlkqhX+IycD
nNT7b1cC6PMNo7BrxDDPfr3jujqEfcEuOqjDEh/8K51iYjw8cPmcwXBn6QvuWuzmCKXhuvRv
4IsMgx/uDL+ijpfzLOO+niOdrplFprLE5uzUxz0tQN1UMJzVYqt3H+CR5jvTDN1BvYdwkDV6
rtg9kpH5npg55S8rTu81lbnqnlDab0U8a+skxxW2qW0hr1PU5o4tt7HTucqgtZ0u82VW1EEZlp
X26pPb3rHLH011PMudrXSPaSUncQb8xUYkEfKzrh7lnVsvR9R8r2mbfAryo+onmOWsLaEedy
KkevCn9QSWECFwsUAAAACAAUBZxIZNCackkBAABYzAwAABwAJAAAAAIAAAAAAIAAAAAA3Ry
LmRhdfVUBQAHglwhV1BLBQYAAAAQAABD4AAAB/AQAAAAA=</Image>
  </Stereotype>
  <Stereotype name="OrderedComparativeLink" metatype="OrderedComparativeLink" notes=""
cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="FormalRelation" baseStereotypes="FormalRelation">
    <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIACydykhtyRfYhwAAAPYAAAAHABEAc3RyLRhdfVUDQAH0hdv9IXW1fSF1tX
jY5BD0IwEEXfPhPuQGCDV8CVcaEH8AIQ0TQpQqC4IZzII3g5vxOiYwd+Mp3fefNbT01fJseh
wXGRvHxF55Ba3f0vE7NhrIzEn0156TpwFnlJpdyVxcY6bWRspUSOjmnSTDiiCyO5G90Y6H
6JfiiUxZe44Uq6SM2dJy29uf3wyq/2f/tmfUEsBAhcLFAAAAAgALJ3KSG3JF9iHAAAA9gAA
AAcACQAAAAAIAAAAAAHN0ci5kYXRvVAUAB9IXWldQSwUGAAAAAEEAAQA+AAAAvQAA
AAAA</Image>
    <TaggedValues>
      <Tag name="type" type="enumeration" description="" unit=""
values="lessThan,greaterThan,lessThanOrEqualTo,greaterThanOrEqualTo" default="lessThan"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="OrderedComparativeLinkGreaterThan"
metatype="OrderedComparativeLinkGreaterThan" notes="" cx="90" cy="70" bgcolor="-1"
fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="OrderedComparativeLink" baseStereotypes="OrderedComparativeLink">
    <TaggedValues>
      <Tag name="nature" type="enumeration" description="" unit=""
values="lessThan,greaterThan,lessThanOrEqualTo,greaterThanOrEqualTo" default="greaterThan"/>
      <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
      <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
      <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymetric,antisymmetric,nonsymmetric" default="assymetric"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="OrderedComparativeLinkGreaterThanOrEqualTo"
metatype="OrderedComparativeLinkGreaterThanOrEqualTo" notes="" cx="90" cy="70" bgcolor="-1"
fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="OrderedComparativeLink" baseStereotypes="OrderedComparativeLink">
    <TaggedValues>
      <Tag name="nature" type="enumeration" description="" unit=""
values="lessThan,greaterThan,lessThanOrEqualTo,greaterThanOrEqualTo"
default="greaterThanOrEqualTo"/>
      <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
      <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="reflexive"/>
      <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymetric,antisymmetric,nonsymmetric" default="antisymmetric"/>
    </TaggedValues>
  </Stereotype>
  <Stereotype name="OrderedComparativeLinkLessThan"
metatype="OrderedComparativeLinkLessThan" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="OrderedComparativeLink"
baseStereotypes="OrderedComparativeLink">
    <TaggedValues>
      <Tag name="nature" type="enumeration" description="" unit=""
values="lessThan,greaterThan,lessThanOrEqualTo,greaterThanOrEqualTo" default="lessThan"/>
      <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>

```

```

    <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="irreflexive"/>
    <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymmetric,antisymmetric,nonsymmetric" default="assymmetric"/>
  </TaggedValues>
</Stereotype>
<Stereotype name="OrderedComparativeLinkLessThanOrEqualTo"
metatype="OrderedComparativeLinkLessThanOrEqualTo" notes="" cx="90" cy="70" bgcolor="-1"
fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="OrderedComparativeLink" baseStereotypes="OrderedComparativeLink">
  <TaggedValues>
    <Tag name="nature" type="enumeration" description="" unit=""
values="lessThan,greaterThan,lessThanOrEqualTo,greaterThanOrEqualTo"
default="lessThanOrEqualTo"/>
    <Tag name="transitivity" type="enumeration" description="" unit=""
values="transitive,intransitive,nontransitive" default="transitive"/>
    <Tag name="reflexivity" type="enumeration" description="" unit=""
values="reflexive,irreflexive,nonreflexive" default="reflexive"/>
    <Tag name="symmetry" type="enumeration" description="" unit=""
values="symmetric,assymmetric,antisymmetric,nonsymmetric" default="antisymmetric"/>
  </TaggedValues>
</Stereotype>
<Stereotype name="Participant" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0" isAbstract="true"
generalizes="ReferableElement" baseStereotypes="ReferableElement" strictness="profile">
  <TaggedValues>
    <Tag name="isDerived" type="boolean" description="" unit="" values="true,false"
default=""/>
    <Tag name="max" type="int" description="" unit="" values="" default="1"/>
    <Tag name="min" type="int" description="" unit="" values="" default="1"/>
    <Tag name="immutable" type="boolean" description="" unit="" values="true,false"
default="true"/>
  </TaggedValues>
</Stereotype>
<Stereotype name="QualityLiteral" metatype="QualityLiteral" notes="A Literal either
represents a Quality's value (QualityLiteral, but can also represent a Datatype's value) such
as $1000.00 or 37%, or some Type (TypeLiteral) used in instantiation relations to indicate
whether an entity is or is not an instance of that class.&#xA;" cx="90" cy="70" bgcolor="-1"
fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="Literal"
baseStereotypes="Literal">
  <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIAHGFNefjvrVEDwEAAFACAAAHABEAc3RyLmRhdFVUDQAHOtFwHv6BcIVegXCFX
1ZFNS8NAElbfq4L/ISSXFCq2RaHgX8UPvHmod6k2jYEOluwqSM1/99kxJAueZJhhZvZ9Z3Zm
nN61113jFEu3wKjU60bEOzo94fsfBky95SXSNpnoD38hjW6LLHvv1D2yhdTgP/hVGos1rgbcZ
eAlvO3Ij5wZyUx80YNT1PN6hWRYfYdCH3jfn1GmGyTVZKjw150NfbIImfT4FXmvBgglckvt
GmhlLnSBTEHO8ZfmzaaJ098ao6OioiketU52ANYH45M4sTmCOrMxvURVepbUdhN/pdQ1WwJ5un
tJnCNCt0aJ3mVB5/U7hfXI9UcnrGloauiO5saxW3CxcJ7JS6rT5t8xMkibYWWAAdMaeI9tbp
B1BLAQIXCQAAAAIAHGFNefjvrVEDwEAAFACAAAHAAKAAAAAAAAAAAAAAGAAAAABzdHIuZGF0
VVFQFAegXCFXUESFbgAAAAABAAEAPgAAAEUBAAAAA==</Image>
  </Image>
  </Stereotype>
<Stereotype name="ReferableElement" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0" isAbstract="true"
generalizes="SituationTypeElement" baseStereotypes="SituationTypeElement"
strictness="profile">
  <TaggedValues>
    <Tag name="isDerived" type="boolean" description="" unit="" values="true,false"
default=""/>
  </TaggedValues>
</Stereotype>
<Stereotype name="RelatorParticipant" metatype="RelatorParticipant"
notes="RelatorParticipant, which represents the external contexts of&#xA;the object classes,
i.e. the relators. Are called participants since they provide identity to the situation type,
but they are not connected to them by participation relations, in the ontological sense."
cx="90" cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="Participant" baseStereotypes="Participant">
  <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIAKAFnEhsWEYwEgEAAFACAAAHABEAc3RyLmRhdFVUDQAHPF0hVzxdIVc8XSFX
1ZFPS8NAEMKfVcHvEJJLCwXbYrHgn4si3hTqXdsSBNsYKpeS7+9sxpueZJZ2Z3Znff
eK30oq2cEq3JC1U60bH2Fo+PnNacpSTXSFP30uDX6khl1uwuOuznP7BOS3AN+FYtU/sHNfY
51XU1pwdOKfYli7hj3YsgOq1lgm393gtChb2Y8yXWopn2Hv+ysvyf7hVxw0ugOdInd0LXE
azhTjTjTIDdOsGmXnWMH9qPhd+iwsXtmcALuZ3r4nefccc9sLQb30VxKUymoE08mqOhtlQ/2
q9x+Fv60wEd200Tz6FWBUa0Qpd09Hoi5sYo2N2afgVTDLMJHVJ61/qwGQyxJNivsPbgYk4b
KdjqC1BLAQIXCQAAAAIAKAFnEhsWEYwEgEAAFACAAAHAAKAAAAAAAAAAAAAAGAAAAABzdHIuZGF0
VVFQFAAc8XSFXUESFbgAAAAABAAEAPgAAAEgBAAAAA==</Image>
  </Image>
  </Stereotype>
<Stereotype name="SelfReference" metatype="SelfReference" notes="" cx="90" cy="70"
bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="SituationParticipant" baseStereotypes="SituationParticipant">

```

```

<Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIACIcm0i3TuO+fQAAALoAAAAHABEAc3RyLmRhdFVUDQAhLQUgVv0FIFctBSBX
TY6xCsIWfEXPqUA/SCYLDroJrg5uDVodFbUGSsilNdSn9d0+jg8N9ee/13JsknpS03F1S213t
ahbMGKkdKdz0NoiIVKphReCs1mzUlh0F+x8deXh/NCLxsVaZjk4Hc6aMt+ctu4O5HS/nYELh
D9pMNO4nlyD37xkz93lp5ANQSwECFwsUAAAAACAAiAptIt07jvn0AAAC6AAAAABwAJAAAAAAAA
AAAAAIAAAAAAC3RyLmRhdFVUDQAhLQUgV1BLBQYAAAAAQAABAD4AAACzAAAAAA=</Image>
</Stereotype>
<Stereotype name="SituationParticipant" metatype="SituationParticipant" notes="" cx="90"
cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="Participant" baseStereotypes="Participant">
<Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIACIcm0i3TuO+fQAAALoAAAAHABEAc3RyLmRhdFVUDQAhLQUgVv0FIFctBSBX
TY6xCsIWfEXPqUA/SCYLDroJrg5uDVodFbUGSsilNdSn9d0+jg8N9ee/13JsknpS03F1S213t
ahbMGKkdKdz0NoiIVKphReCs1mzUlh0F+x8deXh/NCLxsVaZjk4Hc6aMt+ctu4O5HS/nYELh
D9pMNO4nlyD37xkz93lp5ANQSwECFwsUAAAAACAAiAptIt07jvn0AAAC6AAAAABwAJAAAAAAAA
AAAAAIAAAAAAC3RyLmRhdFVUDQAhLQUgV1BLBQYAAAAAQAABAD4AAACzAAAAAA=</Image>
<TaggedValues>
<Tag name="temporality" type="enumeration" description="" unit=""
values="present,past,any" default="present"/>
</TaggedValues>
</Stereotype>
<Stereotype name="SituationType" metatype="SituationType" notes="" cx="90" cy="70"
bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0">
<AppliesTo>
<Apply type="Package">
<Property name="URI" value=""/>
<Property name="_defaultDiagramType" value="SML::SML"/>
<Property name="_makeComposite" value="true"/>
</Apply>
</AppliesTo>
<TaggedValues>
<Tag name="isDerived" type="boolean" description="" unit="" values="true,false"
default=""/>
</TaggedValues>
</Stereotype>
<Stereotype name="SituationTypeAssociation" notes="" cx="90" cy="70" bgcolor="-1"
fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" isAbstract="true"
generalizes="SituationTypeElement" baseStereotypes="SituationTypeElement"
strictness="profile">
<TaggedValues>
<Tag name="isDerived" type="boolean" description="" unit="" values="true,false"
default=""/>
</TaggedValues>
</Stereotype>
<Stereotype name="SituationTypeComplex" metatype="SituationTypeComplex" notes="" cx="90"
cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="SituationType" baseStereotypes="SituationType">
<AppliesTo>
<Apply type="Package">
<Property name="URI" value=""/>
<Property name="_defaultDiagramType" value="SML::SML"/>
<Property name="_makeComposite" value="true"/>
</Apply>
</AppliesTo>
<TaggedValues>
<Tag name="isDerived" type="boolean" description="" unit="" values="true,false"
default="true"/>
</TaggedValues>
</Stereotype>
<Stereotype name="SituationTypeElement" notes="" cx="90" cy="70" bgcolor="-1"
fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0" isAbstract="true"
strictness="profile">
<AppliesTo>
<Apply type="Class">
<Property name="isActive" value=""/>
</Apply>
</AppliesTo>
<TaggedValues>
<Tag name="isDerived" type="boolean" description="" unit="" values="true,false"
default=""/>
</TaggedValues>
</Stereotype>
<Stereotype name="SituationTypeSimple" metatype="SituationTypeSimple" notes="" cx="90"
cy="70" bgcolor="-1" fontcolor="-1" bordercolor="-1" borderwidth="-1" hideicon="0"
generalizes="SituationType" baseStereotypes="SituationType">
<AppliesTo>
<Apply type="Package">
<Property name="URI" value=""/>
<Property name="_defaultDiagramType" value="SML::SML"/>

```

```

        <Property name="_makeComposite" value="true"/>
    </Apply>
</AppliesTo>
<TaggedValues>
    <Tag name="isDerived" type="boolean" description="" unit="" values="true,false"
default=""/>
</TaggedValues>
</Stereotype>
<Stereotype name="TestDiagram01" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0">
    <AppliesTo>
        <Apply type="Activity">
            <Property name="isReadOnly" value="false"/>
            <Property name="isSingleExecution" value=""/>
            <Property name="parameterName" value=""/>
            <Property name="postcondition" value=""/>
            <Property name="precondition" value=""/>
            <Property name="_defaultDiagramType" value="Extended::Analysis"/>
            <Property name="_makeComposite" value="true"/>
        </Apply>
    </AppliesTo>
</Stereotype>
<Stereotype name="TestDiagram02" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0">
    <AppliesTo>
        <Apply type="Class">
            <Property name="isActive" value=""/>
            <Property name="_defaultDiagramType" value="UML Structural::Class"/>
            <Property name="_makeComposite" value="true"/>
        </Apply>
    </AppliesTo>
</Stereotype>
<Stereotype name="TestDiagram03" notes="" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0">
    <AppliesTo>
        <Apply type="Package">
            <Property name="URI" value=""/>
            <Property name="_defaultDiagramType" value="SML::TestDiagram03"/>
            <Property name="_makeComposite" value="true"/>
        </Apply>
    </AppliesTo>
</Stereotype>
<Stereotype name="TypeLiteral" metatype="TypeLiteral" notes="A Literal either represents
a Quality's value (QualityLiteral, but can also represent a Datatype's value) such as $1000.00
or 37", or some Type (TypeLiteral) used in instantiation relations to indicate whether an
entity is or is not an instance of that class.&#xA;" cx="90" cy="70" bgcolor="-1" fontcolor="-1"
bordercolor="-1" borderwidth="-1" hideicon="0" generalizes="Literal"
baseStereotypes="Literal">
    <Image type="EAShapeScript 1.0" xmlns:dt="urn:schemas-microsoft-com:datatypes"
dt:dt="bin.base64">UESDBBQAAAAIAHqO/EiJ7R1lEwEAAFICAAHABEAc3RyLmRhZGVVUDQAHMUaaVzFGmlcxRppX
1ZFLS8RAEITrquB/CMk1CytuFkXFx8UH3jysd1E3xka2LkkUZM1/9+smJAOepOmZnumq6pnu
Vh960Va5Im2IstU60LS2vu6Rfea24r4gE+kKj/UGv1bH2nC6GLDf/8DmWoPrwL/CyPXOKSda
j7ya3Ia7ixOEbVGn3Zga6Vkl7FE7VAhlyfRj/8o0TUWazYq/GUnY50kQEYDfsV9p3s4FXaD
doU3MDOd61RzkEud4BadsU+17pxTURNFJdWhjkEttXBsxm5nYy1GVu+reRvMpfIuWXfCyZhq
p0f/UeG/sv+s8LnXyFcElP84VQPKLV6Yi0cXXK69b6VTM9mYuwY3UZf3vs2FgV9M9YoxMjp
g871+gVQSwECFwsUAAAAACAB6jvxiE0ZZRMBAABSAgAABwAJAAAAAAAAAAAAAAAAIAAAAAAC3Ry
LmRhZGVVUDQAHMUaaV1BLBQYAAAAAAQABAD4AAABJQAQAAAA=</Image>
    </Image>
</Stereotype>
</Stereotypes>
<TaggedValueTypes/>
</Content>
</UMLProfile>

```


Appendix E

FAIR Research Data Management

All artefacts produced during this research, such as the SEMIoTICS documentation, INTER-IoT-EWS code, publications and test data from the validations, are considered research data and were stored in a FAIR data repository, complying to the H2020 and 4TU data management plans.

As described in Chapter 2, the FAIR data point (FDP) is part of the FAIRport tools and leverages on linked data with RDF to expose data and metadata adhering to the FAIR data principles, specifying a set of standardized metadata that a data repository should implement. Data owners can expose datasets, and data users can reuse datasets through RESTful services, enabling interoperability in a web scale. Data repositories and their underlying software only recently started supporting LD, and their metadata are only available as key-value pairs. An open issue in this context is how to enable an existing data repository to comply to the FDP specification, i.e., how to add semantic descriptions to data repositories to ensure the semantic interoperability among data from different repositories?

This section describes the semantic proxy (“wrapper”) solution to enable a data repository, the EUDAT *B2share* service to behave as an FDP in a non-invasive and non-intrusive way, enabling the semantic interoperability through semantic translations. Our solution is based on the SEMIoTICS framework, following a methodology for metadata mapping based on endogenous model-driven transformations from lexicon to semantic models. We show how metadata in key-value pairs from a general-purpose repository can be made compliant with semantic technology without changing the repository software. Then, we describe how the data produced by this research are stored in EUDAT B2share repository and can be accessed through the FDP B2share semantic proxy.

Role of FAIR Data Repository

As described in Chapter 2, data repositories are usually designed to store and manage the outputs of research after the data are produced and the results are published. Therefore, they play a major role on allowing the reusability of research data, for which interoperability is required. The FDP metadata specification is based on standards and vocabularies, e.g., OAI-PMH, DC, DCAT and LDP. Therefore, the FDP specification aims at combining existing standards to meet the FAIR objectives. The FDP specification also gives a recommendation for the minimal metadata needed towards dataset findability (F) and it specifies (how) to declare a data license in order to be reusable (R). The dataset metadata contain links to one or multiple distribution and data record metadata.

The FDP specifies the minimal metadata for data repositories in five complementary layers: FDP Metadata (level 1), Data Catalog Metadata (level 2), Dataset Metadata (level 3), Distribution Metadata (level 4) and Data Record Metadata (level 5). The FDP implementation is optimized for machine discoverability and processing. (Meta)data access starts with the FDP URL, which resolves to metadata that describes the FDP and supports further exploration of the FDP by iterating through the different metadata levels. For example, the metadata include a link to the next FDP level (catalogs), which in turn links to the Dataset-level descriptions.

In a comparison study in which OAIS experts participated [131], several data repositories were inventoried and six were selected to be validated: EUDAT, *B2share*, CKAN, Zenodo, *Dspace*, *Eprints* and *Figshare*. Only CKAN is not natively compliant to OAI-PMH, but all of them provide APIs for exposing metadata records. This study concluded that selecting a data repository is a challenging task and should be based on technical choices. *EUDAT*, CKAN and *Dspace* presented open-source licenses that allow them to be customized and deployed in local servers instead of relying on external storage. Some of these data repositories have support for linked data to improve data interoperability. For example, RDF for CKAN is provided via plugins that allow it to expose and consume metadata according to the W3C DCAT [132]. In particular, the *EUDAT* services provide a way to integrate with CKAN and *Dspace*. *EUDAT* is a European initiative that devises technology and services to support the management of scientific data throughout the whole research process and beyond. It provides services for replicating (*B2safe*), publishing (*B2share*), sharing and versioning (*B2drop*) and searching (*B2find*) data. Furthermore, there are services for ad-hoc data transfers (*B2stage*), authentication and authorization (*B2access*) and for labeling data with persistent identifiers (*B2handle*).

B2share is the data repository service of *EUDAT* and provides the functionality to archive and publish the so-called long-tail data. Data are uploaded in deposits (records), i.e., datasets. A record is always annotated with some metadata, which are collected from the user at the time of upload to *B2share*. In *B2share*, a central

element is the community, which has the roles of creating and maintaining metadata schemas and curating the community datasets, enabling flexibility for domain-specific metadata. An end user of *B2share*, typically a researcher, can be part of one or more communities. A community can maintain metadata schemas but is always compliant to a generic metadata schema that is configured by the system administrator, having as default the EUDAT common metadata schema based on a superset of the DC elements. A community administrator has permission grants needed to change the metadata schema definitions and record curation tasks. A record in *B2share* is a “container” that adheres to both generic and the community metadata schemas, and to which any user can upload data files. A record submitted within a community must conform to the schemas in order to be accepted, i.e., published with an assigned Persistent Identifiers (PID), so that it becomes publicly accessible and cannot be changed. The other possible phase of a record is “draft”, i.e., immediately after its creation it is only accessible by its owner and can be modified (metadata can be changed and files can be uploaded). The community metadata schema is defined through a JSON message (lexicon approach) and can be retrieved through the B2share REST API [311]. We performed an ontological analysis to understand the B2share REST API, with the support of OntoUML, illustrated in Figure E.1, which helped on communicating with *B2share* experts and understanding the system.

Figure E.1. Conceptual model of B2share REST API described with OntoUML

A FAIR Semantic Proxy for B2share

Exposing new and existing datasets following the FDP specifications facilitates the interpretation and combination of heterogeneous and challenging types of research data. The goal of this effort, within the EUDAT-based FAIR project for data interoperability [312], was to implement and deploy a FDP using a combination of semantic web standards and the *B2share*. Therefore, we enabled an existing data repository (*B2share*) to behave as an FDP, i.e., compliant to the FDP specification by capturing metadata as an FDP. Since *B2share* provides a way to access data (a REST API), a requirement emerged to develop a non-invasive and non-intrusive solution, i.e., a solution that is totally decoupled of *B2share*. This is an important constraint since our solution could not directly access the database where the data were stored and obliged the solution to deal only with the API interfaces. Furthermore, some of the metadata recommended by the FDP are supported by *B2share*, thus, we identified mappings between *B2share* and FDP to align *B2share* terminology according to the FDP metadata layers, described in Table E.1.

Table E.1. Mappings between FDP levels and B2share interfaces

FAIR Data Point levels	B2share interfaces	Description
FAIR Data Point (L1)	B2share system	A FDP is a data repository specification based on re3data.org, which defines B2share as a data repository [313].
Group / catalog (L2)	Community	A catalog (FDP) is a way to categorize, i.e., to create a taxonomy of, datasets. The basic categories (facets) of B2share are the Communities.
Dataset (L3)	Record	A dataset (FDP) is comprised of parts of data that can be manipulated as a unit, which roughly corresponds to a record in B2share.
Distribution (L4)	File	A distribution (FDP) is a resource that gives access to the data, similarly to files.

A common approach to implement data transformations based on mappings is the proxy pattern. A proxy is a programming design pattern, which is implemented as a wrapper that is called by a client and performs the access to the real intended service. It can be used to simply forward original data from the client to the service or it can process the data, providing additional logic [314]. Therefore, our solution adopted this pattern to access data from B2share, executing the mappings and delivering the data to the client according to the FDP specifications. To access the B2share data we used a REST API, accessing the data in a non-invasive and non-intrusive way.

The proxy was improved with semantic technologies, turning it into a semantic proxy that gathers data from B2share and delivers to the clients as RDF triples. The

solution applies horizontal and endogenous model-driven transformations, mapping the B2share lexicon model, which is serialized as key-value pairs in JSON to an RDF model that is delivered as JSON-LD. The FDP metadata specification already prescribes the use of RDF, making this solution appropriate. Therefore, the FDP B2share solution is a proxy that implements the a kind of semantic translations¹⁶⁸ on data collected from the B2share REST API (source) at runtime on each client call to FDP compliant data. This method was leveraged by the practice of ODCM. For example, when analyzing the identity of *dct:title* metadata as the name property of a catalog, it is clear that this metadata should be mapped to the *Community name* (*B2share.Community.name*) property.

Therefore, for each FDP level, the procedure used was:

- 1) Pre-analysis of all B2share metadata of an entity and all metadata of the equivalent FDP level
- 2) For each metadata field of the FDP level (ordering by the required ones)
 - a) Understand the FDP metadata description
 - b) Search the similar concept in B2share and create the mapping. Here the mapping can be either straightforward, as the case of *B2share.Community.name* with *FDP.Catalog.dct:title*; or multiple, which may require the definition of transformation rules
- 3) Check if there are required FDP metadata not be mapped, which are crucial gaps of the data source, necessary to change the data source metamodel or set a default value
- 4) Check each metadata field of the B2share entity that was not mapped
 - a) Search the similar concept in common and, preferably, standardized ontologies
 - i) If it is found, then create the mapping
 - ii) If it is not found, i.e., the concept is not present in other ontologies (e.g., it is a domain-specific conceot), it is required to represent this concept in an ontology. An important issue here is the need of only creating the elements in the ontology that are not found in the FDP specification and neither in other ontologies.

¹⁶⁸ Although the outcome of these translations are semantic enriched data, this method cannot be considered as semantic translations because the data source is represented as a lexicon data model (B2share), serialized as JSON.

Figure E.2. Example of the semantic proxy mappings for FDP level 2 (catalog)

Figure E.2 illustrates the mappings created between *Community* (B2share) and *Catalog* (FDP level 2) according to this methodology. Several mappings were straightforward and some of the B2share metadata could not be mapped to FDP (step 4), so existing ontologies were reused, as FOAF and the Publishing Role Ontology (PRO). Some B2share specific metadata, e.g., *publication_workflow* and *restricted_submission*, required to be represented with an ontology. Thus, we created the B2share ontology, where *publication_workflow* is an object property and *restricted_submission* is a data property of *Community* class. Fortunately, we found no required FDP metadata that could not be mapped onto B2share metadata. Our mappings were specified at design time for all four levels and discussed (harmonized) with the B2share and FDP experts.

All components of the solution, including the B2share ontology, the proxy code and test data are available in [315]. The solution was implemented in Python 3 with PyLD library [316], a JSON-LD API for RDF. Along with *jsonld.js* (the JavaScript library), *PyLD* is one of the most popular JSON-LD library implementations with a strong support from the community. The core part of the implementation is the translations module (*translators.py*) with the methods for each translation. For example, Figure 9.9 illustrates the translation method from a *Community* to a *Catalog*, showing all ontologies from the FDP specification in the context element, as well as B2share ontology and PRO, used to deal with the role-based access control of B2share. In addition, the properties are instantiated according to the mappings from the community element.

```

# B2SHARE: Community
# Level 2: Catalog metadata layer
def translate_catalog(community):
    AdminRole = {
    }
    MemberRole = {
    }

    context = {
        # ontologies used in FDP according to spec
        "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
        "rdfs" : "http://www.w3.org/2000/01/rdf-schema#",
        "dcat" : "http://www.w3.org/ns/dcat#",
        "xsd" : "http://www.w3.org/2001/XMLSchema#",
        "owl" : "http://www.w3.org/2002/07/owl#",
        "dct" : "http://purl.org/dc/terms/",
        "lang" : "http://id.loc.gov/vocabulary/iso639-1/",
        "fdp" : "http://rdf.biosemantics.org/ontologies/fdp-c#",
        "foaf" : "http://xmlns.com/foaf/",
        # B2SHARE otology (internal terms)
        "b2" : "https://b2share.eudat.eu/ontology/b2share/",
        # Other ontologies (reused)
        "pro" : "http://purl.org/spar/pro/" # Publishing Roles
    }

    doc = {
        "@id": community.links.selflink,
        "@type": "dcat:Catalog",
        "http://purl.org/dc/terms/identifier": community.identifier,
        "http://purl.org/dc/terms/title": community.name,
        "http://purl.org/dc/terms/description": community.description,
        "http://purl.org/dc/terms/issued": community.created,
        "http://purl.org/dc/terms/modified": community.updated,
        "http://xmlns.com/foaf/logo" : community.logo,
        "https://b2share.eudat.eu/ontology/b2share/publication_workflow" : community.publication_workflow,
        "https://b2share.eudat.eu/ontology/b2share/restricted_submission" : community.restricted_submission,
        "https://b2share.eudat.eu/ontology/b2share/AdminRole" : AdminRole,
        "https://b2share.eudat.eu/ontology/b2share/MemberRole" : MemberRole
    }
    compacted = jsonld.compact(doc, context)
    return compacted

```

Figure E.3. Translation method from Community to a Catalog (FDP level 2).

We assessed the mappings by exhaustively revisiting them with specialists of B2share and FDP. The REST API functional validation was performed on the *B2share* test environment¹⁶⁹ executing the translations for each level through client calls on the endpoints of FDP *B2share* REST API. The first level (data repository) translated the information from the B2share data repository (the test instance) to the FDP level 1, illustrated in Figure 5(a). For the second level (catalog), each community registered in *B2share* was translated. Figure 5(b) shows the result of the translation of Aalto University community as a catalog. The validation of FDP level 3 (dataset) considered both the direct access to a dataset from INTER-IoT-EWS tests, e.g., the dataset UC01_ST01, as well as dataset search feature. Finally, FDP level 4 (distribution) was validated through the JSON-LD files (messages) within each INTER-IoT-EWS test case. For each translation method, an equivalent assertion method was developed to verify whether a mapping rule has been broken after executing.

The impact on the performance of the service calls was also measured, since a performance overhead is expected when a proxy solution is applied. The total time transaction, i.e., total time taken from the client request to the response, was measured in two situations: (1) without the solution, to serve as a baseline for

¹⁶⁹ <https://trng-B2share.eudat.eu/api/>

comparison; and (2) with the solution. Each FDP level was tested, and scripts were created to compute the total time transaction for each situation (1 and 2), comparing them in each execution. To statistically measure this variable and avoid noise caused by network issues, synchronous calls (10) were executed 10, 50, 100 times and an anomaly detection algorithm removed the noisy measurements. All test scripts are available¹⁷⁰. The test environment of the semantic proxy is a local virtual machine running Ubuntu 16.04 with 4GB RAM, 1 CPU, proxying to the B2share test environment.

Figure E.4. Overhead of the proxy compared to direct access to *B2share*

The results of the execution of each test case are illustrated in Figure E.4, which presents the FDP B2share minimum and maximum overhead difference in terms of percentage of the total time transaction between the two situations (with and without the solution). For example, TC1 refers to tests with FDP level 1, having the minimum difference computed as 6.13%, i.e., on average FDP B2share is 6.13% slower than directly accessing the B2share endpoint. TC1, TC4, TC5, TC6 and TC7 presented acceptable differences, impacting less than 20% (TC5 maximum difference). However, TC2 and TC3 presented a huge difference, with the worst-case of almost 43% slower (TC2 maximum difference). This difference comes from the programming approach for accessing indexed Python object arrays mapped from the JSON object (input). In contrast with the other levels in which objects were retrieved by iterating arrays, in level 2 we used our own index-based solution, which presented poor performance. Besides this issue, a validity threat, which affected the results negatively, is that both the proxy and the input test scripts were deployed in the same test environment and executed in parallel. Therefore, the differences are expected to be smaller in a distributed production

¹⁷⁰ <https://github.com/jonimoreira/B2SHARE-FAIR/tree/master/src/proxy/tests>

environment. A possible solution to address critical performance constraints is to implement caching mechanisms in the proxy server.

Conclusions

We applied a FAIR-based research data management, introducing an approach to facilitate the shift from document-based data repository to linked data. Our methodology enabled a document-oriented data repository (EUDAT B2share) to behave as a FAIR Data Point through semantic mappings. Our solution follows the proxy pattern, a wrapper, implementing translations over research data described with B2share lexicon data model (as JSON) to FDP compliant RDF (as JSON-LD). This is a decoupled approach that did not require changes in B2share code. We demonstrated how the data produced by this research were stored in B2share and exposed as RDF triples, enabling semantic interoperability of the data repository. The performance validation of the semantic proxy showed that it has a small impact on the total transaction time of client requests.

Lessons learned on FAIR research data management include the need of a strict methodology to align the metadata used in the data repository to the FDP metadata layers. This is a challenging task, especially in B2share, which enables the community and system administrators to add new metadata “on the fly” (at runtime). We observed that the practice of ontological analysis adds a significant value when analyzing how a data repository is structured.

The FDP level 5 could be addressed as a consequence of the application of SEMIoTICS framework, which already delivers data in compliance with the FAIR data principles. In our solution, MDE played an important role to implement the translations as endogenous model-driven transformations. Finally, we could store the research data with different levels of accessibility, such as the private and sensitive data from the INTER-IoT-EWS test data, following the H2020 data management plan, which is compliant to the EU General Data Protection Regulation (GDPR). The same approach can be applied to other data repositories, such as CKAN, Dspace and Zenodo.

References

1. Othman, S.H., G. Beydoun, and V. Sugumaran, *Development and validation of a Disaster Management Metamodel (DMM)*. Information Processing and Management, 2014.
2. Cvetkovich, G. and T.C. Earle, *Classifying hazardous events*. Journal of Environmental Psychology, 1985. **5**(1): p. 5-35.
3. Bartels, S.A., et al., *Chapter 7 - Complex Emergencies A2 - Ciottone, Gregory R*, in *Ciottone's Disaster Medicine (Second Edition)*. 2016, Elsevier: Philadelphia. p. 40-46.
4. UNISDR. *Disaster statistics*. United Nations International Strategy for Disaster Reduction 2013 31/05/2018]; Available from: <https://www.unisdr.org/we/inform/disaster-statistics>.
5. Rifino, J.J. and S.E. Mahon, *Chapter 3 - Role of Emergency Medical Services in Disaster Management and Preparedness A2 - Ciottone, Gregory R*, in *Ciottone's Disaster Medicine (Second Edition)*. 2016, Elsevier: Philadelphia. p. 13-19.
6. UNISDR. *What is Disaster Risk Reduction?* United Nation International Strategy for Disaster Reduction 2016 1/1/2016]; Available from: <https://www.unisdr.org/who-we-are/what-is-drr>.
7. UN, *Global Survey of Early Warning Systems: An assessment of capacities, gaps and opportunities toward building a comprehensive global early warning system for all natural hazards*. 2006. p. 56-56.
8. UNISDR, *Developing Early Warning Systems : A Checklist - From concept to action*, in *Third International Conference on Early Warning (EWC III)*. 2006.
9. Mescherin, S.A., I. Kirillov, and S. Klimenko. *Ontology of Emergency Shared Situation Awareness and Crisis Interoperability*. in *Cyberworlds (CW), 2013 International Conference on*. 2013.
10. Middleton, S., et al., *The seven main challenges of an early warning system architecture*, in *Information Systems for Crisis Response and Management (ISCRAM)*. 2013.
11. Li, J., et al., *Community-based collaborative information system for emergency management*. Computers & Operations Research, 2014. **42**: p. 116-124.
12. Zambrano, A.M., et al., *Technologies of Internet of Things applied to an Earthquake Early Warning System*. Future Generation Computer Systems, 2017. **75**: p. 206-215.
13. Zambrano, A.M., et al. *Sensor Web Enablement applied to an earthquake early warning system*. in *International Conference on Internet and Distributed Computing Systems (IDCS)*. 2015.
14. Poslad, S., et al., *A Semantic IoT Early Warning System for Natural Environment Crisis Management*. IEEE Transactions on Emerging Topics in Computing, 2015. **3**(2): p. 246-257.

15. UNEP, *Early warning systems: State-of-art analysis and future directions*. 2012, United Nations Environment Programme Publication. p. 1-70.
16. Wächter, J. and T. Usländer, *The Role of Information and Communication Technology in the Development of Early Warning Systems for Geological Disasters: The Tsunami Show Case*, in *Early Warning for Geological Disasters: Scientific Methods and Current Practice*, F. Wenzel and J. Zschau, Editors. 2014, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 227-252.
17. Waidyanatha, N., *Towards a typology of integrated functional early warning systems*. International Journal of Critical Infrastructures, 2010. **6**: p. 31-51.
18. Wang, J., H. Li, and H. Lu, *Application of a novel early warning system based on fuzzy time series in urban air quality forecasting in China*. Applied Soft Computing, 2018.
19. da Costa, C.A., et al., *Internet of Health Things: Toward intelligent vital signs monitoring in hospital wards*. Artificial Intelligence in Medicine, 2018.
20. Du, C. and S. Zhu, *Research on Urban Public Safety Emergency Management Early Warning System based on Technologies for the Internet of Things*. Procedia Engineering, 2012. **45**: p. 748-754.
21. Balis, B., et al., *The UrbanFlood Common Information Space for Early Warning Systems*. Procedia Computer Science, 2011. **4**: p. 96-105.
22. Hartmann, T., et al. *The Next Evolution of MDE: A Seamless Integration of Machine Learning into Domain Modeling*. in *2017 ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 2017.
23. Kim, J.E., et al. *Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes*. in *2012 Eighth International Conference on Intelligent Environments*. 2012.
24. Ninomiya, S., et al. *Seamless Integration of Sensor Network and Legacy Weather Databases by MetBroker*. in *2007 International Symposium on Applications and the Internet Workshops*. 2007.
25. Heiler, S., *Semantic interoperability*. ACM Comput. Surv., 1995. **27**.
26. EC. *New European Interoperability Framework at the website of the European Commission, ISA2 programme*. 2017 31/05/2018]; Available from: https://ec.europa.eu/isa2/eif_en.
27. Rezaei, R., T.K. Chiew, and S.P. Lee, *An interoperability model for ultra large scale systems*. Advances in Engineering Software, 2014. **67**: p. 22-46.
28. Daclin, N., et al., *Writing and verifying interoperability requirements: Application to collaborative processes*. Computers in Industry, 2016. **82**: p. 1-18.
29. UNISDR, *Sendai Framework for Disaster Risk Reduction 2015 - 2030*, in *United Nations International Strategy for Disaster Reduction*. 2015.
30. WHO, et al., *International Network for Multi-Hazard Early Warning Systems (IN-MHEWS)*. 2015.
31. Zwolski, K., *Integrating crisis early warning systems: power in the community of practice*. Journal of European Integration, 2016. **38**(4): p. 393-407.
32. Fang, S., et al., *An integrated system for regional environmental monitoring and management based on internet of things*. IEEE Transactions on Industrial Informatics, 2014. **10**(2): p. 1596-1605.
33. Wang, Y., et al., *A fast mobile early warning system for water quality emergency risk in ungauged river basins*. Environmental Modelling & Software, 2015. **73**: p. 76-89.

34. Ding, L.Y., et al., *Real-time safety early warning system for cross passage construction in Yangtze Riverbed Metro Tunnel based on the internet of things*. Automation in Construction, 2013. **36**: p. 25-37.
35. Keeney, H.J., S. Buan, and L. Diamond, *Multi-Hazard Early Warning System of the United States National Weather Service*, in *Institutional Partnerships in Multi-Hazard Early Warning Systems: A Compilation of Seven National Good Practices and Guiding Principles*, M. Golnaraghi, Editor. 2012, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 115-157.
36. Alamdar, F., M. Kalantari, and A. Rajabifard, *Towards multi-agency sensor information integration for disaster management*. Computers, Environment and Urban Systems, 2016. **56**: p. 68-85.
37. Raape, U., et al., *Decision Support for Tsunami Early Warning in Indonesia: The Role of OGC Standards*, in *Geographic Information and Cartography for Risk and Crisis Management: Towards Better Solutions*, M. Konecny, S. Zlatanova, and L.T. Bandrova, Editors. 2010, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 233-247.
38. Chronaki, C.E., et al., *Interoperability in Disaster Medicine and Emergency Management*. Journal of Health Informatics, 2011. **3**: p. 87-99.
39. Ye, J., S. Dobson, and S. McKeever, *Situation identification techniques in pervasive computing: A review*. Pervasive and Mobile Computing, 2012. **8**(1): p. 36-66.
40. Dogac, A., et al., *RECONSURVE : JC3IEDM and EDXL based Emergency Management Service Oriented Architecture for Maritime Surveillance*. 2012.
41. McGarry, D.P. and C.Y.R. Chen. *IC.NET — Incident Command “Net”: A system using EDXL-DE for intelligent message routing*. in *2010 IEEE International Conference on Technologies for Homeland Security (HST)*. 2010.
42. Lendholt, M., M.Á. Esbrí, and M. Hammitzsch, *Interlinking National Tsunami Early Warning Systems towards Ocean-Wide System-of-Systems Networks*. Proceedings of the 9th International ISCRAM Conference - Vancouver, Canada, 2012(April): p. 1-10.
43. Poulymenopoulou, M., F. Malamateniou, and G. Vassilacopoulos, *Emergency Healthcare Process Automation Using Mobile Computing and Cloud Services*. Journal of Medical Systems, 2012. **36**(5): p. 3233-3241.
44. Waidyanatha, N., et al., *Complexity and Usability of Voice-enabled Alerting and Situational Reporting Decoupled Systems*. Proceedings of the 9th International Conference on Information Systems for Crisis Response and Management, 2012(April 2012): p. 1-10.
45. Weinert, A.J., et al. *Providing communication capabilities during disaster response: Airborne remote communication (ARC) platform*. in *2012 IEEE Conference on Technologies for Homeland Security (HST)*. 2012.
46. Barros, R., et al. *EDXL-RESCUER ontology: An update based on faceted taxonomy approach*. in *CEUR Workshop Proceedings*. 2015.
47. Olivieri, A.C., Y. Bocchi, and G. Rizzo, *Chapter 4 - Integration in the Internet of Things: A semantic middleware approach to seamless integration of heterogeneous technologies*, in *Pervasive Computing*, C. Dobre and F. Xhafa, Editors. 2016, Academic Press: Boston. p. 97-128.
48. Bernabé-Moreno, J., et al., *Leveraging Localized Social Media Insights for Early Warning Systems*. Procedia Computer Science, 2014. **31**: p. 1051-1060.

290 REFERENCES

49. Roy, C., et al., *The current cyclone early warning system in Bangladesh: Providers' and receivers' views*. International Journal of Disaster Risk Reduction, 2015. **12**: p. 285-299.
50. Battistini, A., et al., *Validation of landslide hazard models using a semantic engine on online news*. Applied Geography, 2017. **82**: p. 59-65.
51. Yang, M., M. Kiang, and W. Shang, *Filtering big data from social media – Building an early warning system for adverse drug reactions*. Journal of Biomedical Informatics, 2015. **54**: p. 230-240.
52. Lowe, P., et al., *The evolution of disaster early warning systems in the TRIDEC project*, in *ISOPE 2013*. 2013.
53. Palmer, C., et al., *An ontology supported risk assessment approach for the intelligent configuration of supply networks*. Journal of Intelligent Manufacturing, 2018. **29**(5): p. 1005-1030.
54. Desai, P., A. Sheth, and P. Anantharam. *Semantic Gateway as a Service Architecture for IoT Interoperability*. in *2015 IEEE International Conference on Mobile Services*. 2015.
55. Ullah, F., et al., *Semantic interoperability for big-data in heterogeneous IoT infrastructure for healthcare*. Sustainable Cities and Society, 2017. **34**: p. 90-96.
56. Ramar, K. and T.T. Mirnalinee. *An ontological representation for Tsunami early warning system*. in *IEEE-Advances in Engineering, Science and Management (ICAESM)*. 2012.
57. Scherp, A., et al., *A core ontology on events for representing occurrences in the real world*. Multimedia Tools and Applications, 2012. **58**(2): p. 293-331.
58. Kim, J., et al., *A semantic translation method for data communication protocols*. Journal of Systems and Software, 2012. **85**(12): p. 2876-2898.
59. Solbrig, H.R., et al., *Modeling and validating HL7 FHIR profiles using semantic web Shape Expressions (ShEx)*. Journal of Biomedical Informatics, 2017. **67**: p. 90-100.
60. Azmani, S., et al., *CHALLENGES OF COMMUNICATION SYSTEM DURING EMERGENCY DISASTER RESPONSE IN MALAYSIA: A REVIEW*. 2017, 2017. **9**(4S): p. 15.
61. Clinton, J., et al., *State-of-the art and future of earthquake early warning in the European region*. Bulletin of Earthquake Engineering, 2016. **14**(9): p. 2441-2458.
62. Wilkinson, M.D., et al., *The FAIR Guiding Principles for scientific data management and stewardship*. Nature, 2016. **3**: p. 160018.
63. Kitchenham, B. and S. Charters, *Guidelines for performing Systematic Literature Reviews in Software Engineering*. Engineering, 2007. **2**: p. 1051.
64. Wieringa, R., *Design Science Methodology for Information Systems and Software Engineering*. Springer Berlin Heidelberg, 2014: p. 493.
65. Wohlin, C., et al., *Experimentation in Software Engineering*. 2012: Springer Publishing Company, Incorporated. 259.
66. INTER-IoT, *INTER-IoT deliverable: D2.4. Use cases manual*. 2016.
67. Moreira, J., L.F. Pires, and M.V. Sinderen. *Semantic interoperability of Early Warning Systems: a systematic literature review*. in *Enterprise Interoperability IX: I-ESA proceedings. ICRIM workshop*. 2018.
68. Moreira, J., et al., *Developing situation-aware applications for disaster management with a distributed rule-based platform*, in *RuleML 2015 Doctoral Consortium*. 2015: Berlin, Germany.

69. Moreira, J.L.R., et al., *Improving semantic interoperability of big data for epidemiological surveillance.*, in *I-ESA, BDI4E workshop*. 2016.
70. Moreira, J., et al. *Menthor editor: An ontology-driven conceptual modeling platform.* in *9th International Conference on Formal Ontology in Information Systems (FOIS 2016) - Demonstration Track*. 2016. Annecy, France: CEUR Workshop Proceedings.
71. Ferreira, M.I.G.B., et al. *OntoEmergePlan: variability of emergency plans supported by a domain ontology.* in *12th International Conference on Information Systems for Crisis Response and Management (ISCRAM)*. 2015.
72. Moreira, J.L.R., et al., *Towards ontology-driven situation-aware disaster management.* *Journal of applied ontology*, 2015. **10**(3-4): p. 339-353.
73. Moreira, J.L.R., et al. *Ontology-driven Conceptual Modeling for Early Warning Systems: Redesigning the Situation Modeling Language.* in *MODELSWARD*. 2017.
74. Moreira, J.L.R., et al. *Towards IoT platforms' integration: Semantic Translations between W3C SSN and ETSI SAREF.* in *Semantics. Workshop SIS-IoT*. 2017.
75. Moreira, J., et al. *Improving the semantic interoperability of IoT Early Warning Systems: the Port of Valencia use case.* in *Enterprise Interoperability IX: I-ESA proceedings*. 2018.
76. Moreira, J., L.F. Pires, and M.V. Sinderen. *SAREF4health: IoT standard-based ontology-driven healthcare systems.* in *10th International Conference on Formal Ontology in Information Systems (FOIS 2018). Best paper award*. 2018. Cape Town, South Africa.
77. Moreira, J., L.F. Pires, and M.V. Sinderen. *Semantic interoperability for the IoT: Analysis of JSON for Linked Data.* in *Enterprise Interoperability IX: I-ESA proceedings. BD4EI workshop*. 2018.
78. Way, S.C., *REQUIREMENTS ANALYSIS FOR A CONTEXT-AWARE MULTI-AGENCY EMERGENCY RESPONSE SYSTEM*. 2013, McMaster University.
79. Coppola, D.P., *Chapter 1 - The Management of Disasters*, in *Introduction to International Disaster Management (Third Edition)*. 2015.
80. Wood, L., B. Boruff, and H. Smith, *When disaster strikes ... how communities cope and adapt: A social capital perspective*. In book: *Social Capital: Theory, Measurement and Outcomes*, 2013. **Chapter: 6**.
81. Casado, R., et al., *Data interoperability software solution for emergency reaction in the Europe Union.* *Journal of Natural Hazards and Earth System Science*, 2015. **15**: p. 1563-1576.
82. Hristidis, V., et al., *Survey of data management and analysis in disaster situations.* *Journal of Systems and Software*, 2010. **83**: p. 1701-1714.
83. Othman, S.H., G. Beydoun, and V. Sugumaran, *Development and validation of a Disaster Management Metamodel (DMM)*. *Journal of Information Processing and Management*, 2014.
84. Sena, L., *Disaster Prevention and Preparedness*. LECTURE NOTES For Health Science Students ed. T.C.C. In collaboration with the Ethiopia Public Health Training Initiative, the Ethiopia Ministry of Health, and the Ethiopia Ministry of Education. 2006.
85. Gian Luca, F., F. Manuela, and V. Marco, *Advanced Emergency Response Management in Smart Environments*, in *Encyclopedia of Information Science and Technology, Third Edition*, D.B.A. Mehdi Khosrow-Pour, Editor. 2015, IGI Global: Hershey, PA, USA. p. 1337-1349.

292 REFERENCES

86. Chen, R., et al., *Design principles for critical incident response systems*. Information Systems and e-Business Management, 2007. 5(3): p. 201-227.
87. Endsley, M.R., *Toward a theory of situation awareness in dynamic systems*. Human factors and Ergonomics Society, 1995.
88. Wickens, C., *Situation awareness: Review of Mica Endsley's 1995 articles on situation awareness theory and measurement*. The Journal of the Human Factors, 2008: p. 397-403.
89. Foo, P. and G. Ng, *High-level Information Fusion: An Overview*. Journal of Advances in Information Fusion, 2013. 8.
90. Barwise, J., *The Situation in Logic—I*, in *Studies in Logic and the Foundations of Mathematics*, R. Barcan Marcus, G.J.W. Dorn, and P. Weingartner, Editors. 1986, Elsevier. p. 183-203.
91. Barwise, J., *The Situation in Logic*. CSLI Lecture Notes 17, 1988.
92. Matheus, C.J., M.M. Kokar, and K. Baclawski. *A core ontology for situation awareness*. in *Sixth International Conference of Information Fusion, 2003. Proceedings of the*. 2003.
93. Van de Walle, B., B. Brugghe-mans, and T. Comes, *Improving situation awareness in crisis response teams: An experimental analysis of enriched information and centralized coordination*. International Journal of Human-Computer Studies, 2016. 95: p. 66-79.
94. Foo, P.H. and G.W. Ng, *High-level information fusion: An overview*. Journal of Advances in Information Fusion, 2013. 8(1): p. 33-72.
95. Zambrano, O.M., et al., *2 - An Innovative and Economic Management of Earthquakes: Early Warnings and Situational Awareness in Real Time*, in *Wireless Public Safety Networks 3*, D. Câmara and N. Nikaein, Editors. 2017, Elsevier. p. 19-38.
96. Franke, U. and J. Brynielsson, *Cyber situational awareness – A systematic review of the literature*. Computers & Security, 2014. 46: p. 18-31.
97. Baudoin, M.-A., et al., *From Top-Down to “Community-Centric” Approaches to Early Warning Systems: Exploring Pathways to Improve Disaster Risk Reduction Through Community Participation*. International Journal of Disaster Risk Science, 2016. 7(2): p. 163-174.
98. Amaratunga, D., et al., *Effectiveness of Dissemination and Communication Element of Tsunami Early Warning System in Aceh*, in *4th International Conference on Building Resilience, Incorporating the 3rd Annual Conference of the ANDROID Disaster Resilience Network, 8th – 11th September 2014, Salford Quays, United Kingdom*. 2014. p. 136-142.
99. Horn, D.P., *Chapter 6 - Storm Surge Warning, Mitigation, and Adaptation A2 - Shroder, John F*, in *Coastal and Marine Hazards, Risks, and Disasters*, J.T. Ellis and D.J. Sherman, Editors. 2015, Elsevier: Boston. p. 153-180.
100. Orford, R., et al., *EU alerting and reporting systems for potential chemical public health threats and hazards*. Environment International, 2014. 72: p. 15-25.
101. Wu, X., et al., *Impact of climate change on human infectious diseases: Empirical evidence and human adaptation*. Environment International, 2016. 86: p. 14-23.
102. Fang, Y., C. Li, and L. Sun, *Design of an Early Warning System for Patients with Cardiovascular Diseases Under Mobile Environment*. Procedia Computer Science, 2016. 96: p. 819-825.

103. Yu, V.L. and L.C. Madoff, *ProMED-mail: An Early Warning System for Emerging Diseases*. Clinical Infectious Diseases, 2004. **39**(2): p. 227-232.
104. Comes, T., B. Mayag, and E. Negre. *Decision Support for Disaster Risk Management: Integrating Vulnerabilities into Early-Warning Systems*. in *Information Systems for Crisis Response and Management in Mediterranean Countries (ISCRAM-med)*. 2014. Cham: Springer International Publishing.
105. Hmoudi, A.A. and Z. Aziz. *Challenges for integrated early warning and disaster management in the UAE*.
106. UNEP, *Early Warning Systems A State of the Art Analysis and Future Directions*. Environmental Development, 2012. **4**: p. 136-171.
107. IEEE, *IEEE Standard Glossary of Software Engineering Terminology*, in *IEEE Std 610.12-1990*. 1990: New York, NY.
108. European, C., *European Interoperability Framework (EIF) for European public services*. Com (2010), 2010: p. 1-40.
109. Steenbruggen, J., et al., *Traffic incident and disaster management in the Netherlands: Challenges and obstacles in information sharing*. Research Memorandum, 2013. **2013**: p. 24.
110. Rezaei, R., T.K. Chiew, and S.P. Lee, *An interoperability model for ultra large scale systems*. Advances in Engineering Software, 2014.
111. Heiler, S., *Semantic interoperability*. ACM Comput. Surv., 1995. **27**(2): p. 271-273.
112. Noran, O. and P. Bernus, *Effective Disaster Management: An Interoperability Perspective*, in *On the Move to Meaningful Internet Systems: OTM 2011 Workshops: Confederated International Workshops and Posters: EI2N+NSF ICE, ICSP+INBAST, ISDE, ORM, OTMA, SWWS+MONET+SeDeS, and VADER 2011, Hersonissos, Crete, Greece, October 17-21, 2011. Proceedings*, R. Meersman, T. Dillon, and P. Herrero, Editors. 2011, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 112-121.
113. Benaben, F., et al., *Supporting interoperability of collaborative networks through engineering of a service-based Mediation Information System (MISE 2.0)*. Enterprise Information Systems, Taylor & Francis: STM, Behavioral Science and Public Health Titles, 2015.
114. Gencturk, M., et al. *Interoperability profiles for disaster management and maritime surveillance*. in *eChallenges e-2015 Conference Proceedings*. 2015.
115. Rieser, H., et al. *Sensor interoperability for disaster management*. in *SAS 2015 - 2015 IEEE Sensors Applications Symposium, Proceedings*. 2015.
116. Zdravković, M., et al., *Enabling interoperability as a property of ubiquitous systems for disaster management*. Computer Science and Information Systems, 2015. **12**(3): p. 1009-1031.
117. Daniele, L., et al., *Interoperability for Smart Appliances in the IoT World*, in *The Semantic Web – ISWC 2016: 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II*, P. Groth, et al., Editors. 2016, Springer International Publishing: Cham. p. 21-29.
118. ISA. *European Interoperability Reference Architecture (EIRA) v.1.0.0*. 2016; Available from: https://joinup.ec.europa.eu/asset/cia/asset_release/eira-v100.
119. Janes, D.P. *Semantic Metastandards Will Unlock IoT Interoperability*. 2016. Cham: Springer International Publishing.

120. Auer, S. *Interoperability and Aggregation of Research Data*. in *SEMANTICS*. 2017.
121. Wilkinson, M.D., et al., *Interoperability and FAIRness through a novel combination of Web technologies*. *PeerJ Computer Science*, 2017. **3**: p. e110.
122. Cosío-León, M.A., et al., *The use of standards in embedded devices to achieve end to end semantic interoperability on health systems*. *Computer Standards & Interfaces*, 2018. **57**: p. 68-73.
123. OMG, *Information Exchange Packaging Policy Vocabulary* TM, in *Version 1.0* 2015.
124. Guarino, N., D. Oberle, and S. Staab, *What Is an Ontology?*, in *Handbook on Ontologies*, S. Staab and R. Studer, Editors. 2009, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 1-17.
125. Neiva, F.W., et al., *Towards pragmatic interoperability to support collaboration: A systematic review and mapping of the literature*. *Information and Software Technology*, 2016. **72**: p. 137-150.
126. Asuncion, C.H. and M. van Sinderen. *Towards Pragmatic Interoperability in the New Enterprise — A Survey of Approaches*. in *Enterprise Interoperability*. 2011. Berlin, Heidelberg: Springer Berlin Heidelberg.
127. Hodson, S., et al., *Turning FAIR data into reality: interim report from the European Commission Expert Group on FAIR data (Version Interim draft)*. 2018(June).
128. Imming, M., *FAIR Data Advanced Use Cases: from principles to practice in the Netherlands*. 2018.
129. GO-FAIR. *GO FAIR: a bottom-up international approach*. 2018 05-05-2018]; Available from: <https://www.go-fair.org/>.
130. Wilkinson, M.D., et al., *A design framework and exemplar metrics for FAIRness*. *bioRxiv*, 2017.
131. Amorim, R.C., et al., *A comparison of research data management platforms: architecture, flexible metadata and interoperability*. *Universal Access in the Information Society*, 2017. **16**(4): p. 851-862.
132. CKAN. *Linked data and RDF features for CKAN*. 2018 05-05-2018]; Available from: <http://docs.ckan.org/en/latest/maintaining/linked-data-and-rdf.html>.
133. Brambilla, M., J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*. 2012: Morgan & Claypool Publishers. 182.
134. Andrade Almeida, J.P., *Model-driven design of distributed applications*, in *Enschede : Centre for Telematics and Information Technology (CTIT)*. 2006, University of Twente. p. 225.
135. Hodges, W., *A shorter model theory*. 1997: Cambridge University Press. 310.
136. Amato, F., et al., *A Model Driven Approach to Water Resource Analysis based on Formal Methods and Model Transformation*. *Procedia Computer Science*, 2015. **51**: p. 562-571.
137. Andrade Almeida, J.P., *Model-driven design of distributed applications*, in *CTIT Ph.D.-Thesis Series, ISSN 1381-3617, No. 06-85 ; Telematica Instituut Fundamental Research Series, ISSN 1388-1795, No. 018*. 2006, University of Twente, Centre for Telematics and Information Technology (CTIT): Enschede.
138. Brambilla, M. and P. Fraternali, *Large-scale Model-Driven Engineering of web user interaction: The WebML and WebRatio experience*. *Science of Computer Programming*, 2014. **89**: p. 71-87.

139. van der Aalst, W.M.P., *Business Process Simulation Survival Guide*, in *Handbook on Business Process Management 1*, J. vom Brocke and M. Rosemann, Editors. 2015, Springer Berlin Heidelberg. p. 337-370.
140. Costa, P.D., et al. *A Model-Driven Approach to Situations: Situation Modeling and Rule-Based Situation Detection*. in *Enterprise Distributed Object Computing Conference (EDOC), 2012 IEEE 16th International*. 2012.
141. Kaddari, A., M.O.a.C. Malki, and S.B. Elmdeghri, *Toward a patient-centred and pattern-based workflow for interoperable hospital information systems*. *International Journal of Medical Engineering and Informatics*, 2017. **9**(3): p. 237-252.
142. Guizzardi, G., *Ontological foundations for structural conceptual models*. 2005: Enschede. p. 416.
143. Dorion, E. and S.p. Fortin. *Multi-source semantic integration - revisiting the theory of signs and ontology alignment principles*. in *2007 10th International Conference on Information Fusion*. 2007.
144. Guarino, N., D. Oberle, and S. Staab, *What Is an Ontology ?* 2009. p. 1-17.
145. Mylopoulos, J., et al., *Telos: representing knowledge about information systems*. *ACM Trans. Inf. Syst.*, 1990. **8**(4): p. 325-362.
146. Gonçalves, B., G. Guizzardi, and J.G. Pereira Filho, *Using an ECG reference ontology for semantic interoperability of ECG data*. *Journal of Biomedical Informatics*, 2011. **44**(1): p. 126-136.
147. Nardi, J.C., et al. *Towards a Commitment-Based Reference Ontology for Services*. in *2013 17th IEEE International Enterprise Distributed Object Computing Conference*. 2013.
148. Baumgartner, N. and W. Retschitzegger, *A survey of upper ontologies for situation awareness*, in *4th International Conference on Knowledge Sharing and Collaborative Engineering (IASTED)*. 2006.
149. Scherp, A., et al., *Designing core ontologies*. *Journal of Applied Ontology*, 2011. **6**: p. 177-221.
150. E.M., S., B. S., and M. C. *Events and activities: Is there an Ontology behind BPMN? in FOIS 2014 - 8th International Conference on Formal Ontology in Information Systems*, pp. 147-156, Rio de Janeiro, Brazil. 2014.
151. Rybola, Z. and R. Pergl. *Towards OntoUML for Software Engineering: Introduction to The Transformation of OntoUML into Relational Databases*. in *Enterprise and Organizational Modeling and Simulation*. 2016. Cham: Springer International Publishing.
152. Duarte, B.B., et al., *Ontological foundations for software requirements with a focus on requirements at runtime*. *Applied Ontology*, 2018. **13**: p. 73-105.
153. Guizzardi, G. and V. Zamborlini, *Using a trope-based foundational ontology for bridging different areas of concern in ontology-driven conceptual modeling*. *Science of Computer Programming*, 2014. **96**: p. 417-443.
154. Almeida, J.P.A., P.D. Costa, and G. Guizzardi. *Towards an Ontology of Scenes and Situations*. in *2018 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*. 2018.
155. Guizzardi, G., *Ontological Patterns, Anti-Patterns and Patterns Languages for Next-Generation Conceptual Modeling*, in *33rd International Conference on Conceptual Modeling (ER)*, E. Yu, et al., Editors. 2014, Springer International Publishing. p. 13-27.

156. Guizzardi, G., et al. *Towards Ontological Foundations for the Conceptual Modeling of Events*. in *Conceptual Modeling: 32th International Conference, ER 2013*. 2013. Springer Berlin Heidelberg.
157. Guarino, N. and G. Guizzardi, "We need to discuss the Relationship": Revisiting Relationships as Modeling Constructs, in *27th International Conference on Advanced Information Systems Engineering (CAiSE)*. 2015.
158. Sobral, V.M., J.P.A. Almeida, and P.D. Costa, *Assessing Situation Models with a Lightweight Formal Method*, in *5th International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*. 2015.
159. Ferreira, M.I., et al., *OntoEmergePlan: variability of emergency plans supported by a domain ontology*, in *12th International Conference on Information Systems for Crisis Response and Management (ISCRAM)*. 2015.
160. Fürst, F. and F. Trichet. *Heavyweight Ontology Engineering*. in *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*. 2006. Berlin, Heidelberg: Springer Berlin Heidelberg.
161. Euzenat, J., et al., *Ontology Alignment Evaluation Initiative: Six Years of Experience*, in *Journal on Data Semantics XV*, S. Spaccapietra, Editor. 2011, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 158-192.
162. Beek, W., S. Schlobach, and F. Harmelen, *A Contextualised Semantics for owl: sameAs*, in *Proceedings of the 13th International Conference on The Semantic Web. Latest Advances and New Domains - Volume 9678*. 2016, Springer-Verlag. p. 405-419.
163. Ganzha, M., et al. *Streaming semantic translations*. in *2017 21st International Conference on System Theory, Control and Computing (ICSTCC)*. 2017.
164. Otero-Cerdeira, L., F.J. Rodríguez-Martínez, and A. Gómez-Rodríguez, *Ontology matching: A literature review*. *Expert Systems with Applications*, 2015. **42**(2): p. 949-971.
165. Moreira, J., et al. *OntoWarehousing – Multidimensional Design Supported by a Foundational Ontology: A Temporal Perspective*. in *Data Warehousing and Knowledge Discovery*. 2014. Cham: Springer International Publishing.
166. Kokar, M.M., C.J. Matheus, and K. Baclawski, *Ontology-based situation awareness*. *Information Fusion*, 2009. **10**(1): p. 83-98.
167. Gu, T., H.K. Pung, and D.Q. Zhang, *A service-oriented middleware for building context-aware services*. *Journal of Network and Computer Applications*, 2005. **28**(1): p. 1-18.
168. Santos, L.O.B.d.S., et al., *A Service Architecture for Context Awareness and Reaction Provisioning*. 2007. p. 25-32.
169. Ntanos, C., et al., *A context awareness framework for cross-platform distributed applications*. *Journal of Systems and Software*, 2014. **88**: p. 138-146.
170. Arnaboldi, V., M. Conti, and F. Delmastro, *CAMEO: A novel context-aware middleware for opportunistic mobile social networks*. *Pervasive and Mobile Computing*, 2014. **11**: p. 148-167.
171. Ye, J., S. Dobson, and S. McKeever, *Situation identification techniques in pervasive computing: A review*. *Pervasive and Mobile Computing*, 2011. **8**: p. 36-66.
172. Costa, P.D., *Architectural support for context-aware applications: from context models to services platforms*. 2007: Enschede. p. 290.

173. Costa, P.D., et al. *Situations in Conceptual Modeling of Context*. in 2006 10th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'06). 2006.
174. Sobral, V.M., J.P.A. Almeida, and P.D. Costa. *Assessing situation models with a lightweight formal method*. in 2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision. 2015.
175. Costa, P.D., et al., *Rule-Based Support for Situation Management*, in *Fusion Methodologies in Crisis Management: Higher Level Fusion and Decision Making*, G. Rogova and P. Scott, Editors. 2016, Springer International Publishing: Cham. p. 341-364.
176. Aarab, Z., R. Saidi, and M.D. Rahmani. *Context Modeling and Metamodeling: A State of the Art*. in *Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015*. 2016. Cham: Springer International Publishing.
177. Guizzardi, G., et al., *Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story*. Journal of applied ontology, 2015. **10**: p. 259-271.
178. Itria, M.L., et al., *Identification of critical situations via Event Processing and Event Trust Analysis*. Knowledge and Information Systems, 2017. **52**(1): p. 147-178.
179. Boubeta-Puig, J., G. Ortiz, and I. Medina-Bulo, *Model4CEP*. Expert Syst. Appl., 2015. **42**(21): p. 8095-8110.
180. Hoehndorf, R., *Situoid theory – An ontological approach to situation theory*. 2005, University of Leipzig.
181. Costa, P.D., *Architectural Support for Context-Aware Applications: From Context Models to Services Platforms*. 2007, University of Twente.
182. Guizzardi, G., et al., *Towards ontological foundations for the conceptual modeling of events*, in *32nd International Conference on Conceptual Modeling (ER)*. 2013. p. 327-341.
183. Guizzardi, G., et al. *Towards Ontological Foundations for the Conceptual Modeling of Events*. in *International Conference Conceptual Modeling*. 2013. Springer Berlin Heidelberg.
184. Herre, H. and B. Heller, *Ontology of time and situoids in medical conceptual modeling*, in *Proceedings of the 10th conference on Artificial Intelligence in Medicine*. 2005, Springer-Verlag: Aberdeen, UK. p. 266-275.
185. Carvalho, V.A., et al., *Extending the Foundations of Ontology-Based Conceptual Modeling with a Multi-level Theory*, in *International Conference Conceptual Modeling*. 2015, Springer International Publishing: Cham. p. 119-133.
186. Costa, P.D., et al., *A Model-Driven Approach to Situations: Situation Modeling and Rule-Based Situation Detection*, in *16th International Enterprise Distributed Object Computing Conference (EDOC)*. 2012, IEEE. p. 154-163.
187. Mons, B., et al., *Cloudy, increasingly FAIR; Revisiting the FAIR Data guiding principles for the European Open Science Cloud*. Information Services and Use, 2017. **37**(1): p. 49-56.
188. Teixeira, S., et al., *Modeling and automatic code generation for wireless sensor network applications using model-driven or business process approaches*. J. Syst. Softw., 2017. **132**(C): p. 50-71.

189. Gao, F., *Chapter 4 - Frameworks for Big Data Integration, Warehousing, and Analytics*, in *Big Data Application in Power Systems*, R. Arghandeh and Y. Zhou, Editors. 2018, Elsevier. p. 57-73.
190. Pereira, I., P.D. Costa, and J.P. Almeida, *A Rule Based Platform for Situation Management*, in *3rd International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*. 2013, Ieee. p. 83-90.
191. Raymundo, C.R., et al. *An infrastructure for distributed rule-based situation management*. in *2014 IEEE International Inter-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*. 2014.
192. EC, *Rolling plan for ICT standardisation 2017*. 2017.
193. Ganzha, M., et al. *Semantic Technologies for the IoT - An Inter-IoT Perspective*. in *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*. 2016.
194. Ray, P.P., *A survey of IoT cloud platforms*. *Future Computing and Informatics Journal*, 2016. 1(1): p. 35-46.
195. Intel, *The Intel® IoT Platform Architecture Specification White Paper Internet of Things (IoT)*. 2016. p. 1-11.
196. Haghi, M., K. Thurow, and R. Stoll, *Wearable Devices in Medical Internet of Things: Scientific Research and Commercially Available Devices*. *Healthcare informatics research*, 2017. 23(1): p. 4-15.
197. Zamora-Izquierdo, M.A., et al., *Smart farming IoT platform based on edge and cloud computing*. *Biosystems Engineering*, 2019. 177: p. 4-17.
198. Tekinerdogan, B. and Ö. Köksal. *Pattern Based Integration of Internet of Things Systems*. in *Internet of Things – ICIOT 2018*. 2018. Cham: Springer International Publishing.
199. Li, H., et al., *Enabling Semantics in an M2M/IoT Service Delivery Platform*. *Proceedings - 2016 IEEE 10th International Conference on Semantic Computing (ICSC)*, 2016: p. 206-213.
200. Magnoni, L., *Modern Messaging for Distributed Sytems*. *Journal of Physics: Conference Series*, 2015. 608(1): p. 012038.
201. Götz, B., et al., *Challenges of Production Microservices*. *Procedia CIRP*, 2018. 67: p. 167-172.
202. Ciavotta, M., et al., *A Microservice-based Middleware for the Digital Factory*. *Procedia Manufacturing*, 2017. 11: p. 931-938.
203. Cherradi, G., et al., *Real-Time Microservices Based Environmental Sensors System for Hazmat Transportation Networks Monitoring*. *Transportation Research Procedia*, 2017. 27: p. 873-880.
204. Bittencourt, L., et al., *The Internet of Things, Fog and Cloud continuum: Integration and challenges*. *Internet of Things*, 2018. 3-4: p. 134-155.
205. Boubeta-Puig, J., G. Ortiz, and I. Medina-Bulo, *MEdit4CEP: A model-driven solution for real-time decision making in SOA 2.0*. *Knowledge-Based Systems*, 2015. 89: p. 97-112.
206. Díaz, M., C. Martín, and B. Rubio, *State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing*. *Journal of Network and Computer Applications*, 2016. 67: p. 99-117.

207. Khomh, F. and S.A. Abtahizadeh, *Understanding the impact of cloud patterns on performance and energy consumption*. Journal of Systems and Software, 2018. **141**: p. 151-170.
208. Afsari, K., C.M. Eastman, and D. Castro-Lacouture, *JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange*. Automation in Construction, 2017. **77**: p. 24-51.
209. Rizzardi, A., et al., *AUPS: An Open Source AUthenticated Publish/Subscribe system for the Internet of Things*. Information Systems, 2016. **62**: p. 29-41.
210. Katuwal, K., *Microservices: A Flexible Architecture for the Digital Age Version 1.0*. American Journal of Computer Science and Engineering, 2016. **3**(Microservices): p. 20-24.
211. Wang, D., et al., *A Novel Complex Event Processing Engine for Intelligent Data Analysis in Integrated Information Systems*. International Journal of Distributed Sensor Networks, 2016. **12**(3): p. 6741401.
212. Akzhalova, A., M. Alexeev, and B. Sarsembayev. *Model-Driven Architecture for the QoS-Based Adaptive Sensor Network System*. in *Business Modeling and Software Design*. 2015. Cham: Springer International Publishing.
213. Behmann, F. and K. Wu, *C-IoT: Application Requirements*, in *Collaborative Internet of Things (C-IoT)*. 2015, John Wiley & Sons Ltd. p. 47-129.
214. *Chapter 6 - From Sensor to User—Interoperability of Sensors and Data Systems*, in *Challenges and Innovations in Ocean In Situ Sensors*, E. Delory and J. Pearlman, Editors. 2019, Elsevier. p. 289-337.
215. Brandt, P., et al. *Semantic interoperability in sensor applications making sense of sensor data*. in *2013 IEEE Symposium on Computational Intelligence in Healthcare and e-health (CICARE)*. 2013.
216. Szilagyi, I. and P. Wira. *Ontologies and Semantic Web for the Internet of Things - a survey*. in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*. 2016.
217. Palavalli, A., D. Karri, and S. Pasupuleti. *Semantic Internet of Things*. in *IEEE Tenth International Conference on Semantic Computing (ICSC)*. 2016.
218. Gyrard, A., M. Serrano, and G.A. Atemezing. *Semantic web methodologies, best practices and ontology engineering applied to Internet of Things*. in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. 2015.
219. Gyrard, A., M. Serrano, and P. Patel, *Chapter 11 - Building Interoperable and Cross-Domain Semantic Web of Things Applications*, in *Managing the Web of Things*, Q.Z. Sheng, et al., Editors. 2017, Morgan Kaufmann: Boston. p. 305-324.
220. Khodadadi, F. and R.O. Sinnott, *A Semantic-aware Framework for Service Definition and Discovery in the Internet of Things Using CoAP*. Procedia Computer Science, 2017. **113**: p. 146-153.
221. Ganzha, M., et al. *From implicit semantics towards ontologies — practical considerations from the INTER-IoT perspective*. in *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. 2017.
222. Mahda, N., et al. *Concept Extraction from the Web of Things Knowledge Bases*. in *International Conference WWW/Internet 2018*. 2018. 21-23 October 2018 Budapest, Hungary: Elsevier.

300 REFERENCES

223. Soldatos, J., et al. *OpenIoT: Open Source Internet-of-Things in the Cloud*. in *Interoperability and Open-Source Solutions for the Internet of Things*. 2015. Cham: Springer International Publishing.
224. Battle, R. and D. Kolas, *Enabling the geospatial semantic web with parliament and GeoSPARQL*. *Semantic Web*, 2012. **3**(4): p. 355-370.
225. Kyzirakos, K., et al., *GeoTriples: Transforming geospatial data into RDF graphs using R2RML and RML mappings*. *Journal of Web Semantics*, 2018.
226. Hu, Y., *1.07 - Geospatial Semantics*, in *Comprehensive Geographic Information Systems*, B. Huang, Editor. 2018, Elsevier: Oxford. p. 80-94.
227. Daniele, L., F. den Hartog, and J. Roes. *Created in Close Interaction with the Industry: The Smart Appliances REFERENCE (SAREF) Ontology*. in *Formal Ontologies Meet Industry: 7th International Workshop (FOMI)*. 2015.
228. Daniele, L., F. den Hartog, and J. Roes. *How to Keep a Reference Ontology Relevant to the Industry: A Case Study from the Smart Home*. in *Ontology Engineering*. 2016. Cham: Springer International Publishing.
229. ETSI. *New version of the Machine 2 Machine Standard for Smart Appliances*. 2017.
230. Ganzha, M., et al., *Tools for Ontology Matching—Practical Considerations from INTER-IoT Perspective*, in *Internet and Distributed Computing Systems: 9th International Conference, IDCS 2016, Wuhan, China, September 28-30, 2016, Proceedings*, W. Li, et al., Editors. 2016, Springer International Publishing: Cham. p. 296-307.
231. Tommasini, R., et al. *Towards Ontology-Based Event Processing*. in *OWL: Experiences and Directions — Reasoner Evaluation*. 2017. Cham: Springer International Publishing.
232. Nicola, A.D., M. Melchiori, and M.L. Villani, *A Semantics-Based Approach to Generation of Emergency Management Scenario Models*, in *Enterprise Interoperability VI*. 2014. p. 163-173.
233. Riemer, D., L. Stojanovic, and N. Stojanovic. *SEPP: Semantics-Based Management of Fast Data Streams*. in *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*. 2014.
234. Akbar, A., et al., *Predictive Analytics for Complex IoT Data Streams*. *IEEE Internet of Things Journal*, 2017. **4**(5): p. 1571-1582.
235. Teymourian, K. and A. Paschke. *Semantic Rule-Based Complex Event Processing*. in *Rule Interchange and Applications*. 2009. Berlin, Heidelberg: Springer Berlin Heidelberg.
236. Digital Bazaar, I. *A JSON-LD Processor and API implementation in JavaScript*. 2011 31/01/2018]; Available from: <https://github.com/digitalbazaar/jsonld.js>.
237. Johnson, B., S. Ritchie, and D. Fonseca, *The State of the Art of Lethal Oviposition Trap-Based Mass Interventions for Arboviral Control*. *Insects*, 2017. **8**(1): p. 5.
238. Chen, Y., et al., *Flying Insect Classification with Inexpensive Sensors*. *Journal of Insect Behavior*, 2014. **27**(5): p. 657-677.
239. Silva, D.F., et al., *Exploring Low Cost Laser Sensors to Identify Flying Insect Species*. *Journal of Intelligent & Robotic Systems*, 2015. **80**(1): p. 313-330.
240. WHO, *Early detection, assessment and response to acute public health events: Implementation of Early Warning and Response with a focus on Event-Based Surveillance*. 2014.

241. Božić, B., et al., *Requirements Engineering for Semantic Sensors in Crisis and Disaster Management*. in *Environmental Software Systems. Infrastructures, Services and Applications*. 2015. Cham: Springer International Publishing.
242. Casado, R., et al., *Data interoperability software solution for emergency reaction in the Europe Union*. *Nat. Hazards Earth Syst. Sci.*, 2015. **15**(7): p. 1563-1576.
243. Boissel-Dallier, N., et al., *Mediation information system engineering based on hybrid service composition mechanism*. *Journal of Systems and Software*, 2015. **108**: p. 39-59.
244. Wu, D. and Y. Cui, *Disaster early warning and damage assessment analysis using social media data and geo-location information*. *Decision Support Systems*, 2018. **111**: p. 48-59.
245. alhmoudi, a.a. and Z. Aziz, *Integrated framework for early warning system in UAE*. *International Journal of Disaster Resilience in the Built Environment*, 2016. **7**(4): p. null.
246. Rocklöv, J., et al., *Assessing Seasonal Risks for the Introduction and Mosquito-borne Spread of Zika Virus in Europe*. *EBioMedicine*, 2016. **9**: p. 250-256.
247. Lai, P.-C., et al., *An early warning system for detecting H1N1 disease outbreak – a spatio-temporal approach*. *International Journal of Geographical Information Science*, 2015. **29**(7): p. 1251-1268.
248. Liu, S., K. Smith, and H. Che, *A multivariate based event detection method and performance comparison with two baseline methods*. *Water Research*, 2015. **80**: p. 109-118.
249. Marsh, K., et al., *Multiple Criteria Decision Analysis for Health Care Decision Making—Emerging Good Practices*. *Value in Health*, 2016. **19**(2): p. 125-137.
250. Al-Khudhairi, D., et al., *Towards integrative risk management and more resilient societies*. *The European Physical Journal Special Topics*, 2012. **214**(1): p. 571-595.
251. Martínez-García, A., et al., *Working with the HL7 metamodel in a Model Driven Engineering context*. *Journal of Biomedical Informatics*, 2015. **57**: p. 415-424.
252. Othman, S.H. and G. Beydoun, *A metamodel-based knowledge sharing system for disaster management*. *Expert Systems with Applications*, 2016. **63**: p. 49-65.
253. Lendholt, M. and M. Hammitzsch, *Towards an Integrated Information Logistics for Multi Hazard Early Warning Systems*. *The Open Environmental Engineering Journal* 2012: p. 27-43.
254. Keeney, J.H., S. Buan, and L. Diamond, *Multi-Hazard Early Warning System of the United States National Weather Service*, in *Institutional Partnerships in Multi-Hazard Early Warning Systems: A Compilation of Seven National Good Practices and Guiding Principles*, M. Golnaraghi, Editor. 2012, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 115-157.
255. Atluri, V., et al., *UICDS-based information sharing among emergency response application systems*, in *Proceedings of the 12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times*. 2011, ACM: College Park, Maryland, USA. p. 331-332.
256. Barros, R., et al., *EDXL-RESCUER ontology : Conceptual Model for Semantic Integration*. *Proceedings of the ISCRAM 2015 Conference - Kristiansand*, 2015.
257. Chan, S., et al., *Public Safety and Emergency Management Communications Ontology*. 2017.

302 REFERENCES

258. Petersen, K., *Secure Dynamic Cloud for Information, Communication and Resource Interoperability Based on Pan-European Disaster Inventory*, in *SecInCore project - Deliverable 2.2 - ELSI Guidelines for collaborative design and database of representative emergency and disaster events in Europe*. 2015.
259. Rahmani, A.M., et al., *Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach*. *Future Generation Computer Systems*, 2018. **78**: p. 641-658.
260. GE, *GE Healthcare to Connect 500,000 Imaging Machines with Announcement of GE Health Cloud and Apps*. 2015.
261. Mescherin, S., I.A. Kirillov, and S.V. Klimenko, *Ontology of Emergency Shared Situation Awareness and Crisis Interoperability*. 2013 International Conference on Cyberworlds, 2013: p. 159-162.
262. Liu, S., C. Brewster, and D. Shaw, *Ontologies for crisis management: a review of state of the art in ontology design and usability*, in *10th International Conference on Information Systems for Crisis Response and Management (ISCRAM 2013)*. 2013: Baden-Baden, Germany. p. 349-359.
263. Jin, D. and J. Lin, *Managing tsunamis through early warning systems: A multidisciplinary approach*. *Ocean & Coastal Management*, 2011. **54**(2): p. 189-199.
264. Sherif, H.M., M.A. Shedid, and S.A. Senbel. *Real time traffic accident detection system using wireless sensor network*. in *2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*. 2014.
265. Miller, T., *Explanation in artificial intelligence: Insights from the social sciences*. *Artificial Intelligence*, 2019. **267**: p. 1-38.
266. Sanfilippo, E.M., S. Borgo, and C. Masolo, *Events and Activities: Is there an Ontology behind BPMN?*, in *8th International Conference on Formal Ontology in Information Systems (FOIS)*. 2014.
267. Abbasi, H.M., et al., *A survey of ontology learning techniques and applications*. *Database*, 2018. **2018**.
268. Guizzardi, G. and G. Wagner, *What's in a Relationship: An Ontological Analysis*, in *27th International Conference on Conceptual Modeling (ER)*. 2008.
269. Bringuente, A.C.O., R.A. Falbo, and G. Guizzardi, *Using a Foundational Ontology for Reengineering a Software Process Ontology*. *Journal of Information and Data Management*, 2011.
270. Ansaldi, S., et al., *An Ontology for the Identification of the most Appropriate Risk Management Methodology*, in *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*, P. Herrero, et al., Editors. 2012, Springer Berlin Heidelberg. p. 444-453.
271. Boulton, E., et al., *Developing the FARSEEING Taxonomy of Technologies: Classification and description of technology use (including ICT) in falls prevention studies*. *Journal of Biomedical Informatics*, 2016. **61**: p. 132-140.
272. Baldini, G., et al., *Survey of Wireless Communication Technologies for Public Safety*. *IEEE Communications Surveys & Tutorials*, 2014. **16**(2): p. 619-641.
273. Karma, S., et al., *Use of unmanned vehicles in search and rescue operations in forest fires: Advantages and limitations observed in a field trial*. *International Journal of Disaster Risk Reduction*, 2015. **13**: p. 307-312.

274. Othman, S.H., G. Beydoun, and V. Sugumaran, *Development and validation of a Disaster Management Metamodel (DMM)*. Information Processing & Management, 2014. **50**(2): p. 235-271.
275. Gilbert, C., *Studying disaster: A review of main conceptual tools*. International Journal of Mass Emergencies and Disasters, 1995. **13**: p. 231-240.
276. Costa, P.D., et al. *A Model-Driven Approach to Situations: Situation Modeling and Rule-Based Situation Detection*. in *Enterprise Distributed Object Computing Conference*. 2012.
277. Moody, D., *The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering*. IEEE Transactions on Software Engineering, 2009. **35**(6): p. 756-779.
278. Matheus, C.J., M.M. Kokar, and K. Baclawski, *A Core Ontology for Situation Awareness*, in *6th International Conference on Information Fusion*. 2003.
279. Falbo, R.D.A., et al., *Organizing Ontology Design Patterns as Ontology Pattern Languages*, in *10th International Conference on the Semantic Web: Semantics and Big Data International Conference*. 2013.
280. Apisakmontri, P., et al., *Ontology Construction and Schema Merging Using an Application-Independent Ontology for Humanitarian Aid in Disaster Management*, in *4th International Conference on Semantic Technology*. 2014. p. 187-194.
281. Ganzha, M., et al., *Semantic interoperability in the Internet of Things: An overview from the INTER-IoT perspective*. Journal of Network and Computer Applications, 2016.
282. INTER-IoT, *INTER-IoT deliverable: D2.3. Requirements and business analysis*. 2016.
283. MIMAR, T., *Vehicle security with accident notification and embedded driver analytics*. 2016, Google Patents.
284. Mane, A. and J. Rana. *Vehicle Collision detection and Remote Alarm Device using Arduino*. 2014.
285. San Vicente Foronda, L., *Automotive Events Detection using MEMS Accelerometers*. 2009: p. 129-129.
286. Sulochana, B. and B.A.S.M. Babu, *Monitoring and Detecting Vehicle Based On Accelerometer and MEMS Using GSM and GPS Technologies*. International Journal of Computer Science Trends and Technology (IJCTST), 2014. **2**(4): p. 55-59.
287. Ibrahim, H.A., A.K. Aly, and B.H. Far. *A system for vehicle collision and rollover detection*. in *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. 2016.
288. Thompson, C., et al., *Using Smartphones to Detect Car Accidents and Provide Situational Awareness to Emergency Responders*, in *Mobile Wireless Middleware, Operating Systems, and Applications: Third International Conference, Mobilware 2010, Chicago, IL, USA, June 30 - July 2, 2010. Revised Selected Papers*, Y. Cai, et al., Editors. 2010, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 29-42.
289. Fortino, G. and R. Gravina, *Fall-MobileGuard: a smart real-time fall detection system*, in *Proceedings of the 10th EAI International Conference on Body Area Networks*. 2015, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering): Sydney, New South Wales, Australia. p. 44-50.
290. Nandaniya, K., V. Choksi, and M.B. Potdar, *Microcontroller Based Collision Detection and Warning System*. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2014. **3**(5): p. 9565-9570.

304 REFERENCES

291. Abbott, T.E.F., et al., *Pre-hospital National Early Warning Score (NEWS) is associated with in-hospital mortality and critical care unit admission: A cohort study*. *Annals of Medicine and Surgery*, 2018. **27**: p. 17-21.
292. Xia, H., I. Asif, and X. Zhao, *Cloud-ECG for real time ECG monitoring and analysis*. *Computer Methods and Programs in Biomedicine*, 2013. **110**(3): p. 253-259.
293. Hossain, M.S. and G. Muhammad, *Cloud-assisted Industrial Internet of Things (IIoT) – Enabled framework for health monitoring*. *Computer Networks*, 2016. **101**: p. 192-202.
294. Steinberg, J.S., et al., *2017 ISHNE-HRS expert consensus statement on ambulatory ECG and external cardiac monitoring/telemetry*. *Heart Rhythm*, 2017. **14**(7): p. e55-e96.
295. Shimmer, *Shimmer User Manual Revision 3p*. 2017, Realtime Technologies Ltd.
296. Fortino, G., et al., *BodyCloud: A SaaS approach for community Body Sensor Networks*. *Future Generation Computer Systems*, 2014. **35**: p. 62-79.
297. Mačković, D. and N. Aničić, *The Systems Development Life Cycle to Facilitate Progression Towards Semantic and Organizational Interoperability for Healthcare System*, in *Enterprise Interoperability VII: I-ESA proceedings*, K. Mertins, et al., Editors. 2016, Springer International Publishing: Cham. p. 101-111.
298. HL7, *HL7 Version 3 Implementation Guide: Annotated ECG (aECG) R1, Release 2*. 2005.
299. Bond, R.R., et al., *A review of ECG storage formats*. *International Journal of Medical Informatics*, 2011. **80**(10): p. 681-697.
300. Rubel, P., et al. *SCP-ECG V3.0: An enhanced standard communication protocol for computer-assisted electrocardiography*. in *2016 Computing in Cardiology Conference (CinC)*. 2016.
301. Lee, S. and H. Do, *Comparison and Analysis of ISO/IEEE 11073, IHE PCD-01, and HL7 FHIR Messages for Personal Health Devices*. *Health Inform Res*, 2018. **24**(1): p. 46-52.
302. Gonçalves, B., V. Zamborlini, and G. Guizzardi, *An Ontological Analysis of the Electrocardiogram*. *Electronic Journal of Communication, Information and Innovation in Health*, 2009: p. 1-26.
303. Bo, B., et al., *Ontology Mapping and Reasoning in Semantic Time Series Processing*, in *Proceedings of International Conference on Information Integration and Web-based Applications & Services*. 2013, ACM: Vienna, Austria. p. 443-452.
304. Kauppinen, T. and E. Hyvönen, *Modeling and Reasoning About Changes in Ontology Time Series*, in *Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems*, R. Sharman, R. Kishore, and R. Ramesh, Editors. 2007, Springer US: Boston, MA. p. 319-338.
305. Bera, P.a.P., Geert, *The effects of construct redundancy on readers' understanding of conceptual models*. *JOURNAL OF DATABASE MANAGEMENT*, 2017.
306. Belesioti, M., et al. *e-Health Services in the Context of IoT: The Case of the VICINITY Project*. in *Artificial Intelligence Applications and Innovations*. 2018. Cham: Springer International Publishing.
307. Kalamaras, I., et al. *Towards Big Data Analytics in Large-Scale Federations of Semantically Heterogeneous IoT Platforms*. in *Artificial Intelligence Applications and Innovations*. 2018. Cham: Springer International Publishing.
308. Douzis, K., et al., *Modular and generic IoT management on the cloud*. *Future Generation Computer Systems*, 2018. **78**: p. 369-378.

309. Compton, M., et al., *Reasoning about sensors and compositions*, in *Proceedings of the 2nd International Conference on Semantic Sensor Networks - Volume 522*. 2009, CEUR-WS.org: Washington DC. p. 33-48.
310. Tsadiras, A. and N. Bassiliades, *RuleML representation and simulation of Fuzzy Cognitive Maps*. Expert Systems with Applications, 2013. **40**(5): p. 1413-1426.
311. EUDAT. *EUDAT B2share REST API*. 2018 05-05-2018]; Available from: <https://B2share.eudat.eu/help/api>.
312. EUDAT. *An EUDAT-based FAIR Data Approach for Data Interoperability*. 2018 05-05-2018]; Available from: <https://eudat.eu/communities/an-eudat-based-fair-data-approach-for-data-interoperability>.
313. re3data.org. *B2share data repository entry in re3data.org*. 2018 05-05-2018]; Available from: <https://www.re3data.org/repository/r3d100011394>.
314. Fowler, M., *Patterns of Enterprise Application Architecture*. 2002.
315. Moreira, J., *Enabling B2SHARE to behave as a FAIR Data Point: a proof-of-concept* <https://github.com/jonimoreira/B2SHARE-FAIR>. 2018.
316. Digital Bazaar, I. *Python library for JSON-LD*. 2017 05-05-2018]; Available from: <https://github.com/digitalbazaar/pyld>.

About the author

João Moreira was born in 1980 in Rio de Janeiro (Brazil), he has a bachelor in computer science and a master in informatics from the Universidade Federal do Rio de Janeiro (UFRJ). He has a broad experience in the ICT industry since 2001, and in academy he is a researcher and professor since 2008. João worked in several companies during his career, always bridging academy and industry through applied research. João is highly communicative and usually exercises leadership.

Joni (as family and friends fondly call him) is a sports lover, has run three marathons and more than dozen half marathons. He loves to swim and find his passion in the sea while surfing. Cannot fail to mention that Joni is a loving father and caring husband. Not leaving behind any minute of joy on seeing his son Teo grow up and his wife Isabel cheer life. His breath is for sure endless.