Ontology-Based Context-Aware Model for Event Processing in an IoT Environment

Amer Al-Lahham

School of Computing, Science and Engineering

College of Science and Technology

University of Salford, Salford, UK

January 2020

ABSTRACT

The Internet of Things (IoT) is more and more becoming one of the fundamental sources of data. The observations produced by these sources are made accessible with heterogeneous vocabularies, models and data formats. The heterogeneity factor in such an enormous environment complicates the task of sharing and reusing this data in a more intelligent way (other than the purposes it was initially set up for). In this research, we investigate these challenges, considering how we can transform raw sensor data into a more meaningful information. This raw data will be modelled using ontology-based information that is accessible through continuous queries for sensor streaming data.

Interoperability among heterogeneous entities is an important issue in an IoT environment. Semantic modelling is a key element to support interoperability. Most of the current ontologies for IoT mainly focus on resources and services information. This research builds upon the current state-of-the-art ontologies to provide contextual information and facilitate sensor data querying. In this research, we present an Ontology to represent an IoT environment, with emphasis on temporal and geospatial context enrichment. Furthermore, the Ontology is used alongside a proposed syntax based on Description Logic to build an Event Processing Model. The aim of this model is to interconnect ontology-based reasoning with event processing. This model enables to perform event processing over high-level ontological concepts.

The Ontology was developed using the NeOn methodology, which emphasises on the reuse and modularisation. The Competency Questions techniques was used to develop the requirements of this Ontology. This was later evaluated by domain experts in software engineering and cloud computing. The ontology was evaluated based on its completeness, conciseness, consistency and expandability, over 70% of the domain experts agreed on the core modules, concepts and relationships within the ontology. The resulted Ontology provides a core IoT ontology that could be used for further development within a specific IoT domain.

The proposed Ontology-Based Context-Aware model for Event-Processing in an IoT environment “OCEM-IoT”, implements all the time operators used in complex event processing engines. Throughput and latency were used as performance comparison metrics for the syntax evaluation; the results obtained show an improved performance over existing event processing languages.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

TABLE OF LISTINGS

LIST OF ABBREVIATION

CEP	Complex Event Processing
CQ	Competency Question
C-SPARQL	Continuous SPARQL
DSMS	Data Stream Management System
EP	Event Processing
EPA	Event Processing Agent
EPN	Event Processing Network
EPS-DL	Event Processing Syntax based on Description Logic
EPTS	Event Processing Technical Society
FIPA	Foundation for Intelligent Physical Agents
ICM	Integrated Conceptual Model
IoT	Internet of Things
JAD	Joint Application Development
LSM	Linked Sensor middleware
NED	Networked Embedded Devices
ODP	Ontology Design Patterns
RDF	Resource Description Framework
RFID	Radio Frequency Identification
RSP	RDF Stream Processing
RWI	The Real World Internet
SAREF	Smart Appliance REFerence
SCO	Sensor Cloud Ontology
SR	Stream Reasoning
SSN	Semantic Sensor Network Ontology
URI	Uniform Resource Identifier
WSAN	Wireless Sensor and Actuator Networks

DECLARATION

I declare that the contents of this thesis are of original quality, apart from any particular references that are made regarding other researchers. This work has not been submitted previously for consideration to my current university or at any other institute. The entire content of this research is my personal work.

ACKNOWLEDGMENTS

First, I would like to thank my supervisors who have provided all the support towards the completion of this thesis. My gratitude goes to my supervisor Prof. Farid Meziane for his dedication, support, patience and constructive feedback. I would also like to thank my co-supervisor Dr. Omar Alani for his invaluable support, patience and encouragement.

I also would like to thank Dr Adil Al-Yasiri, who has provided me with perpetual support, expert advice, and for showing unlimited patience and trust in myself. I would like to thank the University of Salford for providing all the support needed during my studies.

Many thanks to my parents, my wife and sons for their constant support and encouragement during this journey.

CHAPTER 1: INTRODUCTION

1.1 Introduction

The Internet of Things (IoT) is characterised by an ever-increasing number of Things embedding sensing, actuating, processing and communication capabilities. The IoT paradigm represents numerous things connected to the Internet (Atzori et al., 2010). The interactions with these connected things could help reach specific goals in various domains. The data generated by these things originate from numerous heterogeneous sources, each sensing a part of the environment. Combining the transmitted data from multiple sources facilitates applications to support context awareness (Su et al., 2015, Barnaghi et al., 2012). This empowers applications to analyse and understand the given situation. For example, a fire detection system could combine the historical room temperature, smoke level, light intensity with various sensor readings such as temperature sensor, smoke detector, and light intensity sensor in order to detect fire.

The IoT aims at building intelligent systems that can provide support to people during their daily activities. In order to achieve this awareness, understanding the raw sensor data is necessary. Collection, modelling, reasoning, and distribution of context in relation to sensor data plays a critical role in order to tackle this challenge (Perera et al., 2014).

Context-aware computing has proven to be successful in understanding sensor data. Context-aware systems can acquire, interpret and use context information to adapt their behaviour to the current context (Byun and Cheverst, 2004). Context-aware systems have played an essential role in tackling this challenge in previous paradigms. Their proven past success makes them a strong candidate solution that is ought to be successful in the IoT paradigm as well (Perera et al., 2014).

According to Perera et al. (2014), one of the important design principles for context-aware systems is scalability and extensibility. Nodrum in Gartner (2016) expects 20 billion connected things to be in use worldwide by 2020 (Nordrum, 2016). Thus, it should be straightforward to add new sensors and devices to a context-aware system. Additional sensors and devices produce new data that might need to be processed differently. Consequently, it should be possible to easily add extra processing services to extract high-level knowledge. Following this thought, the number of services and applications handling the produced IoT data will increase rapidly. Consequently, context-aware platforms for the IoT should be easily extensible (Bonte et al., 2017, Gray et al., 2011b).

According to Strang and Linnhoff-Popien (2004), semantics are the preferred mechanism of managing and modelling context Semantics can aid in the integration of the generated heterogeneous IoT data by enabling interoperability between different sources and providing a uniform model (Strang and Linnhoff-Popien, 2004) (Bergamaschi et al., 2001, Nagib and Hamza, 2016).

Semantic reasoning allows to compute logical consequences defined in the semantic model. For example, the model could define an alarming fire detection as a sensor reading from a smoke detector sensor with an accuracy above a certain threshold (e.g., 80%) resulting from a sensor in a room in a certain building. When such a sensor reading is detected by the reasoner, it will know it has detected fire, and someone should be called to attend to the situation, even when it is not explicitly stated in the sensor data. Utilising semantic reasoning enables transforming the integrated low-level data into high-level knowledge, allowing accurate and intelligent decisions (Bonte et al., 2017).

The research community faces the following challenges: (a) A heterogenous environment, at all levels; data structures, devices, data schemas, and data formats provided by the sensors, this aspect makes it difficult to make use of the data provided and integrate it with various data

sources. (b) The high dynamicity of data streams, presents significant challenges for data management and query processing. The challenges of data volume and velocity are intrinsic in such scenarios, besides the data variety at such environments. These three challenges have been studied since the emergence of the Big Data concept and are referred to as the 'Three V's'. (Laney, 2001).

This thesis addresses some of these challenges, by considering the transformation of raw data into more meaningful concepts based on ontologies, thus making it accessible by implementing continuous queries for the streaming data. The usage of ontologies as a method to provide explicit semantics for data has been adopted by many researchers, in order to faciliate reuse, integration and reasoning over it. Furthermore, recent research in this field has developed continuous querying and event processing, besides the traditional database systems, which depended on relational and transactional principles. These two querying systems are important in order to develop methods and techniques that can coherntly combine them.

This thesis studies methods to access and query streaming data based on high-level ontological concepts. Stored and static data are available in various forms, for example XML files, databases, webpages, etc., streams of data nowadays are accessible on the Web, using various schemas and formats. Moreover, the technologies employed make it extremely difficult to reuse, integrate and reason about these data streams within an IoT environment. This thesis investigates some essential extensions for ontology-based querying, it also investigates semantic technologies in order to support streaming data, and methods to link or map this type of data to higher-level ontological concepts that represents a particular domain. Sensor meta-data can be represented and enriched using semantics. Querying data streams make it necessary to provide models that reflect higher-level domain concepts. Bearing in mind the high heterogeneity at IoT, and the inconsistency and misleading meta-data. This thesis studies the

classification and characterisation of sensor data, which enables providing a richer data descriptions based on ontological concepts.

1.2 Research Problem

Applications based within an IoT environment attempt to solve the challenge of heterogeneity and interoperability (Bonino et al., 2015, Jin et al., 2014). There has been made significant advances in this field, however, much work and investigation are still needed in order to achieve the level of integration required to truly unlock the value of advanced applications in these domains (Manyika, 2015, Sheth, 2016). Specifically, while the Internet of Things is achieving communication layer interoperability, this does not consider the use of this data at the application layer (Wang et al., 2013),(Su et al., 2015). This has recently been observed in research and has led to a new field of studying application layer interoperability, which has been termed the Semantic Web of Things, owing to its grounding in semantic technologies (Zhang et al., 2016), (Pfisterer et al., 2011).

Now, a significant effort is required to accelerate progress in this field by developing and applying Semantic Web of Things processes and artefacts (Gyrard et al., 2014a). This includes the development, adoption, and standardisation of relevant ontologies, but also knowledge surrounding their lifecycle processes and accompanying software (Gyrard et al., 2015).

Regardless of application layer interoperability, how can organisations derive value from the 'rising tide' of big data? Ongoing big data and ICT intervention research should go side by side with the rapidly growing Semantic Web of Things, in order to best capitalise on artificial intelligence (Vázquez Salceda et al., 2014), optimisation (Kuznetsova et al., 2014), simulation (Nguyen et al., 2014), and advanced applications in general. It is critical that IoT systems integrate data effectively and efficiently, and user interfaces apply business semantics intelligently, in order to best empower decision makers and organisations (Aufaure et al.,

2016). This leads to the challenge of capturing complex semantics, then using them to better support advanced applications, built on the technology of the Internet of Things (Chianese et al., 2017, Baars and Ereth, 2016). Again, this requires significant research around the Semantic Web of Things and emphasises the need for a pragmatic mindset of using semantic technologies to provide business value in real-world contexts.

Two main research issues were identified by analysing the state of the art, the two research issues are concerned with sensor data streams. The first one focuses on the data enrichment, raw data observations could be used to extract and mine information that is more useful, the second one explores how semantic technologies can be used to query these observations.

First Issue: Enriching streaming data by using context information and providing events capabilities. Inferring events and context is highly essential, especially in the case of data loss which is highly probable in an IoT environment.

With regard to this issue, concerning the characterisation of semantic properties of sensor data, the following research questions are addressed:

How can ontologies be used to represent the semantic properties of sensor data?

How can sensor data be enriched using context?

By using the temporal and geospatial module, the data stream can be enriched with semantic properties such as time and location.

Second Issue: Accessing streaming data by using description logic based queries and the usage of ontological models to represent the sensors' observations. With an emphasis on time operators to characterise sensor data and facilitate event processing on a higher level.

Regarding this issue, this thesis aims to address the following research questions:

How ontologies can be used to query sensor data streams?

Is it possible to develop SPARQL event processing extensions suitable for querying sensor data?

It is possible to integrate Complex Event Processors and sensor data through the use of ontologies as a data model?

1.3 Research Aim and Objectives

The aim of this thesis is to develop an Ontology-Based Context-Aware model for Event-Processing in an IoT environment for accessing, enriching and querying data streams from heterogeneous sources, by adopting the use of ontologies to represent the data captured by sensors. This thesis originates from the research on ontology-based event query techniques, the extensions for sensor data streams operators and the use of descriptive logic to capture data and extract relevant properties sources. It aims to make sense of the raw sensor data in real time of multiple, heterogeneous, gigantic and inevitably noisy sensor data streams. Specific objectives are:

1. Comparison of existing IoT ontologies
2. Review of current methodologies for ontology development in software engineering.
3. To develop an architectural model. This model interconnects ontology-based reasoning with event processing.
4. To develop an ontology-based access and enrichment. An Ontology for IoT, which could deal with all Things in an IoT environment, tags, sensors, actuators, and software and add context to observations.
5. To develop a time-aware event processing syntax based on description logic to facilitate event processing.
6. Evaluating the proposed ontology using a survey questionnaire.

7. Evaluating the proposed event processing syntax by comparing it to existing models

1.4 Research Contributions

In this thesis, we present the OCEM-IoT mode "an Ontology-Based Context-Aware model for Event-Processing in an IoT environment". It is developed to provide semantic enrichment and reasoning on IoT data. OCEM-IoT uses ontologies to represent sensor data and context information as events and then apply different kinds of reasoning to derive high-level knowledge.

The main contribution of this research is the development of a generic IoT ontology based on the reuse of existing ontologies and modules. Ontology development in software engineering does not provide standards for devising ontologies, this research contributes by studying the existing methodologies and guidelines and outlines an approach to reuse the core modules and concepts for a generic application in an IoT environment. The approach presented in this thesis is superior to existing IoT ontologies, mainly because it covers the core modules and concepts and can be used within any field in IoT, since it is built on a modular basis and this can easily be modified and built upon.

Another contribution of this research is providing a novel method to access and query sensor data from a wide range of heterogeneous sources. This work is based on the investigation of ontology based querying and providing extensions for streaming operators. The thesis contributes within this field by the development of an event processing syntax based on description logic, which combines both complex event processing and description logic in order to provide stream reasoning. The novelty of this syntax is combing the semantic web and complex event processing, this combination of the two fields, facilitates the stream reasoning, firstly by defining events over high level concepts (provided by the ontology) and secondly by supporting the time operators used by existing complex event processors.

1.5 Hypothesis

Hypothesis 1: Sensor data streams could be defined as instances of comprehensive ontological model concepts.

Hypothesis 2: The SPARQL language queries could be extended with various time operators in order to query sensor data streams in real time, with reference to the ontological concepts and properties.

Hypothesis 3: Ontology-based queries for sensor data can be denoted as concrete expressions based on description logic and complex event processing, and the results of a query would be described as instances of the ontological concepts.

1.6 Research Methodology

This research aims to investigate the research issues identified in section 1.2, and to propose an Event Processing Architecture Model based on ontologies for IoT, in order to tackle the identified challenges. Scientific experimental research strategy has been adopted for our research. This study uses qualitative and quantitative methods, as the research problem formalisation uses historical data (documents analysis), while the final results were produced quantitatively using experiments and quantitative user study. The research strategy followed is scientific experimental. According to (Novikov and Novikov, 2013), scientific research includes three main phases:

Design of Scientific Research: This phase comprises determining the problem domain, formalising the research problem and identifying research aim and objectives.

Technology of Scientific Research: This phase includes preparing the required theory, developing the proposed system and finally applying the evaluation methods.

Reflexion in Scientific Research: This phase is related to finalising the results and preparing the final guidelines, then presenting them to the research community. Each phase has been divided into a number of stages; a brief description of each stage can be found below, and Figure 1-1 shows the research stages as a flowchart.

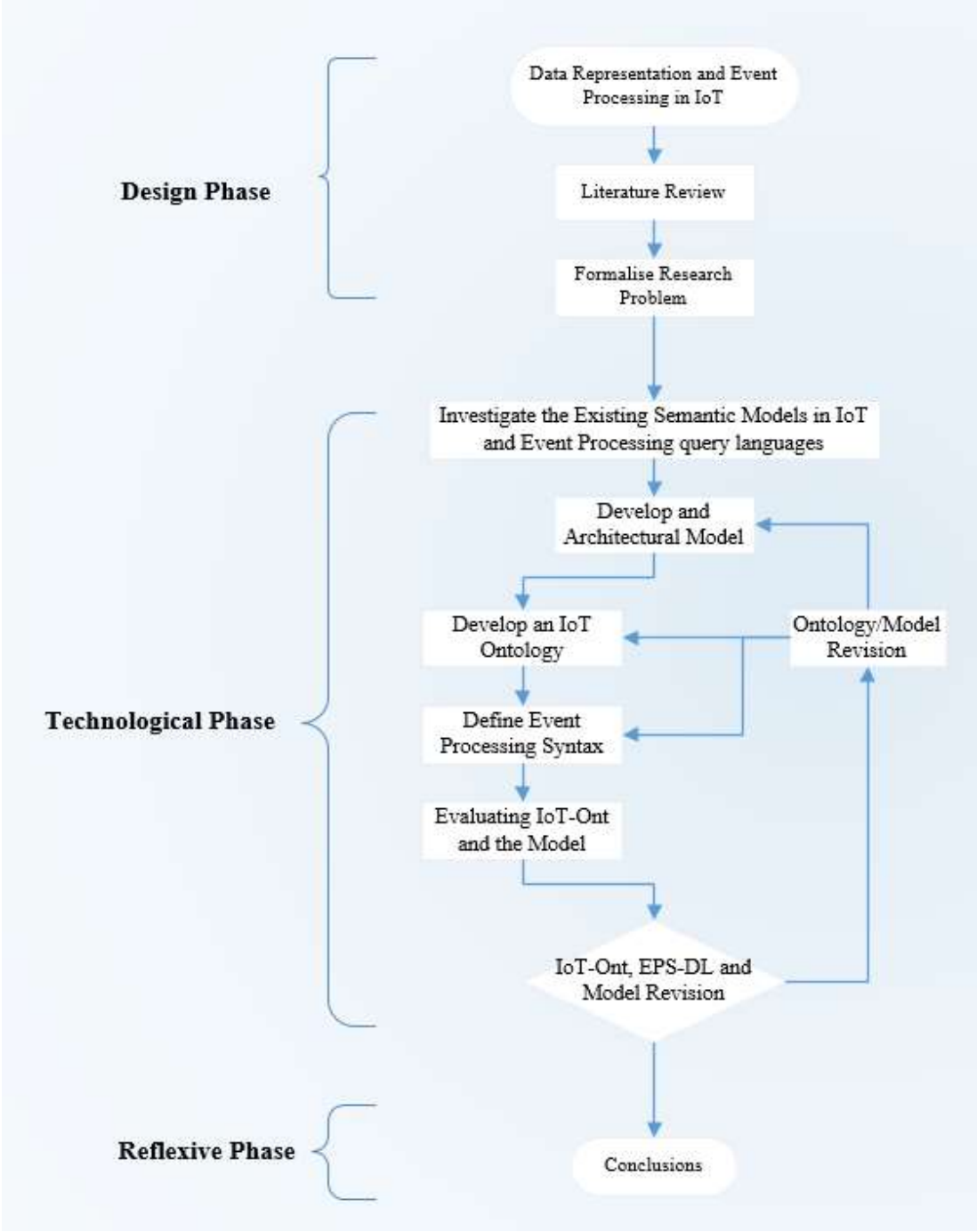

Figure 1-1 Research Main Stages

1) Design of Scientific Research Phase

A. Data Representation and Event Processing in the Internet of Things (IoT): This was the first stage in our research: identifying semantic models and event processing in IoT as the primary research domain.

B. Reviewing Literature: Investigate earlier researchers' work about Semantic Models and Event Processing in IoT in order to get a good understanding and to know the state of the art of IoT; furthermore, to highlight the drawbacks of the current approaches.

C. Formalising the Research Problem: This stage identified the needs of IoT for new techniques to represent and access data in an IoT Environment, recognising the urgent need to define a reference semantic model to enable heterogeneous devices to interact (sharing information and processing), moreover to facilitate a higher event processing based on the defined semantics. In accordance with these findings, the research aim has been addressed as establishing a new semantic model and a new event processing syntax based on DL as an extension to SPARQL for the IoT domain.

2) Technology of Scientific Research Phase

A. Investigate the Existing Semantic Models in IoT and Event Processing query languages: This stage was achieved by investigating and analysing previous works in this field, and highlighting their limitations.

B. Develop and Architectural Model: This stage was about identifying an architectural model that interconnects an IoT semantic model with complex event processing.

C. Develop an Ontology that represents the primary modules and concepts in an IoT domain, and their logical relationships. By analysis the current state-of-the-art IoT ontologies, as well as the methodologies used for ontology development, methods, and techniques proposed by the research community to define requirements and best design practices.

D. Define Event Processing Syntax based on Description Logic (EPS-DL): This stage was defining the required parameters for an event processing language that can deal with temporal aspects. EPS-DL syntax and semantics.

E. Evaluating the Proposed Ontology: The assessment and verification of the IoT-Ont was done using a survey questionnaire targeting domain experts in IoT and Software Engineering.

F. Evaluating OCEM-IoT Model: The assessment and verification of the whole model are crucial to this research, where its usability and functional suitability should be ensured. Many use-cases were applied and compared to other event processing models in IoT. The tests were carried out after defining an evaluation plan.

G. IoT-Ont and OCEM-IoT Revision: According to the findings and outputs of the evaluation plan, amendments and modifications were applied to OCEM-IoT. This stage helps to enhance the results that are produced by this research.

3) Reflexion in Scientific Research Phase

A. The final conclusions and recommendations: The last phase of this research is to reflect on the final findings and recommendations.

1.7 Approach

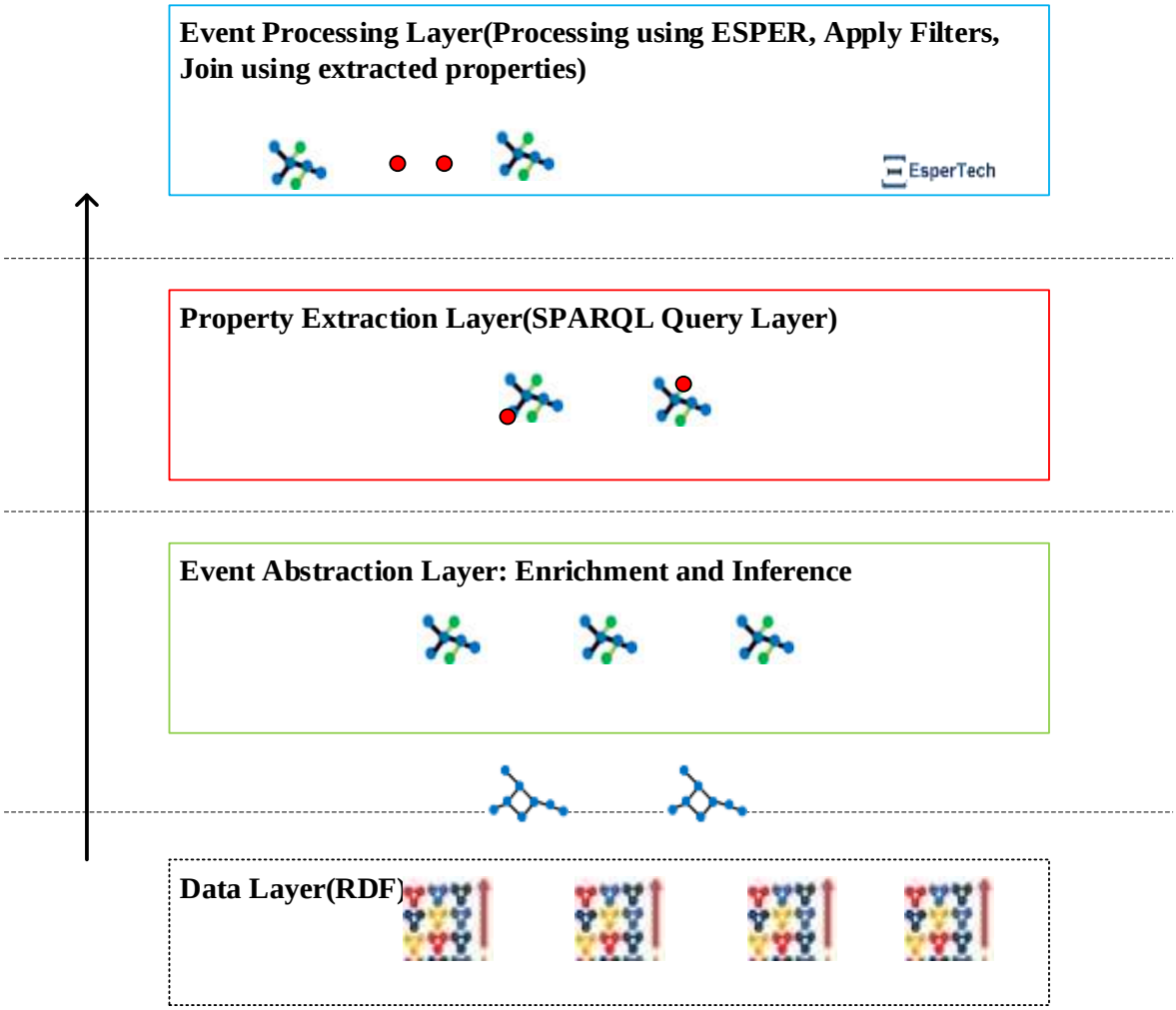

Figure 1-2 OCEM-IoT Architecture

OCEM-IoT architecture is presented in Figure 1-2. It consists of three main layers: 1) Event Abstraction Layer, 2) Property Extraction Layer and 3) Event Processing Layer.

OCEM-IoT supports complex event processing within an IoT environment by applying queries over the high level ontological concepts abstracted from RDF stream data.

1.8 Thesis Overview

The thesis is presented as follows: Chapter 2 provides the State of the Art and background knowledge in areas related to this research, including an introduction to IoT frameworks,

context awareness in IoT, streaming data processing, sensor data classification, Ontology data access, RDF streams and Complex Event processing.

Chapter 3 studies the methodologies used in developing ontologies in software engineering and deduces a set of guidelines to follow in this work.

Chapter 4 describes the IoT-ONT ontology and provides a detailed description of its' modules and concepts. And presents the "EPS-DL" an Event Processing Syntax using Description Logic, its syntax, semantics, and its representation and provide a use case using the Syntax.

Chapter 5 formally introduces the OCEM-IoT model of this thesis.

Chapter 6, presents the evaluation method and the evaluation of OCEM-IoT divided into two main sections, the evaluation of the IoT-Ont ontology and the event processing syntax. This chapter introduces the accomplished experimentation results and evaluation.

Chapter 7 presents the overall conclusions and future research directions.

CHAPTER 2: BACKGROUND KNOWLEDGE AND LITERATURE REVIEW

2.1 Introduction

The dissertation will focus on the use of sensor data sources, and proposes methods for querying them using semantic representations. This research explores the various background research areas, and the methods that have been suggested in recent years to address the research challenges.

This section presents the challenges faced by the research community in the field of IoT:

- A heterogenous environment, at all levels; data structures, devices, data schemas, and data formats provided by the sensors, this aspect makes it difficult to make use of the data provided and integrate it with various data sources.
- The nature of sensor data streams: the high dynamicity of this data poses essential challenges for today's query processing.

The challenges of data volume and velocity are intrinsic in such scenarios, besides the data variety in such environments. These three challenges have been studied since the emergence of the Big Data concept and are referred to as the 'Three V's' (Laney, 2001). This research addresses some of these challenges, with regard to the transformation of raw sensor data into meaningful information. This data will be modelled using the IoT-Ont ontology and will be accessed by queries based on the ontological concepts. The usage of ontologies as a method to provide explicit semantics for data has been adopted by many researchers to faciliate reuse, integration and reasoning over it. Furthermore, recent research in this field has developed continuous querying and event processing, besides the traditional database systems, which depended on relational and transactional principles. These two querying systems are important

to develop methods and techniques that can coherently combine them. By employing structured semantics on the data, adopting ontologies as a model for representing both sensor data and the metadata that describes it, is the basis of the research presented in this thesis.

State-of-the-art solutions in the sensor data processing context, model time relations to a certain limit; However, it does not address the data variety neither it addresses the domain complexity. Advancement in Semantic technologies made it possible to describe the aforementioned aspects. However, existing methodologies either do not provide a unified syntax to model the full processing or have restricted expressiveness (Taylor and Leidinger, 2015).

This chapter is organised as follows: Section 2.2 presents a general introduction to the Internet of Things (IoT) and its main characteristics. Section 2.3 introduces the context-awareness model for IoT. Section 2.4 introduces existing frameworks within an IoT environment.

Section 2.5 presents an introduction to Sensor Ontology Modelling, a background to the main ontology in this field the Semantic Sensor Network Ontology (SSN), and extended ontologies based on the SSN to accommodate an IoT environment and a comparison between these.

Section 2.6 introduces Event Processing and how it is used to detect changes in a highly dynamic environment with Big Data, Section 2.7 presents various Event Processing Models used for Big Data processing.

Section 2.8 introduces Semantic Based models that represent and query sensor data in terms of ontologies including the Resource Description Framework (RDF) and streams reasoning approaches using event processing languages such as SPARQL.

2.2 Internet of Things

2.2.1 What is Internet of Things?

Internet of Things is characterised by an ever-increasing number of Things embedding sensing, actuating, processing and communication capabilities. The most critical challenge for applications development in this field is dealing with a large scale number of users and Things while providing interoperability across the heterogonous things. Rivera and Goasduff in Gartner (2014) predicts that the number of connected things in the world will have a thirtyfold increase between 2009 and 2020.the Internet (Rivera and Goasduff, 2014).

Research on Internet of Things is in its infancy stages, there is no standard definition available yet for the term since it could be viewed from different perspectives: Internet-oriented, Things oriented and Semantic oriented. The term was first used in a presentation by MIT's Kevin Ashton in 1999, where he suggested adding Radio Frequency Identification RFID tags to objects would create the Internet of Things. This suggestion has already become a reality, though now it goes far beyond this concept. The integration of an ever-soaring number of objects connected to the internet. With the main aim of turning high-level interactions with the physical world into a simple interaction with a virtual equivalent and ultimately blending both worlds together (Hachem et al., 2014).

The literature review has shown that there is a number of definitions used to represent the IoT vision, as discussed below:

- The definition provided by the European Research Cluster of IoT is (Vermesan and Friess, 2014):

"A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual "things" have identities,

physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network."

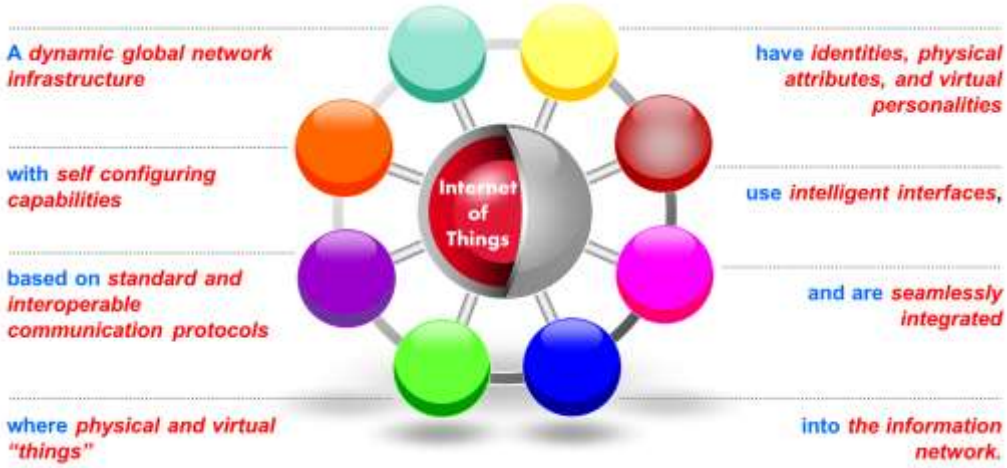

Figure 2-1 IoT Definition by IERC (Vermesan and Friess, 2014)

- The definition provided by the International Communication Union is (Union, 2012):

"Internet of things (IoT): A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.

NOTE 1 – Through the exploitation of identification, data capture, processing and communication capabilities, the IoT makes full use of things to offer services to all kinds of applications, whilst ensuring that security and privacy requirements are fulfilled.

NOTE 2 – From a broader perspective, the IoT can be perceived as a vision with technological and societal implications."

- The definition provided by Future Internet Assembly:

  "The IoT concept was initially based around enabling technologies such as Radio Frequency Identification (RFID) or wireless sensor and actuator networks (WSAN), but nowadays spawns a wide variety of devices with different computing and

communication capabilities – generically termed networked embedded devices (NED). While originating from applications such as supply chain management and logistics, IoT now targets multiple domains including automation, energy, e-health etc. More recent ideas have driven the IoT towards an all-encompassing vision to integrate the real world into the Internet – The Real World Internet (RWI)."

These definitions reveal that IoT makes use of a number of collaborating technologies ranging from networking to sensing. The advancement of numerous technologies for sensors, actuators and cloud computing, and the emergence of a new generation of cheap and small wireless devices, most objects used in our daily life could become wirelessly interoperable by attaching a low powered and passive wireless device such as RFID tags. By easing the access and interaction with a wide variety of things, IoT will promote the development of applications in many domains, such as industrial and agricultural automation, medical aids and emergencies, mobile healthcare, traffic management and many others.

2.2.2 Characteristics of an IoT Infrastructure

This section briefly discusses the main characteristics of an IoT infrastructure:

- Heterogeneity: IoT aims to interconnect large numbers of heterogeneous devices to provide advanced applications. The heterogeneity does not come only from differences in the capacity and the features of the things but from many other reasons such as multi-vendors' products and application requirements (Bellavista et al., 2013). Heterogeneity in IoT could be at any level of functionality within the IoT platform, it could be at the Internet or cloud level, at the high, middle and low-end computing hardware, besides the wireless sensors and actuators (Teixeira et al., 2011).

- Resource-constrained: Embedded sensors need a small device; this factor limits their processing, memory, and communication capacities. RFID tags may not have any processing capacities, whereas devices in the cloud are larger and more expensive and with high capacities (computational, connectivity, and memory) (Want, 2004).
- Spontaneous interaction: As objects move around and come into other objects communication range, sudden interactions can take place which leads to spontaneous generation of events (Razzaque et al., 2016).
- Extremely large scale network: Thousands of devices could interact with each other at the same time, and in one local place, these interactions between this enormous numbers of nodes will lead to the generation of an enormous number of events which could cause different problems such as congestion and a reduced event processing speed.
- Dynamic network and no infrastructure: Mobile nodes within an IoT infrastructure could leave and join the network at any time. Also, some nodes could get disconnected for lack of power. This leads to a very dynamic network infrastructure, and hence the nodes would need to cooperate in order to keep the network in working order (Razzaque et al., 2016).
- Context-aware: The large number of sensors in an IoT environment will generate a big amount of data, this data would not be useful unless analysed and interpreted correctly. Context awareness would help minimise the human interaction and the automation of the process, which is of a vital role in machine to machine communication (Cristea et al., 2013).
- Location-aware: The location of the objects and sensors are of a vital importance for context-aware computing. Interactions of the sensors are highly dependable on their location and the presence of other things.

- Distributed: IoT is distributed at both scales, globally like the internet and locally within the application domain (Banerjee et al., 2011).
- Intelligence: Intelligence is a key element in IoT, in this dynamic and ultra large network entities will be interoperable and should act independently based on the context, surroundings and environment (Kortuem et al., 2010, Kyriazis and Varvarigou, 2013).

Table 2-1 summarises the main characteristics of the IoT infrastructure.

Table 2-1 IoT Infrastructure Characteristics

Spontaneous Interactions	Location-Aware
Heterogeneity	Big Data
Distributed	Increased Security Risks
Ultra Large Network	Resource Constrained
Dynamic	Large Number of Events
No infrastructure	Intelligence

2.3 Context-Aware Internet of Things

A context includes any information that could be used to characterise the environment's situation, the place, the person and any object considered relevant to the interaction between the user and the application, including the user and the application itself. Context-aware IoT systems use context information to adapt their operations without an explicit user intervention and thus aim to increase usability and efficacy in order to provide users with better and more adapted services. This section, introduces the formal definitions of contexts and contextual reasoning.

Definitions of Context

Many researchers have defined the term 'context'. Dey et al. (2001) assessed the weaknesses of the definitions provided by researchers. They claimed that the definition provided by Schilit and Theimer (1994) was based on examples and could be used to identify new context. Dey et al. (2001), further claimed that definitions of Brown (Brown, 1995) Franklin and Flachsbart (Franklin and Flaschbart, 1998), Rodden et al. (Rodden et al., 1998) and Hull et al. (Hull et al., 1997), used synonyms for context, such as situation and environment (Dey et al., 2001).

Consequently, these definitions cannot be used to identify new context. According to Abowd and Mynatt (2000) identified the five W's (Who, What, Where, When, Why) as the minimum information needed to understand context (Abowd and Mynatt, 2000). The term context was also defined by Schilit et al. (Schilit et al., 1994) and Pascoe (Pascoe, 1998). Dey claimed that these definitions were too specific and could not be used in a generic sense to identify context and provided the following context definition (Dey et al., 2001):

"Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves" (Dey et al., 2001)

The definition of context provided by Dey et al. (2001) is widely used in the literature and is used in this thesis because this definition could be used to identify context from data in general. By considering a data element, we can determine whether the data element is context or not using this definition. The word context has also been defined and explained by a number of dictionaries as a circumstance, situation, phase and set of facts surrounding a situation or event. In addition, the difference between raw data and context information has been explained by Sanchez et al. (2006) as follows:

- Raw (sensor) data: The unprocessed data which is collected directly from the data source, such as sensors.
- Context information: Is produced after the processing of raw sensor data. Consistency will be checked and then metadata will be added to it.

For example, the sensor readings produced by temperature sensors can be considered as raw sensor data. Once we put the temperature sensor readings in such a way that it represents a temperature reading with units, we call it context information. The raw data values generated by sensors can therefore generally be considered as data. These data are identified as context only if it can be used to generate context information. Thus, most data that is captured from sensors is raw data and not the context information.

The notion of context is at the centre of IoT solutions. On the one hand, sensor data collection is used to obtain this contextual information. On the other hand, once the context is derived, events could be detected and reasoned about. However, it is not easy to represent and derive context in IoT. This is primarily because of the continuous aspect of the physical sensed data and the distributed and heterogeneous nature of IoT things.

2.3.1 Main Architectures within an IoT Environment

There are several approaches to achieving context-aware systems. These approaches depend on the system's requirements and conditions, such as the location of sensors (local or remote), the number of possible users (one or several users), the available resources of the devices used (small mobile devices or high-end-PCs). Chen et al. identifies three different methods to contextual information acquisition (Chen et al., 2003):

1. **Direct sensor access**: This method is often used in devices with locally integrated sensors. The client software collects the required information from these sensors

directly, i.e., there is no additional layer available to collect and process sensor data. It is therefore not suitable for distributed systems because of its direct access nature, which lacks a component capable of handling multiple simultaneous sensor accesses.

2. **Middleware infrastructure**: Modern software system uses encapsulation methods to separate graphical user interfaces and business logic. The middleware approach introduces a layered architecture to context- aware systems to conceal low-level sensory details.
3. **Context server**: The following logical step is to allow several clients access remote data sources. By introducing a component to manage remote access this distributed approach extends the middleware based architecture. The collection of sensor data is transferred to this so-called 'context server' which make multiple access easier. In addition to sensors reuse, the use of a context server removes the resource-intensive operations from the client's side.

2.3.2 Contextual Reasoning Challenges in IoT

The purpose of reasoning about context in an IoT environment is to make full use of raw context data; to process, combine and eventually translate the sensor low-level data into useful information. Based on which the system could determine the state of its context and react appropriately to certain context changes.

The imperfection and uncertainty of context information and the unique characteristics of the entities that function in IoT environments presents numerous challenges. Sensor or connectivity failures (which inevitably occur in wireless connections) lead to situations where not all context data is available at any time. When data on a context property comes from various sources, the context can then become ambiguous. Lack of precision is prevalent in sensor-derived information, while incorrect contexts arise as a result of hardware or human

errors. The entities operating within an IoT environment will have different objectives, experiences and perceptive capabilities. They might be using different vocabularies. Because of the dynamic and open nature of the IoT environment, many entities join and leave the environment at random times. In addition, the unreliability and the range restrictions of transmitters and wireless communications. IoT things do not typically know *a priori* all other entities present at a particular time nor can it communicate directly with all of them.

Overall, according to the literature, the role of reasoning about context includes (Perera et al., 2014, Bikakis and Antoniou, 2010):

1. To detect possible errors in the context information available.
2. To handle missing values.
3. To Evaluate the quality and the validity of the data sensed.
4. To transform the low-level raw context data into higher level meaningful information so that it can later be used in the application layer.
5. To make decisions about the system's behaviour when certain changes are detected in the system's context.

With regard to these particular characteristics of context reasoning in an IoT environment, the three main challenges, are to enable (Bikakis and Antoniou, 2010):

1. Reasoning with the highly dynamic and imperfect context.
2. Managing the potentially huge piece of context data, in a real-time fashion, considering the restricted computational and storage capabilities of some mobile devices, and the constraints imposed by wireless communications.
3. Collective intelligence, by supporting heterogeneous information sharing, and distributed reasoning with all the available context information.

2.4 Ontology based IoT Frameworks

In order to provide connectivity for sensors and actuators to the Internet several IoT frameworks were proposed to serve as a middleware solution. Many IoT frameworks exist nowadays, they mostly focus sensors and devices integration, there is very little attention given to a more intelligent IoT data processing (Barnaghi et al., 2012).

This section discusses the latest attempts to process data in an IoT environment, mainly through the use of semantics.

LinkSmart platform (Kostelnik et al., 2011) was developed in order to facilitate the integration of various devices, sensors and services and to support interoperability. It provides an abstraction of the devices and sensors as normal programming objects toward the application layer. It enables to compose workflows using business rules to optimise the service composition.

Patkos et al. (Patkos et al., 2010) propose an ambient intelligence framework that combines rule-based reasoning with causality-based reasoning, to reason about actions and causalities. The proposed framework does not provide capabilities to annotate raw IoT data. Gray et al. (Gray et al., 2011b) propose a system to annotate and integrate heterogeneous streaming data with stored data, through the use of ontologies. Their approach focuses on the discovery and integration of data sources, both static and streaming data. Reasoning and service collaboration techniques are not presented.

Sense2Web (De et al., 2012) is a multi-layer platform which enables the annotation and integration of sensor data as Linked Data, and makes it available to other Web applications via SPARQL endpoints. In Sense2Web the data source layer is modelled using expressive ontologies in order to retrieve high-level data. However, for the service and application layer, service collaboration and advanced reasoning capabilities are not provided.

XGSN (Calbimonte et al., 2014) is an end-to-end, semantically enabled IoT platform which enables semantic annotation of sensors, and processes the resultant sensor stream utilising the Linked Sensor middleware (LSM). The stream can be stored or processed using stream processors. External application may query the context through the use of Web Services. These applications, however, cannot share conclusions back to the context layer.

Ali et al. (Ali et al., 2015) propose an IoT-enabled communication system that enables sensory data annotation using the XGSN middleware and the continuous analysis of data streams using stream query processing modules for events' detection. These events can then be further processed in the Stream Reasoning component which can deduce implicit semantic statements. Applications can subscribe to events generated in the centralised stream processing and reasoning layer. However, they cannot collaborate to achieve complex workflows.

OpenIoT (Soldatos et al., 2015) is an open-source IoT platform enabling the semantic interoperability of IoT services in the cloud. It allows the integration and annotation of virtually any sensor.

LSM is utilised and acts as a cloud database which enables the storages of the annotated data streams. Services can access the annotated data through the use of SPARQL queries. However, service collaboration and advanced reasoning capabilities are lacking.

SOFIA2 (Indra, 2017) is an ontology-based Big Data IoT middleware, allowing interoperability and semantic annotation of several heterogeneous devices. It facilitates Complex Event Processing (CEP) to orchestrate the context data between context consumers. However, besides context subscription based on CEP, there is no real semantic reasoning feasible.

Some of these solutions do draw conclusions in an efficient reactive manner facilitated by annotating data semantically. Efficient data processing is often supported. Nonetheless, more

advanced reasoning capabilities are still missing. Furthermore, these platforms fail to mutually collaborate in a high-level manner. An overview of the ontology-based IoT frameworks discussed is summarised in Table 2-2.

Table 2-2 State of the Art of Ontology-Based IoT Platforms

PLATFORM	SEMANTIC ANNOTATION	INFERENCE	CONTEXT MODEL
Patkos et al. (Patkos et al., 2010)	/	Rules and causality	Central
LinkSmart (Kostelnik et al., 2011)	Yes	Rules	Central
Gray et al. (Gray et al., 2011b)	Yes	/	Central
Sense2Web (De et al., 2012)	Yes	/	Central
SeCoMan (Celdrán et al., 2014)	locations	Rules	Distributed
CASF (Kang and Park, 2013)	Yes	/	Distributed
SOFIA2 (Indra, 2017)	Yes	/	Central
XGSN (Calbimonte et al., 2014)	Yes	Basic Stream Processing	Central
Ali et al. (Ali et al., 2015)	Yes	Stream Processing Stream reasoning	Central
OpenIoT (Soldatos et al., 2015)	Yes	/	Central
OCEM-IoT	Yes	OWL DL	Distributed

2.5 Sensor Ontology Modelling

2.5.1 Sensor Data Ontologies

Sensor data and metadata modelling using ontologies have been studied widely in the latest years, this has resulted in the development of various sensor ontologies, a lot of these ontologies were centred mainly on descriptions of sensors, and on measurements provided (Compton et al., 2009).

Metadata is usually static and rarely changes over time since it contains information representing, the types of sensors, procedures, descriptions, value ranges, and technical capacities. The semantic metadata could be exploited for multiple reasons, such as enriched search, sensor configuration or dataset's discovery. Sensor observation data however is characterised by continuous updates and higher data volumes, and includes values captured for a certain feature provided by the sensors.

The initial ontology proposals for describing sensors were presented for wireless sensor networks by (Avancha et al., 2004, Eid et al., 2007, Goodwin and Russomanno, 2006) and others. However in these models, the main focus is limited to sensor's meta information, and description of observation is ignored (Compton et al., 2009). Furthermore, most of these models lack ontological design best practices such as alignment with reference ontologies and standards, and reusability. The OntoSensor ontology (Goodwin and Russomanno, 2006) is based on the concepts defined in the OGC[1] and SensorML[2] standards as a basis. Although it is concerned with syntactic representation that does not explicitly state semantics, the OGC has been leading the development task in order to standardise sensor metadata. The OntoSensor, implements for its class descriptions and properties, the concepts of Sensor, Sensor

[1] Open Geospatial Consortium http://www.opengeospatial.org/
[2] SensorML http://www.opengeospatial.org/standards/sensorml

Capabilities, Platform and Measurement and Observable. Moreover, it implements extensions to the upper ontology SUMO, and also proposes service extensions (Kim et al., 2008). OntoSensor is considered one of the most comprehensive ontologies in relation to sensor metadata coverage. However, it fails to implement observation data and does not follow the OGC Observations and Measurements (O&M) standard[3]. The ideas identified in the OGC SensorML standard were adopted by the SWAMO ontology (Witt et al., 2008), this ontology focuses also on a sensors' descriptions.

Barnaghiet al. (Barnaghi et al., 2009) proposed the extension of ontological models to include representations of the observations captured by sensor networks, this work was also based on the standard defined by the OGC for both the Observations and Measurements (O&M). Moreover, the ontology includes characteristics of sensor nodes such as operation modes, typical values and power supply. Nonetheless, it does not implement configuration and deployment concepts; it neither follows the ontology design principles of modularisation nor it reuses the existing vocabularies.

The researchers in (Eid et al., 2007), illustrate the SDO ontology for sensor data and hierarchy. The SDO is subdivided into multiple interconnected modules, the ontology is described as an extension for the SUMO[4] Ontology. The Extensions for upper-level ontologies such as SUMO is extremely interesting in ontology development since it allows the integration with other vocabularies defined previously and facilitates reuse of potential ones.

In (Compton et al., 2009) the researchers present the CSIRO sensor ontology, it includes, besides sensor properties and descriptions, the ability to represent sensor components and composition, which facilitates the representation of virtual sensors. However, data concepts and observations are neither detailed nor comprehensive in the ontology. The above-mentioned

[3] Observations and Measurements standards OGS http://www.opengeospatial.org/standards/om
[4] SUMO Ontology http://www.adampease.org/OP/

ontologies were deployed to describe and specify actions and complex events that run on an event processing engine (Taylor and Leidinger, 2015).

The evaluation for early ontologies covers their ability to describe: sensor measurements, sensor data, sensor types, sensor hierarchies, sensor systems, structure and composition, etc., and diagnose the limitations in these ontologies, basically regarding operating conditions to a satisfying level or response models, and the ability to describe configurations (Compton et al., 2009).

2.5.2 The Semantic Sensor Network Ontology

Based on the previous methods, and with the collaboration of the W3C SSN-XG group[5], the semantic web and sensor network communities have been working towards a generic ontology, a domain-independent ontology, which adapts to various use-cases, and conforms to the standards proposed by OGC at the sensor level (SensorML) and observation level (O&M). The result was the Semantic Sensor Network Ontology (SSN) (Compton et al., 2012), is based on the stimulus-sensor observation design pattern (Janowicz and Compton, 2010) and the OGC standards. The SSN ontology is aligned to the upper-level DOLCE Ultra Lite ontology[6] in order to improve matching and compatibility with other state-of-the-art ontologies, and as a W3C group initiative, a collaborative development process was followed.

The SSN ontology presented in Figure 2-2 (Compton et al., 2012) is used for describing sensors, their functionalities, and for the processing of the external stimuli. Moreover, it could be centred on the data observations, and the related metadata, or in the platforms and systems of a sensing deployment. The SSN modules reflect these distinct perspectives. The key module

[5] SSN Ontology https://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[6] DOLCE Ultra Lite Ontology http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

Skeleton comprises of concepts used in all the use cases, such as sensors, and the observations they generate, which includes properties being observed for a certain features of interest.

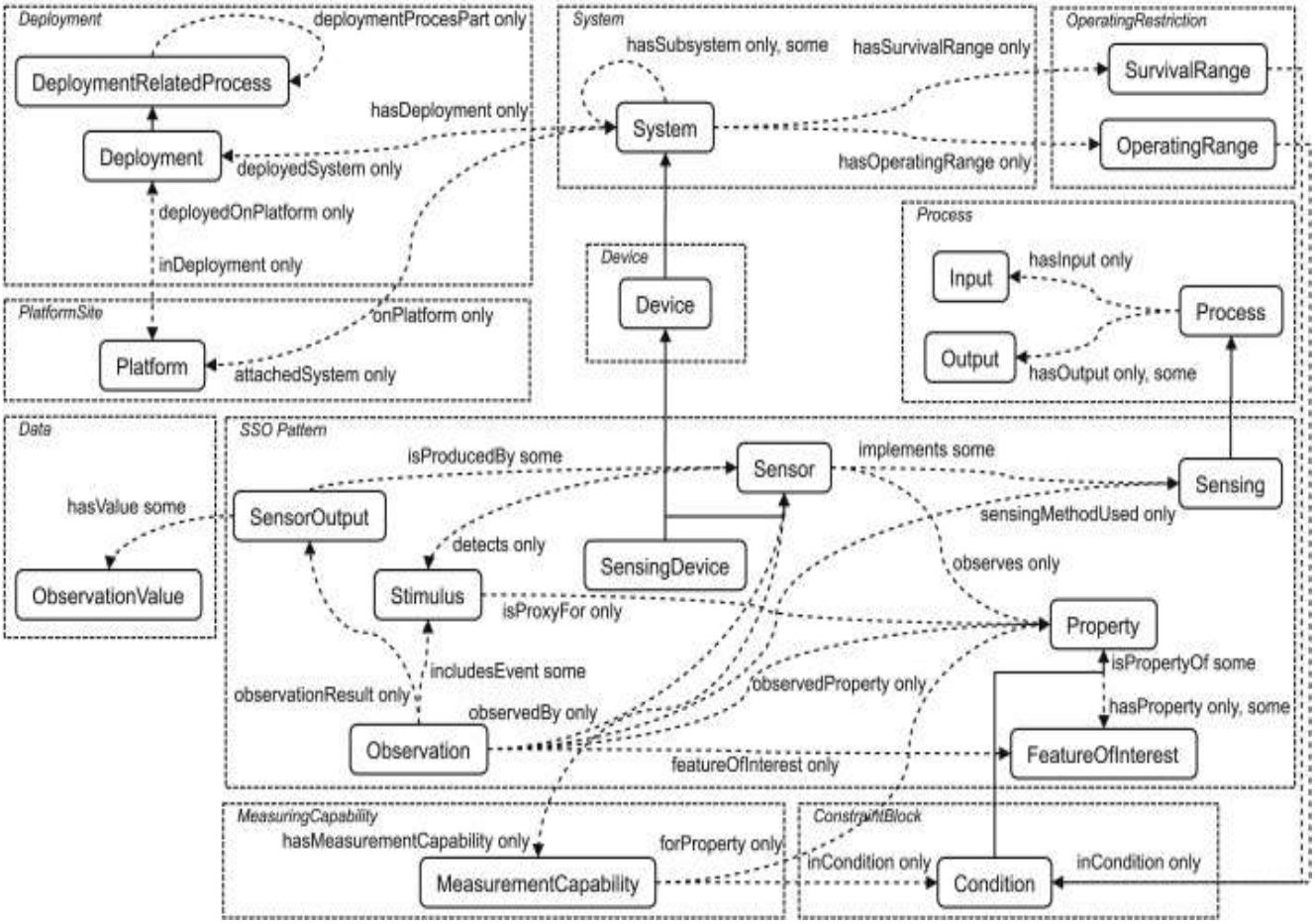

Figure 2-2 SSN Ontology, Key concepts and relations (Compton et al., 2012)

For illustration, consider a humidity sensor deployed within a weather station at a certain location. The humidity sensor is used to measure the humidity level on its specific field site. Consider a different sensor in the same weather station records precipitation every 10 minutes. We can represent the two sensors, by utilising the SSN ontology, as instances of ssn:Sensor, each one holding a different Uniform Resource Identifier (URI) as shown in Listing 2-1. Using the ssn:onPlatform property each sensor could be linked to a weather station platform, e.g. Salford platform, where the sensors are deployed. The platform is geo-spatially located, using

the WGS84 vocabulary[7]. In this example, the location's latitude and longitude coordinates of the platform are supplied, as a geographical point. Moreover, other information can be included such as an initial date of the deployment, the responsible person etc.

Listing 2-1 Representaion of a Humidity Sensor on a platform Station

```
salfweather:sensorHumidity2
rdf:type ssn:Sensor;
ssn:onPlatform salfweather:platformSalford;
ssn:observes cf-property:Humidity_level.
salfweather:sensorPercipitation1
rdf:type ssn:Sensor;
ssn:onPlatform salfweather:platformSalford1;
ssn:observes cf-property:percipitation
salfweather:platformSalford
ngeo:hasGeometry [ rdf:type wgs84:Point;
wgs84:lat " 53.4858 ";
wgs84:long " 2.3836 " ].
```

In this example the "ssn:observes" property is used to describe the type of observed property. Since the SSN ontology is a domain independent ontology, the real values of the observed property are obtained from specific vocabularies, such as the Climate & Forecast properties ontology[8]. Consequently, with regard to the humidity sensor, the "cf:property:humidity_level" term is used to describe the humidity level observed property. Using the SSN ontology's terminology, the measurements for the humidity and precipitation could be viewed as observations, each one of them represents a specific feature of interest (humidity and precipitation in this case), and each one refers to a specific property (humidity and precipitation). Instances of the Observation class in the SSN ontology describe such

[7] Basic Geo WGS84(lat/long Vocabulary)https://www.w3.org/2003/01/geo/
[8] Climate and Forecast Vocabulary https://www.w3.org/2005/Incubator/ssn/ssnx/cf/cf-property

observations and are connected through a “featureOfInterest” property to a certain feature instance (see Figure 2-3). Likewise, the “observedProperty” connects to an instance of a property, such as humidity level. Obviously, further information regarding the observation could be captured, this includes noise, units, precision, and failures. Note that the approach of ontology modelling requires the reuse of the SSN ontology combined with the domain-specific ontologies.

2.5.3 Extending the SSN Ontology

Most ontologies established after the SSN ontologies for wireless sensor networks or to be employed in an IoT environment, were in fact extensions of SSN ontology. The main extension directions were mainly for the following reasons: (Wang et al., 2015) (Bauer et al., 2019)

1. Extensions according to a specific domain or application.
2. Add new concepts to SSN ontology to develop a more general ontology.
3. Combine with the other ontology to solve actual problem.

This section will briefly present ontologies developed for these reasons.

(1) Extending According to a Specific Domain

SSN ontology is extended according to the specific requirements of different applications. The extended ontologies add new concepts on SSN ontology or change SSN ontology for the specific application so that they can describe the application field more clearly.

NIST ontology (Schlenoff et al., 2013) applies to the manufacturing industry. In NIST ontology, SSN ontology is extended according to a series of specific requirements. On the basis of SSN ontology, function, domain, additional entities, resource group, status, purpose, physical location, etc. are added in NIST ontology. NIST ontology describes the detailed sensor

dimension, weights, resolution and the knowledge representation standard of the operating conditions, the representation of the system ability, and the sensor network of the manufacturing environment etc.

SWROAO (Wang et al., 2011) ontology is used on atmosphere observation. The platform class of SSN ontology is extended. In SWROAO ontology, three subclasses are derived from the platform class. They are aircraft class, ground class and spacecraft class. Each class is subdivided. Aircraft class includes LEO satellite and high orbit satellite, ground class includes the ground remote sensing platform, the ground atmospheric boundary layer observation platform and the mobile ground atmospheric observation platform, and spacecraft class includes balloon, aircraft and rocket, etc. The subdivision of these specific platforms makes SWROAO ontology carry out different query or decision depending on different platforms in the applications.

AEMET ontology (Atemezing et al., 2013) applies to the meteorological forecast of Spanish meteorological bureau. It extends from SSN ontology. In AEMET ontology, measurement ontology is related to meteorological observation. The main conceptes reuse “ssn: Observation”, “ssn: FeatureOfInterest” and “ssn: Property”of SSN ontology. But the SSN ontology concepts will be applied to a specific instance. Such as the concept "SSN: Property” is extended and added the specific properties of meteorological aspect in AEMET ontology. The concept “ssn: Sensor” is expanded to a hierarchy of types of sensors in the AEMET ontology and “ssn: Platform” is extended by “aemet: WeatherStation”.

(2) Extending by Adding New Contents to SSN Ontology to Found More General Ontology

The more general ontologies are extended on the basis of SSN ontology in order to have more extensively applicable scope.

WSSN ontology (Bendadouche et al., 2012) enriches SSN ontology. It describes communication by a kind of new pattern and integrates new concepts relating to the wireless sensor network in SSN ontology. WSSN ontology uses a new pattern: SWSNC (Stimulus-WSNnode-Communication) ontology design pattern. This pattern treats the stimulus as the starting point of any process and the trigger of sensor or communication equipment. In the observed phenomenon, stimulation can exist as the measurable change and the unconscious stimulation of a reaction process. Stimulation can also be used as a conscious stimulus in the communication process, and it is an input communication needs. WSSN ontology adds some new concepts of wireless sensor network, such as communication, data flow and state.

SCO ontology (Shi et al., 2012) is also built by reusing and extending SSN ontology. The Sensor module in SCO ontology is the core module which is a bridge connecting to other modules. Sensor module directly reuses the Sensor module of SSN ontology, but the Stimulus class is replaced by SensorInput class. The OpeatingRestriction module is changed to the SystematicCharacter module, and EnvironmentCondition class is added in the module. In addition, on the basis of SSN ontology modules, new modules such as the Component module, the Service module and the Context module, etc. are added. In the Context module, three important classes are added: Space, Time and Theme. These three classes respectively show that Space, Time, and Theme of observations values by which the sensor data information can be described in more detail. SCO ontology is a major progress of SSN ontology extensions.

In order to conduct semantic description of the sensor data on the sensor cloud, SCO (Sensor Cloud Ontology) Ontology appears (Müller et al., 2013). SCO Ontology reuses and extends SSN ontology, and many classes and properties don't have to be redefined. Through the derived subclass, the platform and network and other characteristics of sensor network on the Cloud can be defined. In SCO ontology, the Network class and Platform class derived from the System class of SSN ontology, the Sensor class derived from the SensingDevice class, while

the ObservedPhenomenon class derived from the Observation class, the Observation Result class derived from the SensorOutput

Kim et al. (Kim and Kim, 2015) propose an ontology based on the information provided by mobile device sensors, both physical (e.g. Wi-Fi, Bluetooth, etc.) and virtual (e.g., user schedule, weblogs, etc.) to support context-aware services. The proposed ontology defines the relations between different user locations and the contexts identified. The authors further propose a reasoner upon this ontology and evaluate it to identify locations. The result shows that their reasoner has higher location accuracy than the GPS locations.

(3) Extending by Combining with the other Ontology to Solve Actual Problem

Because the data of Semantic Sensor Web is linked-data, the ontology combination can make the application range greatly extend. The combination of SSN ontology and the other ontologies is necessary.

Alasdair Gray (Gray et al., 2011a) proposed SemSorGrid4Env ontology network to apply in flood emergency prediction scheme. This ontology network is composed of different ontologies. It is divided into four layers: the ontology of the specific fields, the information ontology, the upper ontologies and the external ontologies. These ontologies meet different knowledge representation requirements, and they share and reuse between each other. SSN ontology reuses the DOLCE + DnS UltraLite and SWEET upper ontology. The ontologies in the flood field are the basis of SSN ontology. They also reuse the external ontology. These ontologies combine with each other and play the biggest role in order to solve the flood emergency prediction (Wang et al., 2015).

The Smart Appliance REFerence (SAREF) ontology (Daniele et al., 2015) exists in the domain of smart appliances and aims to reuse and align concepts and relationships in existing appliance-based ontologies. The concept of functions: one or more commands supported by

devices (sensors) defined in SAREF, supports modularity and extensibility of the ontology, and also helps in maintenance of the appliances.

Hirmer et al. (Hirmer et al., 2016) propose an ontology to support dynamic registration and bindings of new sensors to a platform. They borrow the concepts of Sensors, Things and their associated properties from SensorML, and introduce an additional concept of "Adapter" associated with every sensor. The borrowed concepts support sensor discovery while the newly introduced concepts provide/compute additional information about sensor data. For example, adapters are used to compute the average quality of sensor data values from the quality of the sensor provided by the manufacturer and the staleness of the values generated by the sensor.

AEMET ontology is also formed by multiple ontologies (Atemezing et al., 2013).The ontology includes four modules: sensor, time, location and measurement. The Time module reuses the OWL Time Ontology, and the location Ontology reuses part of geobuddies ontology network, and the measure ontology reuses the concepts of SSN Ontology. All these ontologies are used to the meteorological forecasting of the Spanish meteorological bureau. They can transform the meteorological data to linked-data and have more comprehensive description (Wang et al., 2015).

2.5.4 Existing IoT Ontologies

Researchers have developed various sensor ontologies that seek to resolve the issue of heterogeneous software, hardware and data management. Table 2-3 classifies the various sensor-based ontologies according to the problems which they address. It is important to note that 'sensor data' can either signify the raw data produced by perceiving the phenomenon, or it can refer to the metadata information which describes the capabilities of the sensor (for example, its accuracy or range of coverage). In this context, the term 'sensor data' is reserved for the raw data, while 'sensor capabilities' is used for the metadata.

Table 2-3 Areas tackled by existing ontologies

Sensor Ontologies				
Sensor Discovery	**Data Access and Sharing**	**Sensor Capabilities**	**Sensor Data Description**	**Extensibility**
IoT-Lite (Bermudez-Edo et al., 2016)	Xue et al. (Xue et al., 2015)	IoT-Lite (Bermudez-Edo et al., 2016)	(Gyrard et al., 2014b)	Brick (Balaji et al., 2016)
Dynamic Ontology-Based Sensor Binding (Hirmer et al., 2016)	Sensor Core Ontology (SCO), (Shi et al., 2012)	Xue et al. (Xue et al., 2015)	(Compton et al., 2012)	SAREF (Daniele et al., 2015)
Brick (Balaji et al., 2016)		OntoSensor (Russomanno et al., 2005)	MyOntoSens (Nachabe et al., 2015)	

In Chapter 3, we present a more comprehensive comparison of recent Ontologies for IoT based on their conceptual and functional requirements, such as reusability and capacity of extension.

2.6 Introduction to Event Processing

Event Processing (EP) is a software engineering discipline that developed a set of technologies and tools to allow real-time computing. EP techniques are based on principles of event programming. The EP architecture is an important concept in EP that helps us to better understand and control computing in EP applications. This section introduces EP, including

the fundamental principles and the EP architecture. To understand what EP is about, this section starts by explaining what an event is.

Event Definition: An *event* is defined as "an occurrence within a particular system or domain; it is something that has happened or is contemplated as having happened in that domain" (Etzion et al., 2011a).

The word event is also used to mean a programming entity that represents such an occurrence in a computing system. From this definition, we see that events are considered within a particular domain. In many cases, the domain endows events with a context in which they are interpreted. In the first part of the definition, an event denotes something that happened in the real, physical world. The second part, however, treats an event as a programming entity in a computing system. Indeed, an event is a general term, and in most of the applications by observing the physical world surges the need for means to represent and process these observations in a computing system.

In practice, an event represents something that occurs, happens or changes the current state of affairs. For example, an event may signify a problem or an impending problem, a threshold, an opportunity, an information becoming available, a deviation and so forth. These events are directly related to specific, measurable changes of conditions. In many application, it is, however, important to infer more abstract situations. It is the task of EP to effectively derive these situations based on (many) single, ordinary events.

2.6.1 Events: A Technique to Declare a Change

Declarative programming involves declaring facts about things, clarifying whether they are true or false. These facts can help in the implementation of an inference procedure to determine the truth about other things whose truth is not stated explicitly. It is, however, necessary to not

specify who might use the fact about something being true or false and to avoid specifying when a fact could be used to deduce another fact.

In a similar manner, an event in event-driven programming suggests the occurrence of something. This means that an event producer announces that something has happened or the current state of affairs has changed. It is, however, unclear as to who uses this fact. Moreover, it is not established when this fact can be used and when another fact can perhaps be inferred. It is also not predetermined what number of facts could be derived from that fact. (Etzion et al., 2011a) stated that, in Event Processing (EP), the concept of decoupling refers to an event's declaration being independent from possible consequences it might produce (such as being used to derive an unspecified number of events or to utilise for other purpose with no impact on that event's cause). Hence, event-driven programming has a close relationship with the declarative programming principles, while programming based on request-response interactions has an analogous relationship with imperative programming. In this case, a requester can ask to conduct particular processing when posting the request, and it is similar to an imperative program's implementation of a series of commands when calling a sequence. In event-driven programming and in imperative programming, it is important to consider the order in which requests are being processed.

2.6.2 Event Processing Architecture

EP refers to computing which executes operations on events, with the typical event processing operations involving events being read, transformed, created, as well as deleted (Etzion et al., 2011a).

As noted by Luckham (2011), Event Processing Technical Society (EPTS) gives a similar definition of EP (Luckham, 2011). It should also be noted that scientific literature uses the term Complex Event Processing (CEP) for indicating operations similar to EP that are conducted on

complicated events (Luckham, 2011). The two terms are often used interchangeably. The operations on events which this thesis examines can also be applied without limitation to complex events. In this section, an in-depth explanation on EP is presented.

Event processing architecture refers to a pattern of software architecture that promotes events' reading, transformation, creation, as well as deletion. The question here is which architectural pattern can be implemented by the design and implementation of EP applications?. There is a difference in applications based on the domains they are applied to and the requirements they must fulfil. On the other hand, all EP applications include a set of common entities that are arranged in an event processing network (EPN). An EPN includes entities such as event producers and consumers, with EP functioning as an intermediate processing between them. EPN's three major building blocks are illustrated in Figure 2-3. In the following, we briefly present each of these three entities (Etzion, 2010).

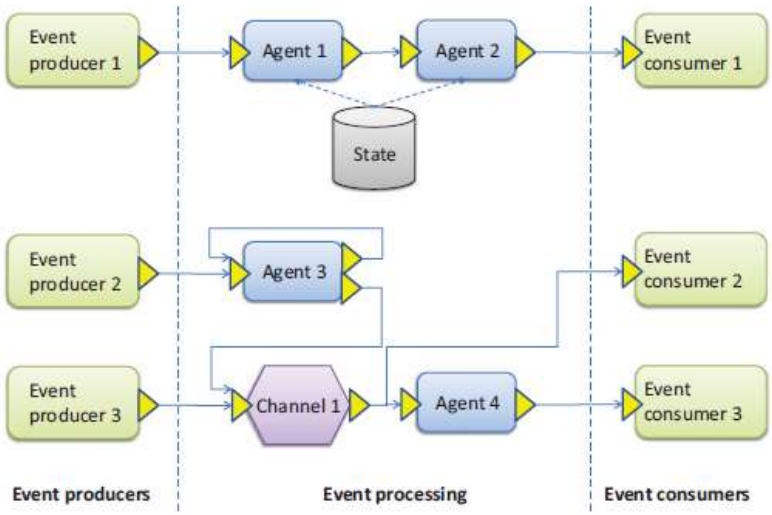

Figure 2-3 Event Processing Network (Etzion, 2010)

Further, an Event Producer, also called an event source, indicates an entity that introduces events in a system implementing EP architecture. In addition, an event producer is responsible

for ‘listening’ to the environment to which it is attached, providing events from that environment to a related event processing agent (EPA), as shown in Figure 2-3. It is possible for an event producer, for example, to be attached to a physical sensor, as it will help the sensor identify a change since the producer develops an object that indicates the change and provides it as an event. An event producer can function simply as a proxy communicating events from other places in other instances.

An event consumer can be considered as an entity receiving events from a system that uses an EPA. Moreover, an event consumer “consumes” events; for example, it reads events and utilises them for further computation, visualisation, or business analytics. It can also write events into a log or even trigger actions from events, such as creating other events or calling a particular service.

The EP in fact refers to the “processing” which occurs between the consumers and producers, as shown in Figure 2-3. This processing includes operations conducted on events such as events reading, transforming, creating, and deleting. This processing is typically not monolithic and comprises a set of agents, all of whom perform particular operations on events.

An *EPA* is a software module that processes events (Etzion, 2010, Luckham, 2011). In an EPA, events are taken as input and an EP operation is performed to output more complex (or derived) events.

Authors in (Etzion, 2010) classified EPAs into three major agents:

- *Filter agent* is responsible for filtering irrelevant events with regard to a specific filtering condition. An agent, for example, can filter stock price events with a price less than £100. Filter agents aim to eliminate events that are considered uninteresting in order to improve the performance of an EP application.

- *Pattern detection* agent identifies an event pattern based on particular conditions such as spatial, temporal, and other semantic relations that might be established between events. For example, an agent can observe a number of stock events occurring and determine the increase in stock price, indicating an *x*% increase in price during a given time based on an event pattern.
- *Transformation agent:* transforms input events according to transformation operations.

Transformation agents can be further classified based on transformation operations:

- *Split agent* – takes one event as input and outputs two or more copies of the input event.
- *Aggregate agent* takes numerous events as input and then performs an aggregation function on these input events to produce one derived event.
- *Compose agent* – combines two event streams in a similar manner two relations are joined in relational algebra.
- *Translate agent* translates an input event based on a translation operation into an output event.

A translation operation can take one of the following two forms:

- *Enrich agent* appends an input event using additional information obtained from an external information source such as an ontology or a database.
- *Project agent* is responsible for eliminating particular information carried by an input event, which is similar to how a relational algebra relationship is projected based on specific attributes.

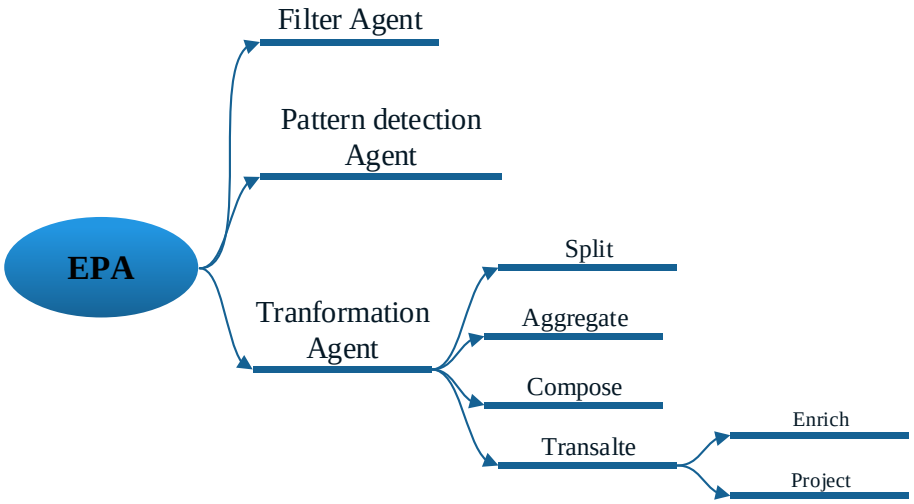

Figure 2-4 EPA Classification

EPAs are connected to event producers and consumers, as well as to themselves through event channels (denoted by directed edges in Figure 2-3). As noted by Etzion (2010), An event channel can be regarded as processing element which receives events from one or multiple source processing elements, defines routing decisions and sends the input events unchanged to one or more target processing elements in accordance with these routing decisions (Etzion, 2010).

Figure 2-4 summarises the EPA classification in literature. An EPN concept can be used for two reasons. The first one concerns the usability when an EP application is being designed. Namely, an architect may have better understanding of the application when the internal intermediate EP is represented as an EPN (Etzion, 2010).

The second reason is related to *efficiency*. An EPN that is deliberately designed can help in decreasing event flow in the network, thus improving the overall performance of the application. Etzion (2010) also stated that an EPN can help EPAs to be implemented on a distributed architecture, thereby improving the application's scalability and performance (Etzion, 2010).

An *event processing language* allows a higher-level specification of an EPN. In certain EP languages, a user is able to develop event flows mapped in an EPN. In others, language statements are written by a user and then compiled into an EPN.

2.7 Streaming Data Processing

The processing of Streaming data focus on data steams querying and management. These data streams could be considered as endless data values varying over time. Examples of data streams include cardiac measurements, inventory tickers or wave elevation observations from a marine sensor network. Such streams could be abstracted in a more elaborate form, such as complex events, where varying patterns could be identified and queried, and subsequently sent to a subscriber if they are relevant (Cugola and Margara, 2012). These types of applications vary from traditional data query and management systems, in recent years different techniques and solutions have been suggested, defining the distinct research areas of event processing and streaming data.

Data streams inherent characteristics and features present challenges within the streaming query processing field, lots of these issues have been discussed in several prior research (Barnaghi et al., 2014). This section provides a brief background summary of the event and stream processing models, languages developed for querying data streams and the implementations of streaming engine system outlined in the literature.

Event Processing Models

Streaming data management differ greatly from traditional static data due to key characteristic features of these streams, such as its infinite nature and the need for continuous data evaluation over time. Database systems largely involve stored and static data, with their queries retrieving

the state of the data at a certain time, compared to the constant data evolution monitoring over time in event and streaming data processing.

Research in this field has examined significantly dynamic data processing from various perspectives that can be classified into Complex Event Processing and Data Stream Processing. Both these processing types differ considerably in the data requirement types from traditional databases. Considering such streaming and event-based applications, the more recent values tend to be more relevant compared to older values. Query processing often focuses on real-time data and current observations, whereas, historical data is outlined, aggregated, and stored to be analysed at a later stage. Therefore, data recency is regarded as a primary factor for designing such systems, this include prioritising time in their data models and processing algorithms.

This domain has another important feature, which is the real-time and uninterrupted processing and delivery of data. As stated by Arasu et al. (2016), this is based on the principle of queries continuous evaluation over the streams, as opposed to the current stored relational models (Arasu et al., 2016). The stream source, such as a sensor, pushes streaming values at unknown rates and without explicit control of data arrival (Garofalakis et al., 2016). Following this, a query processor must regularly observe the tuples' arrival and determine whether they match with one or more queries already registered in the system.

In the following, some of significant aspects concerning data streaming and event models, continuous queries and recency-aware operators are presented.

Stream Data Model

As the streaming data processing field has developed from relational database research, hence in the first proposals streams were only considered as relations that were sometimes restricted to append-only updates, similar to Tapestry (Terry et al., 1992). Based on the database field, it

was convenient to use unbounded relational models for representing streams as it keeps the query languages and operators intact and regarding only the new characteristic as an extension. Although Tapestry's approach was limited concerning query expressiveness and append-only tables, it suggests streams being sequences of data tuples occurring at a particular order.

An extensively used data model views a stream as an unbounded sequence of tuples of continuously appended values (Golab and Özsu, 2003), with every value including a time-based index, which is typically a sort of timestamp. The timestamp forces a sequential order and usually signifies the production of a tuple, although this notion could be extended to regard the sampling, arrival or observation time.

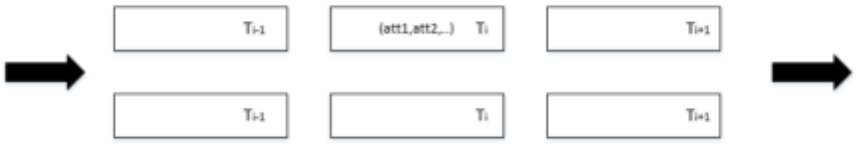

Figure 2-5 Data stream tuples

The tuple involves named attributes, including the atti shown in Figure 2-5 as per a schema, with every one of them involving a particular data type. It should be noted that numerous tuples can exist for every non-decreasing timestamp 'T'. Numerous domains and use cases tend to use this model and its variants. Systems implementing this concept include Aurora (Abadi et al., 2007), STREAM (Arasu et al., 2016), and TelegraphCQ (Chandrasekaran et al., 2003).

Event Data Model

The event processing and data streams fields have considerable differences in terms of their data model. Events, unlike table-like structures and schemas that focus on raw data management, are complicated abstractions of situations and observations that are modelled based on a particular use domain (Cugola and Margara, 2012). Events are typically notified to

subscribers if they are relevant to them (Chandy et al., 2011). Studies on event data management as well as query processing have been conducted since the appearance of Active Object databases that defined events as "something that happens at a point in time" (LeBlanc et al., 2017, Paton and Díaz, 1999). Events could more specifically be characterised by their source: e.g. generated by an operation, invocation, transaction, and time-based. Moreover, events can include a type, such as composite or primitive, which helps in differentiating them and indicating its semantics. A Windspeed_Reading type event, for example, can indicate a wind speed observation that is different from an AtmosphericPressure_Reading. In addition, notifications may be provided to subscribers based on the events' types, such as topic-based or type-based subscription. This, however, is a limiting method because events can include hierarchies and can generate complex as well as composite events.

To conduct complex event query processing, it is vital to consider the relationships between events with regard to causality and time (simultaneousness, occurrence in an interval, and precedence) (Mei and Madden, 2009, Jayasekara et al., 2015). Common event relationships are presented in Figure 2-6. The first relationship denotes the occurrence of event E1 prior to E2. These events might correspond to a specific point in time or to a time-interval, thus enabling an overlap relationship to exist. The second relationship indicates causality, where E1 results in E2, thus indicating precedence, the emphasis however is on the cause of the resulting event. The final example refers to the aggregation of multiple events (E1, E2, E3) into a composite event, E4. Although this situation does not specify the aggregation's nature, it implies the probability of developing complex events based on simpler ones.

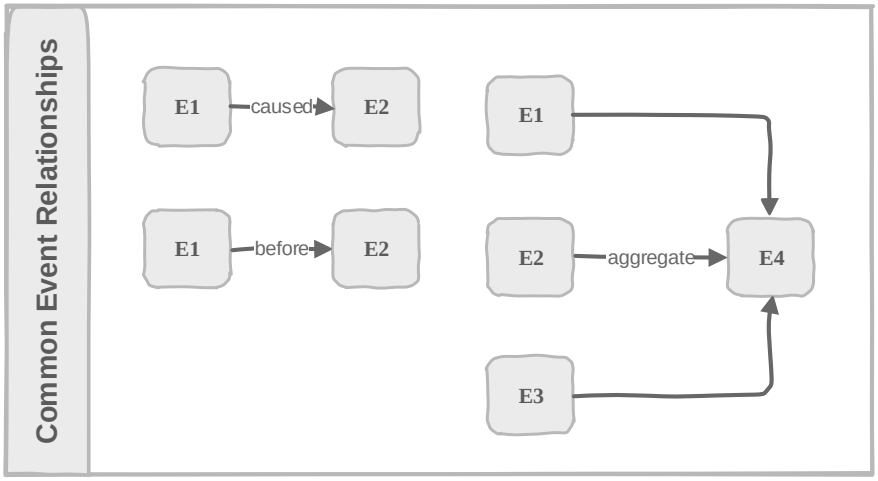

Figure 2-6 Common Events Relationships

Complex events can also be encoded by using other relationships including simultaneity, negation, occurrence in a certain interval, and disjunction. Event filters can be regarded as an advanced method through which users can be notified of the events matching a provided criteria. Such criteria can also be defined as expressions using a filtering language, such as conditions on an event's type and attributes.

Complex events surpasses filtering and enables the definition of new event types based on existing events as well as developing event patterns or pattern templates, which include temporal constraints, ordering conditions, and repetition (Song et al., 2015, Cugola and Margara, 2012). A complex event generation model is depicted in Figure 2-7. Streaming and processing of the incoming events is conducted in accordance with event models and rules matching the input, and producing complex events as outputs.

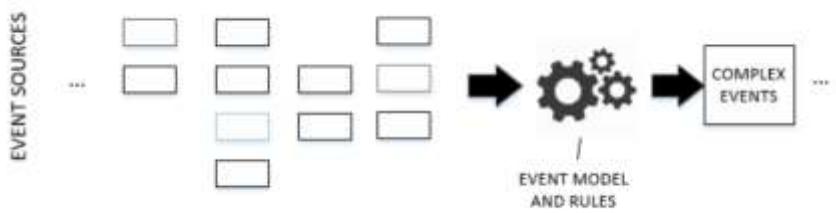

Figure 2-7 Complex Event Generation Model

Though there may be differences in the data model of streams and events, there are also similarities between them. For example, events tend to include interval information, a time-stamp and values of the attributes and depend on basic types of data. There are also certain systems that can define events in terms of queries or compositions of patterns and filters over relational streams.

Continuous processing. The concept of continuous or standing queries was introduced in (Terry et al., 1992) as "queries issued once and [...] run continually over the database". The development of this concept is based on assuming the use of an append-only database, in which traditional querying methods are not applicable because of the ineffectiveness resulting from big volumes and velocity of data. The resulting Tapestry system can periodically execute monotonic queries, where every evaluation considers only new tuples in order to avoid duplicates and to guarantee a deterministic and complete behaviour.

In the subsequent works, continuous processing has changed the execution model in stream-based systems, where data arrival initiates query processing, as opposed to stored relational databases. The result in a traditional one-off database query is instantly returned following the query evaluation, as shown in Figure 2-8 (leftside). On the other hand, the query processor involves continuous querying that delivers results when the streaming data matches the query

criteria, as shown in Figure 2-8 (rightside). Following this, the query processor can push the data when it is available to the clients or the client can actively retrieve the data in pull mode.

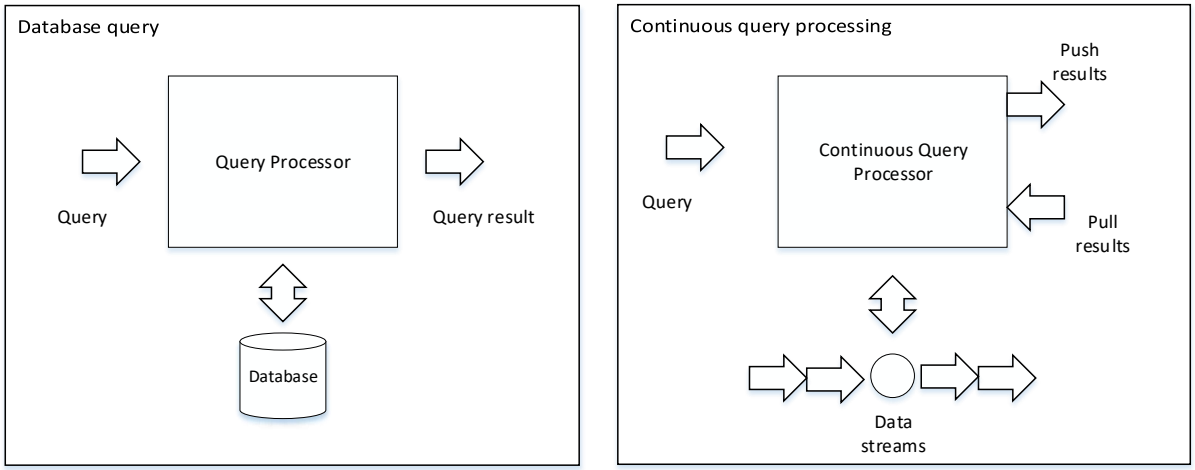

Figure 2-8 Relational vs Continuous query processing

Through sharing and adaptability, TelegraphCQ is capable of handling big numbers of simultaneous queries.

Golab and Özsu (2003) further developed continuous queries, surpassing one-time queries or triggers presented in the Alert system (Schreier et al., 1991) or in the OpenCQ trigger-stop queries (Liu et al., 1999) and they included join semantics as well as optimisations definitions (Golab and Özsu, 2003). There are certain challenges that have been addressed including the possibility of simultaneously handling hundreds of standing queries, sharing data between query operators, performing static, and runtime optimisations. NiagaraCQ (Chen et al., 2000) examined and exploited the concept of combining similar queries based on their signature and sharing a global plan within the group for XML-based data streams. Eddy operators were used by TelegraphCQ (Chandrasekaran et al., 2003) which was based on previous systems CACQ (Madden et al., 2002) and PSoup (Chandrasekaran et al., 2003), to route tuples among query operators. Moreover, it promoted sharing temporary repositories of homogeneous tuples

through state modules., TelegraphCQ, through adaptability and sharing was capable of handling large numbers of simultaneous queries.

Windows. As noted by Garofalakis et al. (2016), the sliding window model is the most common and widespread norm on stream data processing (Garofalakis et al., 2016). The Windows idea restricts the tuples' scope by the query operators. As shown in Figure 2-9, for example, the window has a starting point as well as an offset that restricts its scope over particular stream tuples. A subsequent window 1 limits data tuples in the stream's latter interval. It should be noted that windows can instantly follow one another, overlap, and become non-contiguous regarding time.

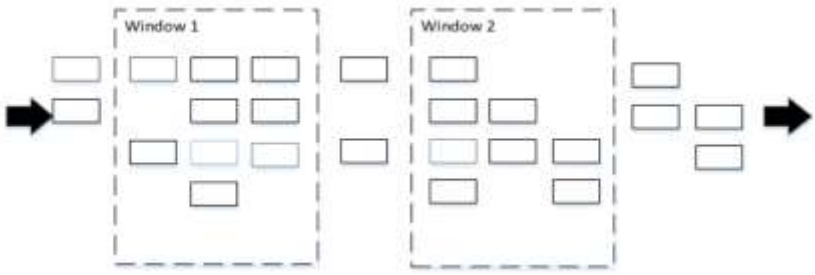

Figure 2-9 Window Operator

Typically, windows are described in terms of width or offset as well as a slide indicating the periodicity used to create a new window. For example, considering a query which requests the average temperature within the latest five minutes, it is possible to establish a window starting from the present time to the previous five minutes, while excluding all the remaining data. On the other hand, regarding continuous queries, over time, the window slides, which means that the window boundaries move for every execution time. Various systems such as Aurora, NiagaraCQ, STREAM, TelegraphCQ, or Tribeca use variations of windows including sliding windows, landmark windows, tumbling windows, and update-interval windows. There are

many cases, however, where such systems netiher include precise semantics regarding the window operators nor the relationships between them. Arasu et al. (2016) provided an abstract semantics by introducing the concept of stream-to-relation operators (Arasu et al., 2016). These operators transform a stream into a relationship with the same schema. Different types of windows were successfully defined by this simple conceptualisation: (Calbimonte, 2013)

1. Time-based windows: Given a time interval T over a stream s, the output relation is defined by the window of size T, sliding over time. Moreover, the interval's specification can enable the definition of tumbling windows or punctual windows.
2. Tuple-based windows: Given an integer n over a stream s, the output is defined as a relation of size n of the latest values of s.
3. Partitioned windows: Given an integer n and a subset a of the attributes of a stream s, it divides s in sub-streams, with one sub-stream for every attribute of ~a. Every sub-stream groups the tuples having the same value on the corresponding attribute of ~a, which is further limited in number by n. All windowed sub-streams are then combined into final output.

2.8 Semantic Based Approaches for Event Processing

Although current semantic technologies are being constantly improved dealing with time-invariant domain knowledge, it does not support processing time-sensitive data. Adi et al. (2000) emphasised the significance of expressing the event semantics and relationships among events and other entities, such as the subclass relationships (Adi et al., 2000). They also defined the semantic abstractions and the implied knowledge representation scheme for events and provided a detailed event model that included several semantic properties for events. In recent years, few approaches have emerged for addressing problems from this area, acknowledging

time to be a vital dimension to process knowledge. This section reviews few of these approaches.

2.8.1 Temporal RDF

The Resource Description Framework (RDF), as noted by Klyne and Carroll (2006), has been commonly used to express graph-structured data (Klyne and Carroll, 2006). Gutierrez et al. (2007) introduced time as RDF graphs' new dimension (Gutierrez et al., 2007). They provided semantics for temporal RDF graphs and a temporal query language for RDF which adheres to temporal databases concepts. They are concerned with the evolution of RDF graphs over time, and provide a framework for temporal entailment and querying over changing graphs.

The present study differs from this approach as this study intends to identify temporal complex patterns considering time constraints instead of posing a single query and receiving one response. This study aims to detect situations of interest continuously whenever they occur. Therefore, it is necessary to continuously evaluate patterns in order to process the occurrence of relevant triples and also to detect complex events. SPARQL-ST (Perry et al., 2011) is an extension of the SPARQL language designed for complex temporal and spatial queries (Prud and Seaborne, 2006). The language and a corresponding implementation deal with temporal data as well as the possible reasoning over that data. On the other hand, according to a study by Gutierrez et al. (2007), only when SPARQL-ST queries are invoked, they are evaluated; that is, they are not continuously active (Gutierrez et al., 2007).

The same argumentation can also be applied to other SPARQL approaches such as Temporal SPARQL (Tappolet and Bernstein, 2009), T-SPARQL (Grandi, 2010), and stSPARQL (Koubarakis and Kyzirakos, 2010). Moreover, EP-SPARQL (Anicic et al., 2012) encourages the necessity of a semantic management for streaming data. RDF format is used to represent streaming data so that it be exploited in semantic-web applications, such as semantically

annotated data and reasoning services. For this purpose, they proposed a Time-Annotated RDF model as well as a Time-Annotated SPARQL. However, the authors stated explicitly that continuous queries are a typical requirement of streaming data management systems, but it is not considered in that work.

2.8.2 Stream reasoning approach: Continuous query languages

Continuous SPARQL (C-SPARQL) (Barbieri et al., 2010) is a language for continuous query processing and Stream Reasoning. C-SPARQL provides extensions to the SPARQL language by implementing operations for window and aggregation. The set of currently valid RDF statements, in C-SPARQL, is determined according to a query (such as its window specification), and typical reasoning on that RDF set is conducted as if it were static. Chapter 6 highlights the RDF triples detection in a particular *temporal* order, in order to capture more complex event patterns over RDF streaming data. Moreover, queries in C-SPARQL are classified into a static and dynamic part. A separate RDF triple storage is used to evaluate the static part, whereas the dynamic part is evaluated by a stream processing engine. These two parts in such cases function as “black boxes”, where C-SPARQL is unable to use query pre-processing and optimisations over the unified (static and dynamic) data space. The proposed model in this work is on description logic, which is capable of handling the both parts within a uniform framework.

Streaming Knowledge Bases are reasoning tools used to stream RDF triples and to compute the RDFS closures of an ontology. The tool, for example, is able to detect a triple in a stream that has a subject that represents a certain class’ instance or an instance of any of its subclasses defined in an ontology. The approach is based on TelegraphCQ model, which is an effective DSMS. For accelerating the upstream reasoning, the authors proposed to pre-calculate all inferences in advance and to record them in a database.

The MASSIF (Bonte et al., 2017) platform enables semantic annotation of IoT data and the high-level coordination between semantic IoT services. *Services* are able to indicate their input data on an abstract level since the platform semantically represent the data and is totally data driven. Every *Service* is able to process the received data and share the obtained knowledge with other *Services* using the Semantic Communication Bus. Thus, the development of data-driven workflows that can accomplish complex reasoning chains becomes feasible. Since the *Services* define their input data, they can operate only on a subset of the available data, thus improving the reasoning efficiency compared to other models in this field. The MASSIF platform, however, has failed to provide time aware operators and lacks support for events joining, which are one of the crucial features in the processing of complex events.

ODECP (Taylor and Leidinger, 2015) proposes the use of ontologies for specifying and recognising complex events that arise as selections and correlations, such as temporal correlations, of structured digital messages, usually streamed from multiple sensor networks. Moreover, ontologies are utilised as a basis for defining contextualised complex events of interest that are then translated into selections and temporal combinations of streamed messages. Description logic reasoning is used to support the translation of event descriptions into the native language of a commercial Complex Event Processor (CEP), and also to support their execution under the control of the CEP. However, the ODECP platform does not provide a query language interface which limits its capabilities to the language interface adopted in a given scenario. Chapter 6 provides a more comprehensive comparison of CEP models within an IoT environment and draws the shortfalls that are addressed in order to improve event processing within an IoT environment based on semantic techniques and description logic.

2.9 Discussion

This chapter described the most relevant approaches in the various areas that this work is based upon. With regard to sensor ontologies used with IoT environment, we have studied the existing ontologies and classified the various sensor-based ontologies according to the problems that they address. We have also compared the existing IoT ontologies against their use of Semantic annotation, inference and context model used.

We have also studied various event processing agents in order to understand how event processing is carried upon streaming data and investigated the old static model of processing data versus the new continuous stream processing approach.

Table 2-4 Research Parameters and Associated References

Areas tackled by existing sensor ontologies	
Sensor Discovery (Bermudez-Edo et al., 2016), (Hirmer et al., 2016), (Balaji et al., 2016).	**Data Access and Sharing** (Xue et al., 2015), (Shi et al., 2012).
Sensor Capabilities (Bermudez-Edo et al., 2016), (Xue et al., 2015), (Russomanno et al., 2005).	**Sensor Data Description** (Gyrard et al., 2014b), (Compton et al., 2012), (Nachabe et al., 2015).
Extensibility (Balaji et al., 2016), (Daniele et al., 2015).	
Ontologies compared against	
Semantic Annotation (Kostelnik et al., 2011), (Gray et al., 2011b), (De et al., 2012), (Celdrán et al., 2014), (Kang and Park, 2013), (Indra, 2017), (Calbimonte et al., 2014), (Ali et al., 2015), (Soldatos et al., 2015), (Patkos et al., 2010).	**Inference** (Kostelnik et al., 2011), (Gray et al., 2011b), (De et al., 2012), (Celdrán et al., 2014), (Kang and Park, 2013), (Indra, 2017), (Calbimonte et al., 2014), (Ali et al., 2015), (Soldatos et al., 2015), (Patkos et al., 2010).

Context Model (Kostelnik et al., 2011), (Gray et al., 2011b), (De et al., 2012), (Celdrán et al., 2014), (Kang and Park, 2013), (Indra, 2017), (Calbimonte et al., 2014), (Ali et al., 2015), (Soldatos et al., 2015), (Patkos et al., 2010).	
Event Processing Agents	
Filter (Etzion, 2010), (Luckham, 2011).	**Pattern** (Etzion, 2010), (Luckham, 2011).
Transformation (Etzion, 2010), (Luckham, 2011).	
Continuous Stream Processing	
(Perry et al., 2011), (Tappolet and Bernstein, 2009), (Grandi, 2010), (Koubarakis and Kyzirakos, 2010), (Anicic et al., 2012), (Barbieri et al., 2010), (Bonte et al., 2017), (Taylor and Leidinger, 2015), (Schreier et al., 1991), (Liu et al., 1999), (Golab and Özsu, 2003), (Chen et al., 2000).	

Table 2-4 shows the parameters used to carry out the literature review for the state of the art sensor ontologies, and for the Ontology-based IoT platforms. Section 2.4 provided a comprehensive comparison of existing Ontology-based IoT platform, shown in Table 2-2. Section 2.5 studied the areas tackled by sensor ontologies and showed the limitations in existing ontologies. The table provides a summary of various areas tackled by existing ontologies (Sensor Discovery, Data Access and Sharing, Sensor Capabilities, Sensor Data Description, and Extensibility) and the relevant researchers in this area. Moreover, it provides the parameters used to study the existing ontologies (Semantic Annotation, Inference, and Context Model). The table also shows the continuous stream processing models investigated in this thesis.

Within an IoT environment, the devices involved tend to be highly heterogeneous, in terms of nodes' profiles and models, communication protocols, the data collected (for example, coding schemes, sensing ranges, formats and metadata) and applications. Since different nodes may be made by different manufacturers, they may have various processing functionalities. The resources they require may depend on their role – sensor, actuator, software or tag. Moreover, the rate and type of data that is required may also differ between applications.

These issues concerning heterogeneity and interoperability hinder the functioning of the IoT. Indeed, they may, in fact, make it impossible for different sorts of sensor to interact, whether at the level of application, monitoring or management, for example, by reusing and sharing data across various applications.

Resolving these problems depends in particular on developing an explicit semantic that is shared by all the terminologies used. It is thus necessary to define a data model specifically designed for the IoT. In this thesis we address the heterogeneity problem within an IoT infrastructure, this involves solving data variety (produced from heterogeneous sources), and is able to combine data with background knowledge, that deducts related information and operates temporal reasoning combining data streams from sensors and events. Temporal extensions of deductive reasoning extend the ontological language with time relations and, thus, easily diverge into intractability. Semantic Complex Event Processing is limited to a semantic description of events and does not focus on the processing. This thesis proposes an ontology based event-processing approach that operates the event processing over high-level concepts deduced through deductive reasoning, but without including time relations at the ontological level. Chapter 3 presents a review study of the methodologies used in ontology development processes.

CHAPTER 3: REVIEW OF ONTOLOGY DEVELOPMENT METHODOLOGIES

3.1 Introduction: Why Ontology?

The Internet of Things (IoT) has grown rapidly over recent years as a result of the ongoing development of wireless technology and the increasing miniaturisation of hardware. As the usage of the IoT continues to develop, it is coming to affect every area of human life, including energy management, healthcare, home automation and even military activities. The essential idea of the IoT is that everyday objects are interconnected through networks so as to work together in collecting, processing and transferring information (Atzori et al., 2010).

Any IoT environment contains a range of characteristics and constraints. Among the most serious constraints stems from the perpetual changes in the network, which contains stationary and moving nodes formed by millions of different sensors and actuators that are distributed at random across the sensing field. Since such networks are multi-scalar and dynamic, failure is common.

Within an IoT environment, the devices involved tend to be highly heterogeneous, in terms of nodes' profiles and models, communication protocols, the data collected (for example, coding schemes, sensing ranges, formats and metadata) and applications. Since different nodes may be made by different manufacturers, they may have various processing functionalities. The resources they require may depend on their role – sensor, actuator, software or tag. Moreover, the rate and type of data that is required may also differ between applications.

These issues concerning heterogeneity and interoperability hinder the functioning of the IoT. Indeed, they may, in fact, make it impossible for different sorts of sensor to interact, whether at the level of application, monitoring or management, for example, by reusing and sharing data across various applications.

Resolving these problems depends in particular on developing an explicit semantic that is shared by all the terminologies used. It is thus necessary to define a data model specifically designed for the IoT.

Ontologies have been defined as "tools for specifying the semantics of a terminology system in a well-defined and unambiguous manner" (Bittner et al., 2005). As such, ontologies may be deployed in formalising a specific data model that suits an IoT environment. This could be achieved through the development of a common vocabulary, rooted in unambiguous semantics, for sharing different forms of data – including sensed, monitoring, control, alarm data – between the various services, applications and components that make up an IoT environment.

This chapter presents the existing methodologies used for developing ontologies and the methodology used in our development. Chapter Four presents the devised ontology specifically for the IoT. This semantic model, which is formalised and pre-validated in Chapter 4, has been designed to deal with all types of sensors and actuators, as well as the data they generate. The proposed model thus addresses IoT's crucial problems of interoperability and heterogeneity.

Section 3.2, provides a summary about methodologies used for the development of ontologies within an IoT paradigm.

Section 3.3 presents the development process used for developing the IoT-Ont ontology in Chapter 4. Sections 3.4 and 3.5 respectively represent the conceptual and functional requirements for developing IoT-Ont.

3.2 Methodologies in Ontology development

This section presents a brief summary of the most popular methodologies used for developing ontologies, Section 3.2.1 introduces how these methodologies accomplish their lifecycle

development. Section 3.2.2 presents the state-of-the-art on ontology requirements specification and finally Section 3.2.3 presents the importance of resource reuse in developing ontologies.

3.2.1 Lifecycles of Principal Methodologies for Ontology Development

This section introduces a brief description of the four most adopted methodologies for devising ontologies. It presents the ontologies in a chronological order starting with, METHONTOLOGY methodology developed in 1998 (Blázquez et al., 1998), followed by On-To-Knowledge developed in 2001 (Staab et al., 2001), the DILIGENT methodology developed in 2004 (Pinto et al., 2004) and finally the NeOn methodology developed in 2012 (Suárez-Figueroa et al., 2012).

3.2.1.1 METHONTOLOGY

The METHONTOLOGY methodology (Blázquez et al., 1998), was devised by the Ontology Engineering Group at the Technical University of Madrid. METHONTOLOGY facilitates ontology's construction at the knowledge level.

This methodology defines a set of activities to be accomplished in order to build ontologies; this set of activities is defined as the identification process of ontology development. METHONTOLGIY's life cycle is based on evolutionary prototyping, and on various techniques to accomplish each of these activities during the development process, support activities, and management.

In order to provide technical support for METHONTOLOGY, the same engineering group at the Technical University of Madrid developed two workbenches ODE (Blázquez et al., 1998) and WebODE (Arpírez et al., 2003) that allow collaborative development of ontologies and includes various other features to provide scalable architecture for the building ontology development tools and ontology-based applications. Various ontology tool suites and tools

support building ontologies based on METHONTOLOGY, for example, Protégé and the NeOn Toolkit.

The Foundation for Intelligent Physical Agents[9] (FIPA) has adopted the use of this methodology for ontology development, which promotes inter-operability across agent-based applications.

The ontology development process as a set of activities that are performed when devising ontologies (Fernández-López et al., 1997, Blázquez et al., 1998) was established on the framework of METHONTOLOGY for ontology development. The development process proposal was built on accordance with the IEEE standard for software development (IEEE, 1996). The METHONTOLGY development process is characterised in three phases: Management, development-oriented and support:

- **Ontology management activities:** This phase includes activities to initiate, control and monitor an ontology project during its life cycle.
- **Ontology technical development activities,** which range from specification to maintenance. As shown in Figure 3-1.
- **Ontology support activities:** This phase includes activities necessary to affirm the successful realisation of the ontology project.

[9] http://www.fipa.org/

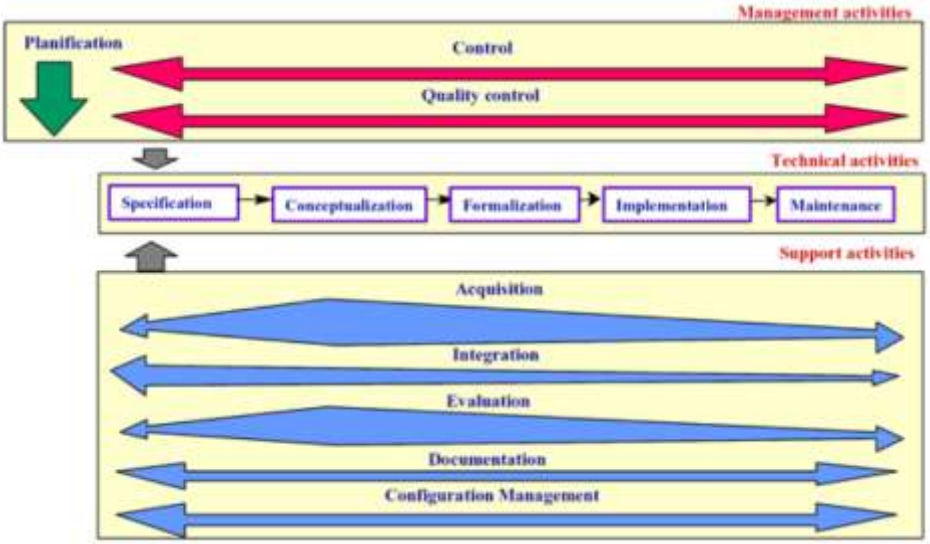

Figure 3-1 Lifecycle of ontology development process based on METHONTOLOGY (Fernández-López et al., 1997)

For ***requirements specification***, METHONTOLOGY recommends the usage of competency questions and intermediate representations in order to describe the requirements that should be fulfilled by an ontology. However, no detailed guidelines are provided for accomplishing this activity. The initial prototype is specified within the specification activity, which is then followed by the conceptualisation activity where the ontological conceptual model is developed.

For ontology reuse METHONTOLOGY comprises of a list of activities to be accomplished, however, again no detailed guidelines are provided regarding these activities. Moreover, there is no consideration of various granularity levels during the ontology reuse (as for example, ontology statements). Concerning reusing and reengineering, the methodology does not consider the reuse of design patterns nor the reuse and re-engineering of non-ontological resources.

3.2.1.2 On-To-Knowledge

The On-To-Knowledge methodology project (Staab et al., 2001) aims to apply ontologies to information available electronically in order to improving knowledge management quality in large distributed organisations. Various partners were involved in this project in order to develop the methodology and multiple tools for intelligent access of big volumes textual and semi-structured information sources in Internet-based environments. The methodology proposes developing ontologies considering their future utilisation in different knowledge management applications. Thus, the developed ontologies using the On-To-Knowledge methodology are particularly dependant on the application.

An important feature this methodology proposes is ontology learning with the aim of reducing efforts made to devise an ontology. But unlike the METHONTOLOGY methodology, this methodology does not consider collaboration development of ontologies.

Goals identification is included in this methodology, this is achieved on analysis of usage scenarios by knowledge management tools (Staab et al., 2001).

The proposed processes by On-To-Knowledge are summarised below:

1) Feasibility study: The feasibility study according to this methodology is concerned of the application as a whole, thus, it should be accomplished before devising ontologies. This is based on the feasibility study presented in the CommonKADS methodology (Schreiber and Akkermans, 1999).
2) Kick-off: This process concludes with the requirements specification document for ontology development. These requirements describe the following: the ontology's domain and goal; the design guidelines (for example, the naming conventions); knowledge sources (for instance, magasines, books, interviews); use cases and possible users; and applications supported by the ontology. The methodology proposes the usage

of competency questions (CQs) (Grüninger and Fox, 1995b) in order to deduce the *requirements specification*; however, detailed guidelines for accomplishing this activity are not provided by this methodology. CQs can be very useful to develop the ontology requirements specification. The purpose of the requirements specification document is to facilitate the job of the ontology developer in order to determine whether to include or exclude concepts in an ontology. Actually, the specification document is used to develop a draft version, which contains the most influential elements for future development. The first draft is referred to as a "baseline ontology". This draft identifies the main concepts and the relations on an informal level. During the kick-off process, engineers should search for possible ***reusable ontologies*** developed earlier. Although On-To-Knowledge refers to the identification of potentially reusable ontologies, there are no guidelines to identifying these ontologies. In addition, the methodology does not provide guidelines for reusing and re-engineering of non-ontological resources, nor for the reuse of available design patterns.

3) Refinement. The aim of this process is producing a more mature "target ontology" which is application-oriented based on the specification document provided within the kick-off process. The refinement process comprises of two activities:

- Activity 1: Knowledge gathering process to research the requirements with domain experts.
- Activity 2: Formalisation, where the ontology is implemented using an ontology language

It is essential to keep in mind that the developed ontologies using this methodology are application-dependant. An incremental and cyclic ontology life cycle is adopted by the On-To-Knowledge methodology, which is built on the evolutionary prototyping life cycle model

(Gómez-Pérez et al., 2003). The On-To-Knowledge life cycle is demonstrated in Figure 3-2 (Staab et al., 2001).

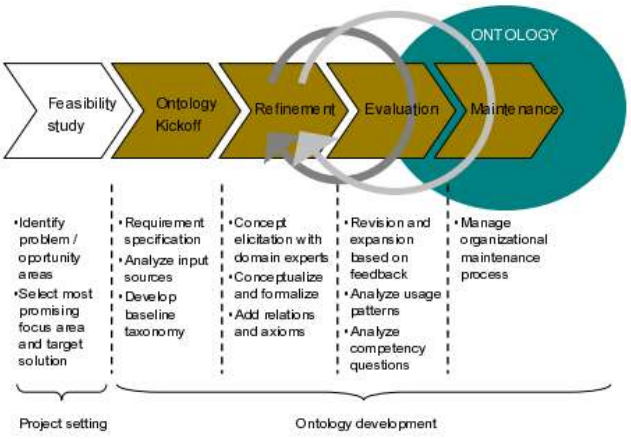

Figure 3-2 the On-To-Knowledge lifecycle (Staab et al., 2001)

3.2.1.3 DILIGENT

This methodology (Pinto et al., 2004) was proposed and developed by the AIFB Institute at Karlsruhe University with the collaboration of the Tecnico Lisboa at the University of Lisbon. This methodology was developed in order to support domain experts to develop ontologies within a distributed setting. The DILIGENT methodology focuses on collaborative development of ontologies; it is based on the idea of tracking change of arguments.

The development process within this methodology comprises of five phases:

1) Build: The goal of this phase is quickly creating a draft version of the ontology to allow the stakeholders use the ontology as soon as possible (Engler et al., 2006). The ontology initial version is created collaboratively by domain experts, knowledge engineers, users and ontology developers. The team involved in building the draft version should be small to facilitate this process and to reach a consensus of this version. Unlike the previous methodologies, the DILIGENT methodology does not require an ontology requirements specification nor does it emphasise on the importance of reusing available

resources. Moreover, completeness of the first draft ontology concerning the domain is not accomplished at this phase.

2) Local adaptation: The next phase involves users adapting the ontology to their own requirements, while the ontology is in use, for example, to organise knowledge (Engler et al., 2006). Once the shared ontology is made available, users can begin to use it and locally adapting it for their own purposes. Normally, the local ontologies develop in a way similar to the folder hierarchies in a file system; this is due to new organisations demands and new business requirements or user changes.
3) Analysis: During the analytical phase the ontology control board assesses the changes that the stakeholders have suggested (Engler et al., 2006). The input of the users provides the necessary arguments to highlight requests for change, the control board, however, examines the local ontologies and the requests for change and attempts to detect similarities in users' ontologies. The objective of this phase is to develop a core and shared ontology because it will quickly become bigger otherwise, and become completely un-manageable.
4) Revision: The ontology is then revised by the board and decisions are taken to what changes should be applied to the ontology (Engler et al., 2006). In order to prevent further divergence between the local ontologies and the shared ontology, the board should review the shared ontology frequently.
5) Local update: This is the last step, at this stage based on the revised ontology version, the stakeholders update their local ontologies (Engler et al., 2006).

The ontology lifecycle proposed by the DILIGENT methodology is based on the evolutionary prototyping life cycle model, illustrated in Figure 3-3. (Pinto et al., 2004).

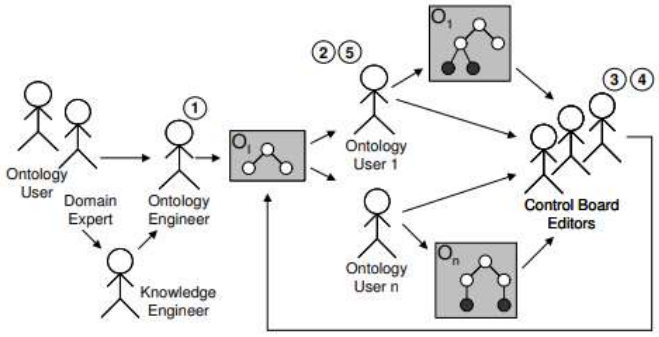

Figure 3-3 DILIGENT Ontology Life Cycle (Pinto et al., 2004)

3.2.1.4 NeOn Methodology

Unlike the alternative approaches to ontological engineering which provide a methodological guide, the NeOn approach does not specify a strict workflow but rather proposes a number of different ways in which ontologies can be developed. The methodology proposes nine scenarios to cover the most common situations. For example, if available ontologies need to be re-engineered, aligned, modularised, localised to support different languages and cultures, and integrated with ontology design patterns and non-ontological resources, such as thesauri. (Suárez-Figueroa et al., 2012)

Morever, this methodology framework presents (a) a glossary of activities and processes involved in ontologies' development, (b) two lifecycle models, and (c) a set of methodological guidelines for different activities and processes, which are described (a) functionally, in terms of objectives, inputs, outputs, and relevant limitations; (b) procedurally, by means of workflow specifications; and (c) empirically, through a set of illustrative examples (Suárez-Figueroa et al., 2012).

Figure 3-4 presents the various scenarios for developing ontologies according to the NeOn methodology (Suárez-Figueroa et al., 2012).

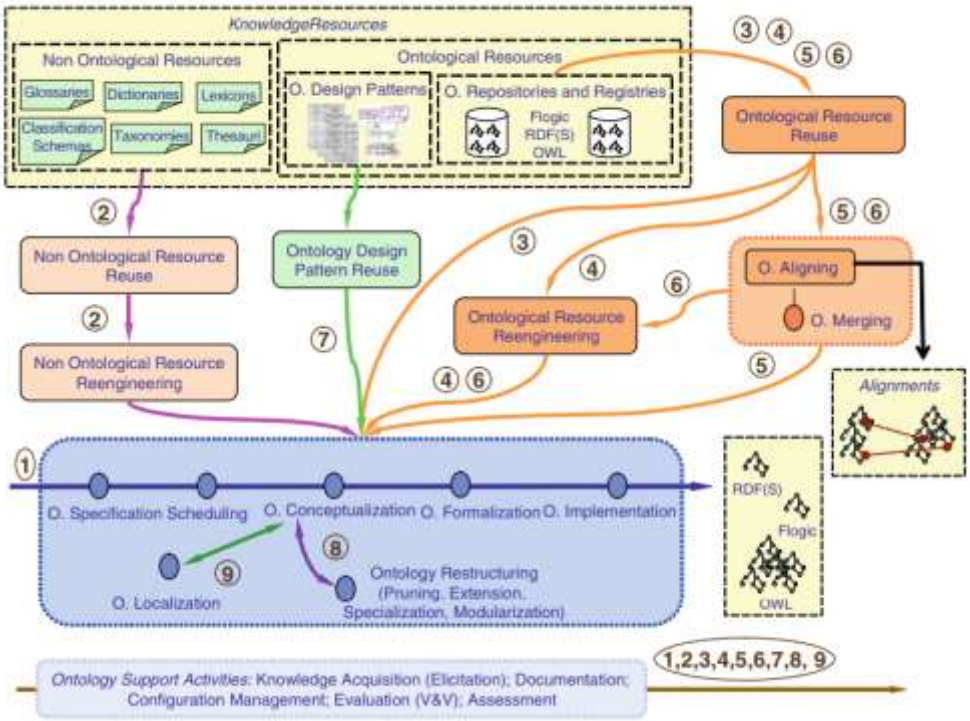

Figure 3-4 Scenarios for building ontologies. (Suárez-Figueroa et al., 2012)

- Scenario 1: From gathering specification to implementing. The ontology network is implemented from scratch, i.e. without the reuse of available resources (Suárez-Figueroa et al., 2012).
- Scenario 2: Reusing and re-engineering non-ontological resources. This scenario includes the situation where ontology engineers need to assess non-ontological resources to determine which of them could be reused, according to the requirements the ontology should fulfil, to build the ontology network. The scenario also includes the task of re-engineering the chosen resources into ontologies (Suárez-Figueroa et al., 2012).

- Scenario 3: Reusing ontological resources. Here, the developers of the ontology reuse ontological resources (the whole ontologies, modules of an ontology, and/or ontology statements). (Suárez-Figueroa et al., 2012)
- Scenario 4: Reusing and re-engineering ontological resources. Here, the developers of the ontology both reuse and re-engineer ontological resources. (Suárez-Figueroa et al., 2012)
- Scenario 5: Reusing and merging ontological resources. This scenario only occurs in those situations where multiple ontological resources are selected for reuse, for the same domain, and where ontology developers aim to develop a new ontological resource from two or more ontological resources. (Suárez-Figueroa et al., 2012)
- Scenario 6: Reusing, merging, and re-engineering ontological resources. This scenario is similar to Scenario 5; however, here developers decide not to use the set of merged resources as it is, but to re-engineer it. (Suárez-Figueroa et al., 2012)
- Scenario 7: Reusing ontology design patterns (ODPs). Ontology developers access ODPs repositories to reuse them. (Suárez-Figueroa et al., 2012)
- Scenario 8: Restructuring ontological resources. Ontology developers restructure (modularising, pruning, extending, and/or specializing) ontological resources to be integrated in the ontology network being built. (Suárez-Figueroa et al., 2012)
- Scenario 9: Localising ontological resources. Ontology developers adapt an ontology to other languages and cultural groups, and therefore produce a multilingual ontology. (Suárez-Figueroa et al., 2012)

3.2.2 Ontology Requirements Specification: Methods and Techniques

This section presents a brief summary of the available techniques methods for ontology requirements specification according to the state-of-the-art methodologies. Ontology

Requirements Specification as defined earlier is the activity to collect the requirements to be met by the ontology (Suárez-Figueroa, 2010).

The methodology proposed by Grüninger and Fox (Grüninger and Fox, 1995a), the On-To-Knowledge methodology (Staab et al., 2001), and the methodology proposed by Uschold (Uschold, 1996) all propose the following basic steps to obtain the requirements specification document of an ontology:

- To identify the purpose of the ontology to be developed.
- To identify the intended users and uses of the ontology to be developed.
- To identify the set of ontology requirements to be met by the ontology when it is formally implemented.

Various techniques for collecting requirements can be applied. Brainstorming, joint application development (JAD) (Pressman and Maxim, 2015), scenario exploitation and use cases using templates, interviews with domain experts and users, and competency questions are examples of these techniques.

Most of the methods (Grüninger and Fox, 1995a, Staab et al., 2001) (De Hoog, 1998) (Hristozova and Sterling, 2003) (Uschold, 1996) and guides (Noy and McGuinness, 2001) for ontology development recommend the identification of **competency questions** as a technique to identify the ontology requirements. Competency questions (CQs) firstly proposed by (Grüninger and Fox, 1995b), were defined as the questions that the ontology to be implemented should be capable of answering.

The rest of this section shows how the various guides and methodologies mentioned above propose to carry out the activity of requirements specification for ontology development utilising competency questions.

The methodology of Grüninger and Fox's (Grüninger and Fox, 1995a) was influenced by the advancements of first-order logic knowledge-based systems. This methodology suggests intuitively identifying the main motivating scenarios, i.e. potential applications where the ontology could be used. These scenarios outline a set of the ontology requirements to be met by the ontology after formal implementation. Industrial partners may present the scenarios, these scenarios could be issues or problems experienced in their companies. The motivating scenarios often have the form of story problems or examples that exiting ontologies do not adequately address. A motivating scenario offers a range of intuitive alternatives to solve the scenario problems. These solutions give an initial idea of the intended informal semantics of the objects and relations to be included in the ontology later on.

Provided with the set of informal scenarios, the developer can identify a set of informal competency questions to define the ontology's scope. Informal competency questions are those expressed in natural language that the ontology must answer when the ontology is expressed in a formal language. Both, the questions and their answers are used to extract the main concepts and their ontological properties, formal axioms, and relations within the ontology. Competency questions and their answers constitute the of requirement specification against which the ontology could be assessed.

The On-To-Knowledge methodology (Staab et al., 2001) asserts the importance of competency questions and their usefulness in developing the requirements specification document. The requirement specification document should allow the ontology developer to determine whether to include or exclude concepts in the ontology and about their hierarchical structure.

The methodology proposed by Uschold **(Uschold, 1996)** suggests identifying (a) the ontology's purpose and, in particular, the identification and definition of the potential users, (b) the uses and application of the ontology, and (c) (fairly general) motivating scenarios and

competency questions, and to produce a user requirements document for the target software system. The methodology then suggests determine how formal the ontology should be. This decision is largely determined by the purpose of the ontology and its users. Eventually, this methodology suggests to identify the ontology's scope by means of (a) establishing the comprehensive motivating scenarios that arise in the applications, which were also proposed by Grüninger and Fox's (Grüninger and Fox, 1995a) or (b) employing brainstorming to carry out a more comprehensive and precise scoping task.

The EXPLODE methodology incorporates ideas from the methodology of extreme Programming methodlogy. Due to its focus on immediate feedback and evaluation, this methodology is especially suited for dynamic and open environments. It proposes fetching the system's requirements and identifying the competency questions (De Hoog, 1998, Hristozova and Sterling, 2003).

The "Ontology Development 101" guide proposes to determine the scope and the domain of the ontology by answering a number of fundamental questions ("What is the domain that the ontology will cover?", "What will be the uses of the ontology?", "Who will use and maintain the ontology?", etc.). The guide also suggests identifying the ontology competency questions, however it does not provide more details regarding its identification (Noy and McGuinness, 2001).

3.2.3 Ontological Resource Reuse: Methods and Techniques

Ontology Resource Reuse is the process of using available ontological resources (ontologies, modules, statements, or ontology design patterns) to solve different problems (e.g., the development of different ontology-based applications, the ontology alignment activity (as background knowledge)) (Suárez-Figueroa, 2010). This section presents a short overview of

the available techniques and methods for reusing ontological resources according to the state-of-the-art methodologies.

METHONTOLOGY (Gómez-Pérez et al., 2003) proposes the following activities to reuse ontologies in a particular domain:

- Searching for possible ontologies to be reused.
- Content and granularity inspection of the candidate ontologies.
- Selecting the ontologies to be reused.
- Evaluating the selected ontologies from knowledge representation point of view.

Uschold et al. (Uschold, 1996) identify the primary involved tasks in reusing available ontologies as follows:

- The comprehension of the ontology and finding a core to reuse.
- The translation of the ontology.
- The specification and refinement of the ontology into executable code. The aim here is to define refinements of the specifications produced in the above steps and producing the executable code.
- The verification of the refined ontology, this shall ensure that the executable code is precise to the initial specification.
- The integration of the ontology with the application.

Pinto and Martins (Pinto and Martins, 2001) introduces the following activities for reusing ontologies as part of the integration process:

- Search and select the potential ontologies. The ontologies to be analysed, and possibly reused, are selected from ontologies available in libraries that meet a number of requirements.
- Evaluation with an integration approach. The potential ontologies must be assessed from an integration point of view. The authors have defined a range of criteria that domain experts should consider when analysing an ontology for integration. The ontology should be evaluated with particular attention to the missing knowledge; These criteria for evaluation demonstrate the weaknesses and strengths of the potential ontology from domain experts' perspectives.
- Assessment with an integration approach. The potential ontologies should be assessed from an integration point of view.
- Selecting the appropriate ontology. Following the analysis of different potential ontologies, and given the fact that they may not perfectly match the ontology needed, another choice should take place. This choice must be taken amongst the potential ontologies that have complied with the strict requirements.
- Integrating the chosen ontology.
- Assessing and evaluating the final ontology.

Paslaru and Mochol (Bontas and Mochol, 2005) propose an incremental reuse process that comprises of :

- Taking into account the vocabulary of the potential ontologies (concepts, relations, and axioms) and deriving a common vocabulary based on the natural language in which the ontological primitives have been originally termed.

- Merging the vocabularies of potential ontologies, the generation of distinct ontological primitives lists based on the formality levels of the models considered, and dropping duplicates in order to prevent additional computations.
- Computation of syntactic resemblance between concept names coming from various resource ontologies (Cohen et al., 2003).
- Enhance the accuracy of the resemblance computation, creating a container of terms in the source vocabularies for each concept name.
- Computing the concepts ranks by taking into account the frequency of occurrences (when names of a concept occur in several resource ontologies they rank higher), the priority of the source (measure of relevance of the respective source to the targeted application domain) and the requirements of the application.
- Identification of relevant concepts. The user determines the adequate relationships that could be integrated within the ontology incrementally until a certain complexity level is attained.

3.3 IoT-ONT DEVELOPMENT

IoT is increasingly growing both in the number of devices and the volume of data produced, objects or smart things are now part of our daily lives, these things are deployed within dynamic systems, characterised mainly by mobility constraints, fluctuating network connectivity controlled by various parameters (e.g. duty cycle or battery level, etc.). Dynamic knowledge representations are essential in order to capture knowledge that describes these ever-growing environments. These knowledge representations must be adaptable over time to address the changing state of these environments.

Chapter 4 presents IoT-Ont, which is an IoT ontology based on a modular design that supports, reuse, one of the most sought-after functionalities in developing ontologies. IoT-Ont introduces

a vocabulary that describes the connected things and the main relations that govern their interaction within an IoT environment.

A number of components are involved in any application that uses IoT technology, whether a cloud-based application which functions through using sensors, or a standalone mobile app. These components function through interaction with a range of sensors. These sensors, which are connected to a platform, may either be static or mobile if there is a need to gather data across a wider area.

Requirements for ontology development are usually divided in two types: Conceptual (non-functional requirements) and functional requirements: (Suárez-Figueroa, 2010).

1) Conceptual (non-functional requirements) include the main concepts that must exist in the ontology. Section 3.4 presents a detailed perspective of gathering the conceptual requirements, followed by a comparison between existing IoT-Ontologies against these requirements.
2) Functional requirements include the design guidelines for ontologies and semantic best practice on the web, which is presented in a domain-agnostic manner. Section 3.5 presents the functional requirements, followed by a comparison between existing IoT ontologies against these requirements.

3.4 Conceptual Requirements: Key Concepts in Developing an IoT Ontology

To design an optimal IoT ontology, it is necessary to describe both horizontal concepts (i.e., those shared by all applications in the IoT) and vertical concepts (i.e. those that are distinct to specific applications), as shown in Figure 3-6.

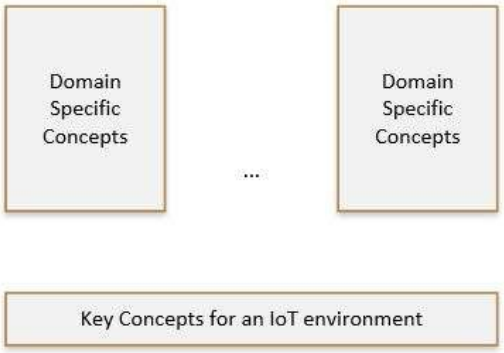

Figure 3-5 Horizontal vs. Vertical concepts

Nevertheless, for the purposes of the current study, we restrict our investigation to the horizontal concepts that are shared by all IoT domains. Identifying these core concepts requires defining competency questions, that is, queries which experts pose to the knowledge base (Yan et al., 2015). Various authors, including Bermudez-Edo et al. (Bermudez-Edo et al., 2016), Haller et al. (Haller et al., 2013) and Wang et al. (Wang et al., 2013) have located some of these competency questions. They have identified various core concepts that are necessary in the IoT field, including Service, Resource, Device, User and Augmented Entity. However, some of these studies (Bermudez-Edo et al., 2016) (Haller et al., 2013) fail to take into account all of the core concepts that different applications may require to generate complex information in IoT environments. One widespread approach for describing the basic features of a particular event or situation is the 4W1H methodology (Niitsuma et al., 2009, Zhang et al., 2014), which is deployed in this work to identify the core concepts of IoT-Ont ontology. To identify the necessary competency questions, it is necessary to define five variables: four 'W's (Who, What, Where, When) and one 'H' (How). These five variables and their respective corresponding competency questions are presented below:

- Who will present the necessary information for designing the IoT application? Answering this question involves concepts that can locate the sources of data for developing an application for the IoT environment. These sources lie in sensors

implanted in a platform. Therefore, to provide an answer to the 'who' question, the IoT ontology must hold concepts for sensors and platforms.

- Under what conditions must the source gather data? Are the particular circumstances in which it is appropriate to collect the data? To answer this question, the IoT ontology must include concepts that can define the data source's context. The context might include the performed action or the mobility of a particular sensor. It is also necessary to determine whether the measurement involved a human intervention or whether it was done automatically by the device.
- Where is the data source? The data source's location may be defined through landmarks, geo-coordinates or names of buildings. If developers are to be able to identify the source of the data, it is, therefore, necessary for the IoT ontology to include where these sources might be located.
- When should the data be collected? It is necessary to ask whether the sensor should be collecting the data over a particular time period, at regular intervals, or in designated timestamps. Within the IoT technology, there should be concepts to allow for a range of formats for defining when data collection is to occur.
- How, after the sensors have gathered the relevant data, should it be disseminated to the developers in order for them to construct the application? Concepts are needed to maintain services that allow developers to access sensor data.

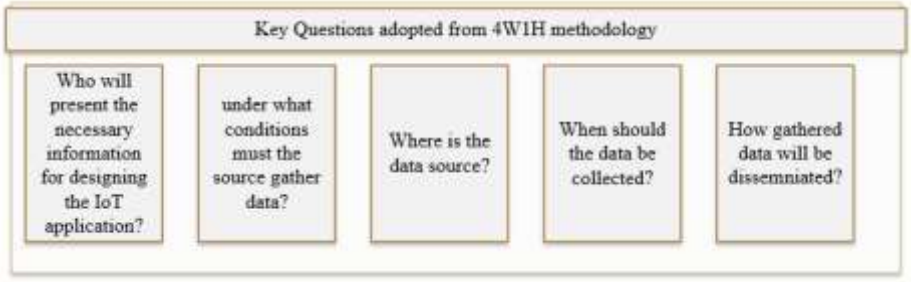

Figure 3-6 The CQs based on the 4W1H technique.

Based on these competency questions, it is possible to identify which core concepts are needed to generate a comprehensive IoT ontology. These concepts are as follows:

1. Sensor data gathered from a range of heterogeneous sensors form the basis of any IoT application. A sensor is a source that generates a value to capture the quality of a particular phenomenon (Compton et al., 2009). In this context, sensor data signifies not only the raw data generated by the sensor but also the metadata describing the sensor itself.

2. A power supply is necessary for the functioning of these sensors, which may receive their energy from a battery or another source. In IoT-Ont this is limited to the current state of the device whether it is on or off, this can be expanded in future work.

3. A realistic testbed, on a large scale, is necessary if these **platforms** are to offer extensive IoT solutions (Gluhak et al., 2011). The testbed should be capable of catering to the various functionalities required by different sorts of sensors, which depend on the sort of phenomenon that they are detecting. It is also necessary for the testbed to have mechanisms which allow the sensor data to be transferred to the applications or actuators. In some cases, the sensor data that is captured is also stored internally within the testbed; this storage, which is done in the proprietary format, enables local data analysis to be conducted.

4. Once the testbed infrastructure has been devised, it is necessary for an identifiable **service** to access the data, whether in raw or processed form. This service should have the capacity to

receive various sorts of sensor data in order to gain optimal contextual information (that is, assimilation) and to remove, or filter, the redundant sensor data generated by a collection of different data sources.

5. Moreover, the majority of sensor data is specific both to **time** and **location**. Therefore, it is necessary to use information regarding time and location so as to generate a common context. This context includes data about a particular location and its environmental conditions, as well as the people, objects, devices and software agents present there (Chen et al., 2003). All this data is necessary for designing services within IoT environments (Flury et al., 2004).

Table 3-1 Coverage of Conceptual Requirements in existing state-of-the-art ontologies

	IoT-Thing	Sensor	Actuator	Service	Location	Time	Status(Power)	Event
SAREF (Daniele et al., 2015)	No	Yes	No	No	Yes	Yes	No	No
IoT-Lite (Bermudez-Edo et al., 2016)	No	Yes	Yes	No	Yes	No	No	No
Spitfire (Nagowah et al., 2018)	No	Yes	Yes	No	Yes	Yes	Yes	No
SSN (Compton et al., 2012)	No	Yes	Yes	No	No	No	No	No
oneM2M (Yun et al., 2019)	Yes	Yes	Yes	Yes	No	No	No	No
IoT-Ont	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

As seen in Table 3-1, all ontologies cover only some of the necessary requirements and thus they are incapable of providing a comprehensive solution to the problems of heterogeneity and interoperability in IoT technologies. Moreover, it would not be possible to combine existing ontologies in the interests of developing a comprehensive framework for reasoning about sensor measurements and data. This is because the resultant model would lack extended semantics designed specifically for the task. IoT-Ont addresses the limitations of previous models, by using a modular semantic data model where existing modules from existing ontologies are reused to develop the ontology.

3.5 Functional Requirements: Best Practices in Developing Ontologies

After gathering the conceptual requirements of an IoT ontology, this section presents the functional requirements with regard to the best practices in developing ontologies.

Reusability is one of the key problems facing the development of an ontology for a domain as large as IoT. Work conducted to define an ontology for a specific project is unlikely to be relevant to other projects. There is a range of overlapping approaches that can be applied to resolve this crucial issue:

Modular Design

If ontologies are designed in separate modules ('modularisation'), it is easier to extend or reuse them (d'Aquin, 2012). Since IoT applications are connected with multiple domains, combining all these domains into a single ontology is complex and would result in a very complicated ontology. However, modular ontologies allow an approach that can be scaled up or down, as they can be combined or separated as required.

Reusability

Reuse of available resources: avoids redefinitions, and prevents from having to align a posteriori the redefined concepts to the existing sources for interoperability. It is a primary requirement of interoperability, which is a key issue in heterogeneous systems.

By reusing existing sources, it is possible to avoid having to redefine concepts. It also means that it is not necessary to logically align new concepts with the pre-existing sources. Reusing existing sources is thus crucial for ensuring interoperability, which often presents problems within heterogeneous systems such as IoT.

If ontologies are designed in a way that conforms to Ontology Design Patterns (ODP) (Gangemi, 2005), they are more likely to be reused and can be more easily aligned (Scharffe et al., 2008). ODP's records the efforts made at modelling ontologies, using them is essential to make use of previous work.

Reusability can be ensured by aligning with upper-level ontologies, which define abstract concepts horizontally. Since upper-level ontologies specify a wide range of different domain-specific ontologies, they are of critical importance to IoT, which is itself a very wide domain.

Expressiveness

It is necessary to choose the correct formalism level. Semantic descriptions of data and devices should allow inferences and reasoning. When applied to data, semantics would give rise to context-awareness (Henson et al., 2012); in the case of devices, it would enable device's self-configuration and facilitates discovery of Things (Chatzigiannakis et al., 2012); in the case of services, it facilitates automatic composition (Han et al., 2012). However, for specific applications, the chosen semantic model should accomplish inferences within a given timeframe. In reality, this renders an OWL-full model unfeasible. OWL-DL has been proposed by all the IoT ontologies surveyed, and is the recommended one by the Semantic Web Consortium.

Table 3-2 Non-Functional requirements coverage in existing IoT ontologies

	Modular	**Reuse existing ontologies**	**ODP Based**	**Allignment with upper ontologies**
SAREF (Daniele et al., 2015)	Yes	No	No	No
IoT-Lite (Bermudez-Edo et al., 2016)	No	Limited	No	No

Spitfire (Nagowah et al., 2018)	No	No	No	Yes
SSN (Compton et al., 2012)	Yes	Yes	Yes	Yes
oneM2M (Yun et al., 2019)	No	No	No	No
IoT-Ont	Yes	Yes	Yes	Yes

Table 3-2 illustrates how the semantic web best practices concerning reusability are often not adhered to in developing IoT ontologies: External ontologies are never reused, except the reuse of SSN and a limited usage within the IoT-Lite ontology. The rest of the researched IoT ontologies redefine concepts used in other ontologies. SAREF is a popular ontology in IoT, however, the ontology does not use the concepts present in multiple previous ontologies, it redefines these concepts and is not aligned to upper ontologies, however it does propose alignments separately in a textual document. Design patterns have only been used in ontologies importing SSN. Upper ontologies used are DUL22 (especially used by SSN) and SWEET23 (for SA). The limited reuse of modules and previously developed ontologies in this field is a limitation to unifying them and to the development of a generic model for IoT. SSN is the only ontology that adheres to the semantic web good practices, being a modular ontology that facilitates the adoption of its' module to develop a generic or a domain specific IoT ontologies.

Chapter 4 presents the main modules and the main concepts of IoT-Ont respectively.

3.6 Discussion

The identification of functional and non-functional requirements is one of the key activities in the development of ontologies. This chapter has presented the ontology requirements

specification activity employed by state-of-the-art methodologies and presented the methodological guidelines for ontology requirements specification based on CQs.

Since the technique proposed by by Grüninger and Fox (Grüninger and Fox, 1995a) on the specification of ontology requirements based on CQs, there have not been substantial contributions in this field. The purpose of specifying ontology requirements is to explain the reason behind developing the ontology, the intended usage of the ontology, the potential end-users, and the requirements the ontology needs to meet. In this thesis, the competency questions technique will be deployed in order to specify the ontology requirements as proposed in (Grüninger and Fox, 1995a).

The field of ontology engineering lacks tools and methods that can help guide ontology developers in the planning and scheduling of their ontology development projects. The chapter aimed at reviewing the current methodologies and techniques used for ontology development projects.

For many knowledge-intensive applications, ontologies play an important role. The process of developing ontologies from scratch as proposed in METHONTOLOGY (Gómez-Pérez et al., 2003) and in On-To-Knowledge (Staab et al., 2001) involves high consumption of time and cost. One technique to reduce the time and costs associated with ontology development projects is to reuse the ontological resources available. The reuse of existent and (well-developed) ontological resources enables the spread of best practices and increases the overall quality of ontological models.

To summarise, there is no standard methodology for the process of ontology development, there exists a set of guidelines and techniques in this field. The development of IoT-Ont is based on and compliant with the 'NeOn' methodology, which is presented in (Suárez-Figueroa et al., 2012), The NeOn methodology is one of the most cited methodologies for ontology

development and it has strong emphasis on gathering requirements (Suárez-Figueroa et al., 2012).

The NeOn methodology have also systematised the reuse of ontological resources. The methodology provides thorough and descriptive methodological guidelines. A precise method to reuse ontological resources at various granularity levels is established : (a) through either the reuse of the whole ontological resources, and (b) or through the reuse of specific ontological statements (Suárez-Figueroa et al., 2012).

Figure 3-5 summarises the development process for IoT-Ont based on the NeOn methodology. The NeOn process starts by identifying the requirements, this is illustrated in details in the next chapter in Sections 4.1 and 4.2.

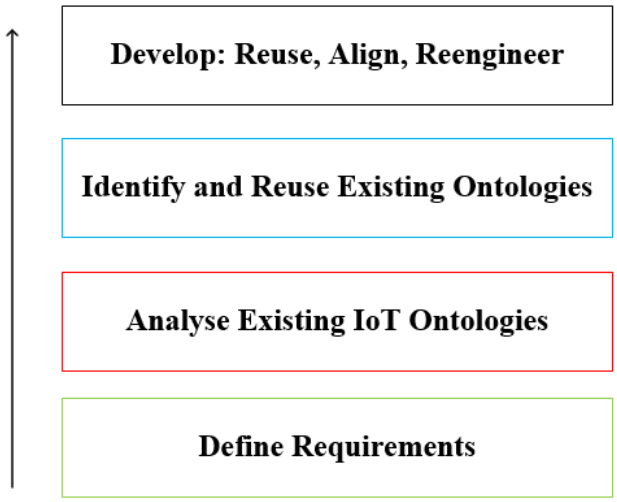

Figure 3-7 Development Process IoT-Ont

In the second step, the defined requirements were used to analyse existing IoT ontologies, this was shown in Tables 3-1 and 3-2. The third step is the identification and reuse of existing ontologies and modules in order to avoid redefinition of existent concepts and relations.

Chapter 4 demonstrates the IoT-OnT development process. Section 4.1 presents the main modules in IoT-Ont, followed by the main concepts in section 4.2

CHAPTER 4: DESIGN OF IOT-ONT

4.1 INTRODUCTION: IoT-Ont

The ontology's main modules are characterised as follows. Input data is described by the SSN/Sensors Module, the main classes of which come from SSN: ssn:Observation; ssn:Sensor. A generic sensor description is offered by ssn:Device together with its associated characteristics (for example, ssn:Deployment; ssn:OperatingRange). Meanwhile, the Resources Module characterises the system's interaction with physical objects. The main classes of this module are taken from SAN: san:Actuation; san:Actuator. Further, this module reuses some SSN classes, including ssn:Device, which are not specific to sensing. In addition, the Resources Module describes spatial coverage, device status (such as on/off) and RIFD tags. Figure 4-1 shows the main modules used to build IoT-Ont.

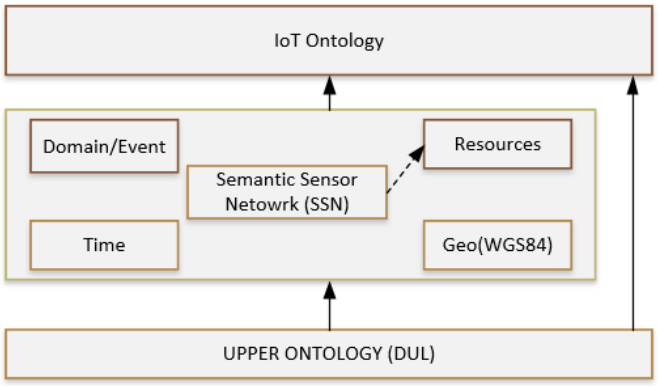

Figure 4-1 Main Modules of the IoT-Ont Ontology

Time is crucial in real-world systems. Timestamps can be used to describe temporal aspects; the time at which an observation was made or transferred. Multiple sensor's readings could be ordered according to their occurrence time. Users of an IoT application should be able to query sensors' observations in a specific timeframe. In order to model this, IoT-Ont reuses a well-established ontology for time as well as temporal properties and relations. *OWL Time*[10]

[10] http://www.w3.org/TR/owl-time/

describes temporal relationships and properties. It also supports durations as well as time intervals; these characteristics are advantageous when describing complex event specifications as well as inaccurate measurement times.

Location is an essential concept in the real world that is modelled in the IoT-Ont ontology. Various ontologies and location models are available today, including symbolic locations and geographical models. IoT-Ont, follows a practical approach by using the WGS84[11] ontology which facilitates the system usage and is flexible enough for more advanced use cases. The WGS84[12] coordinates are in fact the standard model for outdoor scenarios using GPS. This is a widely used ontology for location purposes in IoT.

Domain/Event: An event could be a simple event such as a high-temperature event in a servers' room, where direct action could be needed as turning on the Air Condition. The goal of this module is to relate with domain specific ontologies that needs to use IoT-Ont.

Table 4-1 summarises the main modules and provides a brief description of each.

[11] http://www.w3.org/2003/01/geo/
[12] http://www.w3.org/2003/01/geo/

Table 4-1 Main Modules of IoT-Ont

Module Name	Description
SSN	Input data is described by the SSN/Sensors Module, the main classes of which come from SSN: ssn:Observation; ssn:Sensor. A generic sensor description is offered by ssn:Device together with its associated characteristics (for example, ssn:Deployment; ssn:OperatingRange)
Resources	the Resources Module characterises the system's interaction with physical objects. The main classes of this module are taken from SAN: san:Actuation; san:Actuator. Further, this module reuses some SSN classes, including ssn:Device, which are not specific to sensing. In addition, the Resources Module describes spatial coverage, device status (such as on/off) and RIFD tags.
Time	*OWL Time*[13] describes temporal relationships and properties. It also supports durations as well as time intervals; these characteristics are advantageous when describing complex event specifications as well as inaccurate measurement times
Geo	The WGS84[14] ontology which facilitates the system usage and is flexible enough for more advanced use cases. The WGS84[15] coordinates are in fact the standard model for outdoor scenarios using GPS.
IoT-Ont Domain/Event	The idea behind the event/domain module is the enrichment of the received stream and converting it in a meaningful event according to a specific-domain characteristics, An event could be a simple event such as a high-temperature event in a servers' room, where direct action could be needed as turning on the Air Condition. This is dependant on a domain ontology and is used here only to facilitate possible extension of the ontology. (i.e., Any event source of the system instantiates concepts from the domain ontology. Then, some static information about the event source are stored in the data properties defined by its relative concepts.)

[13] http://www.w3.org/TR/owl-time/
[14] http://www.w3.org/2003/01/geo/

A more comprehensive overview of the ontology's concepts and relations is presented in Section 4.2.

4.2 Main Concepts

This section presents the main ontology concepts and illustrates its representation using the Protégé Software[16] in figure 4-2. As shown in the figure, the IoT-Thing concept represents a key concept that characterises the main actors in an IoT domain and relate them to various domains using the event concept as a mapper which makes more sense in real-world scenarios, where domain specific ontologies need not to worry about the development of an IoT ontology. Figure 4-2 shows how the introduction of the IoT-Thing concept facilitates the interaction between various actors in the IoT domain, such as a sensor, an actuator, a software, a service and a tag device.

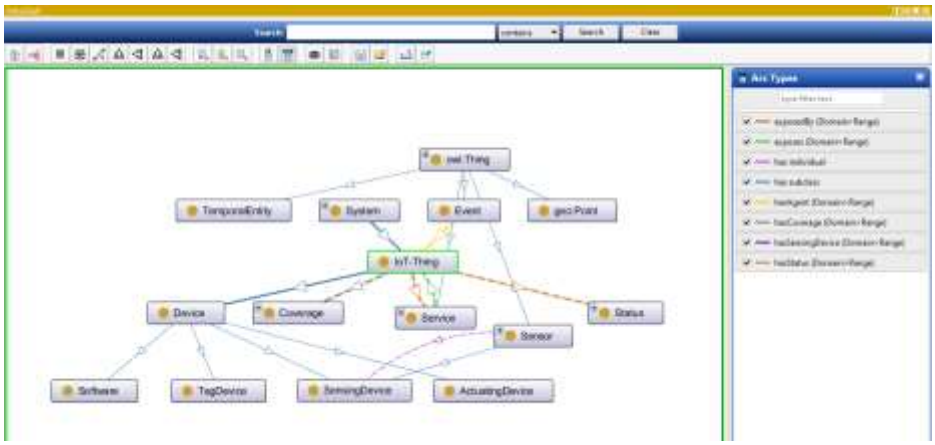

Figure 4-2 Simple Protégé Representation of main concepts in IoT-Ont

Table 4-2 provides a brief description of key concepts in IoT-Ont, for a more detailed description and implementation using OWL-DL, please see AppendixA.

[16] https://protege.stanford.edu/

Table 4-2 Key Concepts IoT-Ont

Concept Name	Description
IoT-Ont Thing	The concept Iot-Ont Thing enables us to define events based on their features in the context of smart environments and relate it to one of the actors in an IoT environment. These actors are defined using a subclass of IoT-Thing.
IoT-Ont Device	IoT-Ont Device is based on the ssn:Device which describes an abstract device and inherits all the properties of the class ssn:System (subcomponents, platform to which a system is attached, deployment in which a system participates, operating and survival range). In IoT-Ont ssn:Device is categorised in four sub-classes SensingDevice, ActuatingDevice, TagDevice and Software. All physical sensor devices are represented by the class ssn:SensingDevice in the ontology. Instances of this class possess all properties of the classes ssn:Sensor and ssn:Device.
IoT-Ont temporal entity	The pattern Iot-Ont Temporal Entity represents temporal intervals between any two timestamps through an observation process. It also could represent any instant of time. The two main classes of this pattern are IoT-Ont time:Interval (expresses an interval between two specific time-date values) and Iot-Ont time:instant (expresses a temporal entity with zero extent or duration), which are subsumed by the class time:temporalentity.
IoT-Ont Point	IoT-Ont follow a practical approach of using a simple representation of location using GPS. This is based on the class WGS84:Point, whose members are points. Points can be described using the 'lat', 'long' and 'alt' properties, as well as with other RDF properties defined elsewhere. For example, we might use an externally defined property such as 'bornNear' or 'withinFiveMilesFrom', or perhaps other properties for representing lat/long/alt in non-WGS84 systems. The 'lat' and 'long' properties take literal (i.e. textual values), each in decimal degrees. The 'alt' property is decimal metres about local reference ellipsoid.
IoT-Ont Sensing Device	Sensing Device is a subclass of SSN:Device and represents an IoT-Ont Thing that implements sensing.

IoT-Ont Sensor	This concepts is based on SSN:sensor it is an IoT-Ont Thing that can do (implements) sensing. A sensor is any entity that can follow a sensing method and thus observe some Property of a FeatureOfInterest. Sensors may be physical devices, computational methods, a laboratory setup with a person following a method, or any other thing that can follow a Sensing Method to observe a Property.
IoT-Ont Actuating Device	Represents a SSN:Device that can actuate over an object. It is a subclass of SSN:Device which in turn is a subclass of IoT-Ont Thing.
IoT-Ont Tag Device	Tag is a subclass of SSN:Device and could be a QR code, RFID or barcode.
IoT-Ont Software	IoT-Thing Software is a subclass of SSN:device and represents a piece of software or a service that runs in an IoT node. Can also be an abstracted device, issued from the composition of physical devices and abstract processing.
IoT-Ont Coverage	Represents the 'physical space' covered by an IoT-Ont Thing (i.e. a humidity sensor inside a room has a coverage of that room).
IoT-Ont Status	Represents the current status of an IoT-Ont Thing. Currently, this is limited only to represent whether the IoT-Ont Thing is on/off, this could be extended in the future.
IoT-Ont Property	Property refers to a feature of an IoT-Thing that is the interest of an observation process in an IoT environment. These properties differ based on the category of objects (e.g., mobile or non-mobile objects) besides the measurement criteria (e.g., location of objects) depending on which the observation process is conducted. The concept Iot-Ont Property is used to more tightly couple the representation of IoT-Ont Thing with some features or properties, which are measurable by sensors.
IoT-Ont Event	The IoT-Ont Event concept represents a bridge between the IoT-Ont and a domain-specific ontology that could be easily added to the ontology.
IoT-Ont Domain	A domain is a concept that represents one of the domain applications in an IoT environment(Agriculture, HealthCare, Military, etc.). The main usage of this class is to provide more context to the observed data and thus provide data enrichment based on data properties.

IoT-Ont Observation	An DUL:Observation is a Situation in which a Sensing method has been used to estimate or calculate a value of a Property of a FeatureOfInterest. Links to Sensing and Sensor describe what made the Observation and how; links to Property and Feature detail what was sensed; the result is the output of a Sensor; other metadata details times and so forth.

4.3 EPS-DL EVENT PROCESSING SYNTAX BASED ON DESCRIPTION LANGUAGE

Events were presented in Section 2.6.1 as methods for reporting changes and drawing a parallel between declarative and event-driven programming. Actually, an event occurrence could be considered as a declaration regarding something that has altered the actual state or as something has occurred (*Who* can use what, and *how it* is used, is not stated). However, patterns and rules are specified to identify these complex matters of interests.

A complex event is derived, at any point of time, if any rule's premises are proven true based on the existing events and another available background knowledge. This means that statements are declared about complex events alongside simple events. These statements could involve semantic, temporal, or any possible relations amongst events. However, their nature is *declarative*, that is, the rules provides the exact definitions of the patterns. However, it does not identify applicable methods to detect patterns nor it specifies the order these rules should be evaluated.

This section introduces EPS-DL, and lay out the requirements for developing an event processing syntax based on description logic for an IoT environment.

Why EPS-DL?

Stream Reasoning (SR), investigated, envisioned and proved the possibility to make sense of heterogeneous streaming data (Anicic et al., 2012). Current research in this field investigates methods to enhance existing models to provide high reasoning level. The Event Processing

Syntax Based on Description Logic (EPS-DL) combines both Complex Even Processing and Description logic in order to provide stream reasoning. This syntax is based on RDF Stream Processing Engine. Section 5.3 briefly introduces the RDF stream model.

The current state of the art in RDF Stream Processing (RSP) proposes various implementations and models to integrate Semantic Web technologies with operators used within a Data Stream Management System (DSMS) such as the windows operator presented in section 2.7.1. In the meantime, there are only a small number of solutions combining the Semantic Web with Complex Event Processing (CEP), which involves relevant features, like the identification of events sequences in streams. RSP query languages that currently support CEP features comprise various limitations: C-SPARQL supports limited pattern detection by using a timestamp function, while EP-SPARQL is capable of identifying sequences, but it lacks more sophisticated time operators and its selection policies are not defined formally. This chapter introduces an event processing syntax that builds upon state-of-the-art RSP query languages. EPS-DL, is developed to support CEP operators. We show that it provides more features than the ones offered by present RSP query languages. Moreover, we provide a comparison with other languages in this field, against the development requirements in Section 4.6. Section 4.4 provides a requirement study for developing EPS-DL based on literature.

4.4 EPS-DL: Requirements

This section gives a short description of the requirements we build upon to develop the proposed syntax. The design of EPS-DL design is based on the following requirements Requirement1 – Requirement 5:

Requirement 1

Semantic Event Representation: This enables multiple heterogeneous sources to be integrated, and to derive implicit data based on the background knowledge by Using description Logic. This fits perfectly within a highly heterogeneous IoT environment. (Taylor and Leidinger, 2015)

Requirement 2

Event Processing: This enables high-level ontological concepts that capture temporal dependencies to be combined, and thus allows creating complex events. The usage of syntax time operators to create and directly manipulate abstracted events (such as pattern matching) should be possible. (Anicic et al., 2012)

Requirement 3

The Syntax should be capable of processing RDF graph-based streams. While the stream data items are described in single RDF statements by early RSP data models, the standardisation effort proposed recently by the W3C RSP-CG[17] adopts the use of RDF graph. The RDF graph model generalises of the single RDF statements, as a stream of time-annotated RDF statements could be modelled as a stream of time-annotated RDF graphs, each comprising of one statement. For this reason, acknowledging this requirement is essential to realise a generic RDF stream query model.

Requirement 4

The language should capture SEQ behaviour proposed in the EP-SPARQL. EP-SPARQL is the RSP language with the most extensive support for features of CEPs, implementing many

[17] https://www.w3.org/community/rsp/

operators to identify complex events, e.g., EQUALS, EQUALSOPTIONAL, SEQ, and OPTIONALSEQ. Features of semantically-enabled streaming language.

Requirement 5

The language must capture the features of CEPs provided by current RSP engines. In this research, we are focusing on the SEQ operator, AND operator and OR operator: the most fundamental building block in CEP. Intuitively, E1 *SEQ* E2: is used to identify events that match the E1 pattern followed by events matching the E2 pattern. Modifiers such as *Every*, *Within*, and *Not* are also being implemented. Section 4.5 introduces briefly the RDF Stream Model, which is backbone in any stream reasoning model.

4.5 RDF Stream Model

The Resource Description Framework (RDF) is a foundation for processing metadata; it supports interoperability amongst applications exchanging machine-understandable data on the Web. RDF is the standard web-based model for data exchange, and is the model used with all the researched IoT models. A key factor for the EPS-DL is the data model utilised to represent the RDF streams. Section 4.5.1 briefly presents the formalisation of RDF triples and graphs, followed by the extensions used to represent stream graphs.

4.5.1 RDF Statements and Graphs

An RDF statement is the basic data entity in the RDF data model, for such a data item is defined as an RDF triple, which is a set of three entities. In literature it is usually referred to as ***'t'*** composed of **<s,p,o>** where, *s, p, o* are the subject, predicate and object respectively, such as the triples in Listing 4-1:

Listing 4-1 RDF triple examples

```
<:sara :meets :sam>
```

Whereas an RDF graph comprises a *set* of RDF triples, such as the example in Listing 4-2:

Listing 4-2 RDF graph example

```
<:sara :participates :meeting>
<:sam :participates :meeting>
<:meeting :haslocation :meetingroom2>
```

4.5.2 RDF Stream Graphs

An RDF Stream graph is a timestamped RDF graph, which is a pair (G; t), where G is the RDF graph, and t is a time instant. An RDF stream S is (possibly) an infinite timestamped RDF graphs sequenced in a non-decreasing timely order:

$$S = (G_1; t_1); (G_2; t_2); (G_3; t_3); (G_4; t_4), \ldots$$

Where, for every, i > 0, $(G_i; t_i)$ is a timestamped RDF graph and $t_i < t_i+1$.

4.6 Time Relations

This section presents the time operators and modifiers supported by EPS-DL and provides a comparison with the state of the art event processing models against the requirements of an event-processing model presented earlier in Section 4.4.

In order to simplify this section, we use a forest fire-detection example. The aim is to detect the existence of fire in a forest, having no direct way to do so. Instead, the forest contains various sensors for the presence detection of Carbon Monoxide (CO) and for measuring the temperature. Here, the RDF stream is a timestamped sequence of numbers used to represent the average temperature in one spot of the forest, and an event is a notification of detecting a certain CO threshold level".

In order to detect a fire in one spot, it is crucial to represents the time relation between abnormal temperature and CO detection. If the temperature is high at a certain time, t_1, this needs to be combined with CO detection at either a similar time, t_1, or within a certain timeframe.

This need is captured by creating event-processing operators within EPS-DL where direct manipulation (such as pattern matching) over events is feasible.

Simple temporal pattern such as sequence 'SEQ' should be included in order to provide enough expressiveness and this could be used in combination with modifiers that allows even more expressiveness to capture the complexity of the entire domain (such as Every, Within, and Not).

Concerning time, a *point-based time semantics* (Bohlen et al., 1998) for events is adopted in state-of-the-art models; where an event ***e*** is defined as a pair of an RDF graph (G) which contains the event's statements, and a timestamp (t) associated to the occurrence time. This allows events to be partially ordered, and does not prevent events from occurring at the same time. At EPS-DL development, the following time-aware operators were considered for event processing:

- Sequence, SEQ: $E_1(G_1, t_1)$ seq $E_2(G_2, t_2)$, represents a sequence of two events, it returns true if and only if both events occur and $t_1 > t_2$, on other words, the occurrence of E_1 is followed by the occurrence of E_2.
- AND: $E_1(G_1, t_1)$ and $E_2(G_2, t_2)$, this pattern is detected if instances of E_1 and E_2 has occurred in whatever order. Regardless the ordering of their occurrence, this will return true when both of them have occurred.
- OR: $E_1(G_1, t_1)$ or $E_2(G_2, t_2)$, returns true if and only if one of the events occur.

Next are the optional time-modifiers that were developed to enhance event processing:

- EVERY: This modifier keeps evaluating the occurrence of a particular event.

- WITHIN: This modifier limits the pattern's validity, it restricts the evalaution of a pattern to a certain time period.
- NOT: This modifier simply acts as a logical not, it contradicts the pattern's value.

EPS-DL supports the usage of Event Filtering and Event Joining. Event Filtering allows removing irrelevant events, and Event Joining allows combining events over multiple event streams to achieve intelligent decision making. Filters and joins are utilised by most CEPs models in literature. However, the existing models do not implement all the time operators and modifiers together. Table 4-3 provides a comparison of existing CEP models in regard with their support of various operators, filter and joins.

Table 4-3 Comparison of existing CEP Models

	Semantic Event Declaration	**Inference**	**Event processing Operators**	**Filters**	**Joins**
EPL	Relational	None	All	Yes	Yes
MASSIF (Bonte et al., 2017)	DL Axioms	OWL 2DL	None	Yes	No
EP-SPARQL (Barbieri et al., 2010)	RDF BGP	RDFs	SEQ	Yes	Yes
C-SPARQL (Barbieri et al., 2010)	RDF	RDFs	WINDOW	Yes	Yes
ODECP (Taylor and Leidinger, 2011)	OWL	RDFs	SEQ,AND,OR	No	No
OCEM-IoT	DL Axioms	OWL 2DL	All	Yes	Yes

Table 4-3 summarises the similarities and differences between the state-of-the-art models, presented in Section 2.7 and 2.8 for event processing and EPS-DL. The table emphasises the

additions made by the EPS-DL against the aforementioned requirements in Section 4.4. This approach combines both the semantic declaration of events and event processing. It also supports temporal inference over events, since all the typical event processing operators are implemented, and introduces new optional modifiers allowing more expressiveness in defining patterns, whereas other models implements only some of these operators.

4.7 EPS-DL Basic Syntax

EPS-DL was influenced by previous SPARQL streaming-oriented extensions proposals, mainly EP-SPARQL (Anicic et al., 2011), C-SPARQL (Barbieri et al., 2010) and by RSEP-QL (Dell'Aglio et al., 2016). EPS-DL syntax, is developed on the concept of continuous querying of RDF triples (RDF streams). EPS-DL syntax is totally based on the SPARQL syntax, this allows it to support aggregates, which are an important feature of SPARQL that is crucial for processing data streams.

This section presents the EPS-DL syntax. It focuses on the grammar that has not been provided by literature solutions. The parts that depend on other grammars, specifically: The Event Declaration Clause is used to define events; it is based on the classes formulation which is a basic feature of the Manchester Syntax[18]. Figure 4-3 provides an example for declaring an event using the Manchester Syntax. The Constraint clause is used to define filters; it is based on the SPARQL 1.1 grammar; Listing 4-4 provides an example for a constraint filter definition based on SPAQRL. The MATCH clause is used to define time relations over events' declarations.

This section presents the basic syntax of EPS-DL as follows:

[18] https://www.w3.org/TR/owl2-manchester-syntax/

Event Declaration

The Event Declaration clause is used to define events. It is based on the classes formulation which is a basic feature of the Manchester Syntax.[19] OWL 2 ontologies are used with information written in RDF, the Manchester Syntax is a user-friendly and compact syntax for OWL 2 ontologies.

```
description ::= conjunction 'or' conjunction { 'or' conjunction }
        | conjunction
conjunction ::= classIRI 'that' [ 'not' ] restriction { 'and' [ 'not' ] restriction }
        | primary 'and' primary { 'and' primary }
        | primary
primary ::= [ 'not' ] ( restriction | atomic )
restriction ::= objectPropertyExpression 'some' primary
        | objectPropertyExpression 'only' primary
        | objectPropertyExpression 'value' individual
        | objectPropertyExpression 'Self'
        | objectPropertyExpression 'min' nonNegativeInteger [ primary ]
        | objectPropertyExpression 'max' nonNegativeInteger [ primary ]
        | objectPropertyExpression 'exactly' nonNegativeInteger [ primary ]
        | dataPropertyExpression 'some' dataPrimary
        | dataPropertyExpression 'only' dataPrimary
        | dataPropertyExpression 'value' literal
        | dataPropertyExpression 'min' nonNegativeInteger [ dataPrimary ]
        | dataPropertyExpression 'max' nonNegativeInteger [ dataPrimary ]
        | dataPropertyExpression 'exactly' nonNegativeInteger [ dataPrimary ]
atomic ::= classIRI
        | '{' individualList '}'
        | '(' description ')'
```

Figure 4-3 Manchester Description Syntax for Event Declaration[20]

A simple example of an event declaration in EPS-DL is presented below:

$$EVENT\colon\ PlaneAbnormalPressureEvent\ subClassOf$$

$$AbnormalPressureEvent$$

$$and\ \left(observation_{result} some\left(hasValue\ (hasDataValue \geq\ 12)\right)\right)$$

$$and\ (hasLocation\ some\ Plane)$$

Event Clause

[19] https://www.w3.org/TR/owl2-manchester-syntax/
[20] https://www.w3.org/TR/owl2-manchester-syntax/#description

An event clause in EPS-DL is described as follows:

$$[\mathbf{NAMED}]\ '\mathbf{EVENT}'\ EVENT_{IRI}(EVENT_{DECLARATION})|(PATTERN_{DECLARATION})$$

Pattern Declaration

Event pattern declaration is used to match event or multiple events whenever an event matches the definition of a particular pattern. In order to declare a pattern in EPS-DL the keyword 'WHEN is used as follows:

$$'\boldsymbol{WHEN}'\boldsymbol{PATTERN}_{\boldsymbol{EXPRESSION}}\ [\boldsymbol{IF}_{\boldsymbol{DECLARATION}}]$$

Pattern Expression

The pattern expression uses the keyword MATCH, then the Followed By Expression "–>" and finally the time modifier WITHIN to indicate the Time Period.

$$'\mathbf{MATCH}'EXPRESSION\ [\mathbf{WITHIN}\ TimePeriod]$$

Example:

MATCH: HighAverageTemperature –> :CODetectionEvent WITHIN(10m)

Time Period

Represents a period of time, it is always an integer value followed by a unit which could represent millisecond(ms), second(s), minutes(m), hour(h), day(d), week(w).

Time Period: INTEGER(UNIT), i.e (10ms), (5m)

$$'\mathbf{INTEGER}'(ms, s, m, h, d, w, ..)$$

Followed by Expression '–>'

The followed by operator ' ->' specifies that the left hand expression must first become true and after that, the right hand expression is evaluated for matching events.

Consider this example ‘X -> Y’. First, evaluate the occurrence of event X and only if this event is encountered, then evaluate the occurrence of event Y. X and Y can themselves be nested event pattern expressions.

The followed by expression is expressed in description logic as follows:

$$Or_{EXPRESSION}\left(([\mathbf{NOT}]\ '\mathbf{SEQ}')\ Or_{EXPRESSION}\right)$$

EVERY Expression

Whenever a pattern evaluates to true or false, the search for such a pattern stops. However, the every operator introduces a new way to keep looking at the pattern even if it evaluates to true or false. In other words, using the Every operator means that sub expression of a pattern restarts whenever this sub expression evaluates to either true or false. The example below shows the benefit provided by this operator.

Example:

X:

This pattern evaluates to true when encountering an X event but then it stops looking for any X events that can occur afterwards.

EVERY X:

This pattern keeps looking for an Event X, even when an event X is encountered. It does stop looking.

Or Expression

Represents a logical OR for use in pattern expressions. The **‘**OR’ operator requires either one of the expressions to become true before the whole expression becomes true.

Look for either event X or event Y; “X OR Y”. As always in event processing, X and Y can themselves be nested expressions as well. This is described in description logic as follows:

$$\text{AND}_{EXPRESSION}(\textbf{OR}\ \text{AND}_{EXPRESSION})$$

Example:

EVERY (*AverageTemperature(value > 40)) OR* (*AverageTemperature (value < 0)*

This expression detects all temperatures above or below a given specific threshold.

AND Expression

The ‘AND’ operator requires both nested pattern expressions to become true before the whole expression becomes true (a join pattern). It represents a logical AND and is described as follows in DL:

$$EveryOrNot_{EXPRESSION}(\textbf{'AND'}\ EveryOrNot_{EXPRESSION})$$

This pattern matches when both event X and event Y event arrive, at the time the last of the two events arrive “X AND Y”.

The pattern below matches on any sequence of X event followed by Y event and then W event followed by an Z event, or an event W followed by an event Z and an event X followed by an event Y: (X –>Y) AND (W –>Z).

If Declaration

An If Declaration is just like an if clause in any programming value, it checks against predefined conditions(rules) and returns true if the conditions were met. The If Declaration is represented as follows in DL:

$$\textbf{IF}\ \{\textbf{'EVENT'}\ (\boldsymbol{EVENT_{IRI}|VAR})\boldsymbol{FILTER_{EXPRESSION}}\}$$

An Example of an IF declaration in order to verify that the CO detection event and the high temperature detection event are in the same location is expressed in EPS-DL in Listing 4-3 as follows:

Listing 4-3 If Declaration example in EPS-DL

```
IF{
EVENT: CODetection_EVENT {?Location1 DUL: HasValue ?v}
EVENT: HighTemperature_EVENT {?Location2 DUL:HasValue ?v}
}
```

Filter Expression

Filtering allows to eliminates irrelevant events and thus improve reasoning over the more relevant events, which also saves reasoning time. The Filter defines the type of event to be filtered against, it also provides an optional expression that returns true if the filter should consider the event, or false to reject the event. There are a set of operators used in event processing for filtering (such as, equals, not equals, giving ranges of values to filter against, using comparison operators >, <, etc.). The filter expression is defines in DL as follows:

$$\{(BGP|'\mathbf{FILTER}'CONSTRAINT)\}$$

The constraint filter is widely used in event processing, in all the surveyed models that implements filtering, it is based on the SPARQL grammar. The example in Listing 6-4 shows a Filter constraints definition.

Listing 4-4 Constraint Filter Based on SPARQL.[21]

Input graph:

```
@prefix dc:   <http://purl.org/dc/elements/1.1/> .
:book1  dc:title  "SPARQL Tutorial" .
:book2  dc:title  "Semantics" .
```

Query:

```
PREFIX  dc:  <http://purl.org/dc/elements/1.1/>
SELECT  ?title
WHERE   { ?x dc:title ?title
          FILTER regex(?title, "^SPARQL")
        }
```

Query Result

Title
"SPARQL Tutorial"

Listing 4-4 illustrates a simple constraint filter based on SPARQL, where only the book that has 'SPARQL' in its title is chosen.

4.8 Event processing based on Description Logic

This section explains the usage of operators in event processing alongside the ontological concepts. The fire detection example would be used to simplify the domain and to illustrate how EPS-DL declare events and defines filters. Assuming no straightforward method to detect fire, but the assumption of its existence could be confirmed through detecting CO and unusual temperature measurements in the same period. This should sound as a simple example, however there are many complexities in trying to define such simple rules.

1. Complexity of the domain: How to determine whether the temperature observed is unusual or not?
2. Integration of data: How to combine the local data sources and external data sources be combined?

[21] https://www.w3.org/TR/rdf-sparql-query/#rConstraint

3. Time Relations: How the time relations between CO detection events and unusual temperature can be modelled in order to detect fire?

4.8.1 Semantic Event Representation

For the mentioned use case, the objective is to detect abnormal temperature measurements and by combining them with CO detection events, a fire event could be detected. Semantic event representation has been widely used to tackle the challenges of Domain complexity and data integration. Regarding the domain complexity, the normal temperature measurement could differ according to each city, building, room, etc. Through an Integrated Conceptual Model (ICM), systems for Static Information Integration (such as Ontology Based Data Access systems) provide a solution for these circumstances. The ICM allows queries to be answered using a common vocabulary specified formally by ontological languages (such as OWL, DL, etc.) and across heterogeneous data sources. The ICM of the above-mentioned example comprises the axioms illustrated in Table 4-4.

Table 4-4 Example definition using DL axioms

CODetectionEvent ≡ ∃hasContext.(∃observedProperty.COthreshold) TemperatureEvent ≡ Observation	1
∩ (∃observedProperty.Temperature)	2
AbnormalTemperatureEvent ⊆ TemperatureEvent OfficeAbnormalTemperatureEvent ⊆ AbnormalTemperatureEvent ∩ (∃observationResult.[hasValue>35])	3
∩ (∃hasLocation.Office) ServersRoomTemperatureEvent ⊆ AbnormalTemperatureEvent ∩ (∃observationResult.[hasValue>15)	4
∩ (∃hasLocation.ServersRoom)	5

The Data integration challenge, necessitates a generic data model. The Semantic Web community commonly uses the RDF model in order to resolve heterogeneity for static data. In

this work, RDF may be sufficient in representing the background static knowledge, however, in order to represent the continuous streams, RDF streams are required (Section 4.5.2).

Finally, if combined with the reasoner, the ICM may benefit fully of the background knowledge in order to obtain information that described only implicitly by the raw data, this is shown in the axioms (4) and (5) in Table 4-4. The entailment to be used to represent any ICM is dependent on the domain; usually it is a compromise with the system's final complexity. With regard to the use case, and for the purpose of providing meaningful expressions, OWL 2 DL[22] is used in this work.

4.8.2 EPS-DL Event Declaration

In the above-mentioned example, our interest is in both the temperature abnormal events and CO detection readings in order to detect the presence of fire. Section 4.8.1 showed that semantic event representation (Requirement 1) is achievable using simply any ICM. Higher-level events, however, could be described using an EPS-DL query. This can be done by deploying an Event Declaration clause (Section 4.7). Listing 4-5 provides an example for a declaration of an EPS-DL event. The Manchester syntax[23] has been selected since it is more concise than the RDF model and focuses on the idea of specifying events using high-level abstractions. In addition, it has been already recommended and combined with SPARQL in literature (Sirin et al., 2010).

Listing 4-5 Event Declaration in EPS-DL

```
EVENT : RoomAbnormalTemperaturEvent subClassOf
AbnormalTemperaturEvent
and ( observation_result some ( hasValue ( hasDataValue >= 30)))
and ( hasLocation some Room ) )
```

[22] OWL 2 https://www.w3.org/TR/owl2-direct-semantics/
[23] OWL2-Manchester Syntax https://www.w3.org/TR/owl2-manchester-syntax/

The Event defined using this clause is added to the T-Box of IoT-Ont ontology. The defined events will go through Reasoner in order to accomplish the inference process. Each defined event in EPS-DL will be translated to class expressions using OWL. Since the event definition is based on the DL Manchester syntax, the translation becomes simple. For example, the translation of the example in Listing 4-5 is illustrated in EPS-DL in Listing 4-6 as follows:

Listing 4-6 Event in EPS-DL

```
RoomAbnormalTemperaturEvent ⊆ AbnormalTemperatureEvent
∩ (∃observationResult.[hasValue>30])
∩ (∃hasLocation.Room)
```

EPS-DL utilises the time operators introduced in Section 4.6. Listing 4-7 shows how to define the fire detection event based on the exploitation of temporal relations between the CO_DetectionEvent and the AbnormalTemperaturEvent. EPS-DL carries out Event processing over high-level concepts as stated in the requirements for developing an event processing syntax in Section 4.4. An example of which is available in Listing 4-8, this is accomplished by the use of the sub-clause Pattern-Expression of the Pattern-Declaration clause. The event patterns' definition relies on user-defined concepts or existent concepts within the ontology. Here it is assumed that the CO-Detection Event is already defined in the ontology.

Listing 4-7 Fire declaration event in EPS-DL

```
NAMED EVENT : FireEvent { MATCH : AbnormalTemperaturEvent SEQ : CO_DetectionEvent
WITHIN (10m)
}
```

Listing 4-8 Using filters in event patterns

```
NAMED EVENT : Fire {
      MATCH : AbnormalTemperaturEvent SEQ : CO-DetectionEvent WITHIN (10m)
      IF {
            EVENT : AbnormalTemperaturEvent { ?tmpSnsLoc : hasValue ?v}
            EVENT : CO-DetectionEvent { ?COSnsLoc : hasValue ?v ;
            ?CO-Obs ssn : observationResult ; : hasValue? COLevel
            FILTER (? COLevel == "7" ^^xsd : integer ) }
           }
}
```

Finally, The IF Declaration clause allows expressing joins and filters over RDF Streams. Deploying a syntax based on SPARQL language, it becomes possible to identify a particular graph pattern matching for each event, for example, the "Abnormal Temperature Event" in Listing 4-8, and joins that utilise the name-based notation; variables that have the same name get the same binding e (e.g., variable ?v in Listing 4-8). Filters could be declared based on the SPARQL grammar for the clause Filter, (e.g., variable ?COLevel in Listing 4-8).

4.9 Summary

The ontology was initially semantically validated using the automated check of the ontology's consistency through the semantic reasoners Hermit[24] and Pellet[25] which are built-in Protégé, during the development process, the ontology has consistently passed many times through the reasoners, this however, only validates that there are no contradictory statements in the ontology.

This was followed by competency question checking and iterative ontology revisions until the questions could be answered successfully. However, it is critical to further test whether these stated objectives are an adequate representation of the intended objectives, and even further testing to determine whether these intended objectives are a valid representation of what the

[24] http://www.hermit-reasoner.com/
[25] https://www.w3.org/2001/sw/wiki/Pellet

objectives should be. This testing has been conducted through the evaluation of the model by domain experts in terms of: whether it meets their view of what an IoT ontology should be, whether the concepts and main modules are adequate, and whether this is genuinely a valid representation of the wider IoT domain. The survey used for the evaluation process is presented in (Appendix A), and the evaluation process and results are presented in Chapter 6.

This chapter presented the ontology development process for an IoT environment by researching the conceptual and functional requirements for ontology development. Competency questions were developed in order to derive main concepts and modules for an IoT ontology. A review on state-of-the-art IoT ontologies was presented, these ontologies were compared then against the deduced requirements. Chapter 4 also presents the EPS-DL syntax deployed alongside the ontology in order to extract properties used for filtering and joining over time which aims to facilitate and enhance reasoning.

In the state-of-the-art Complex Event Processing models (CEPs), none of the existing solutions implement all the time manipulation operators together. Table 4-3 provided a comparison of existing CEP models in regard with their support of various operators, filter and joins and methods for inference and reasoning. Table 4-3 summarises the similarities and differences between the state-of-the-art models, presented in Section 2.7 and 2.8 for event processing and EPS-DL. The table emphasises the additions made by EPS-DL against the aforementioned requirements in Section 4.4.

This approach combines both the semantic declaration of events and event processing. It also supports temporal inference over events, since all the typical event processing operators are implemented, and introduces new optional modifiers allowing more expressiveness in defining patterns, whereas other models implements only some of these operators. In Chapter 5, we

present the overall architecture of OCEM-IoT and illustrate its different layers based on the IoT-Ont and the EPS-DL syntax.

CHAPTER 5: OCEM-IOT SPECIFICATION AND DESIGN

5.1 Introduction: Ontology-Based Event Processing

Information in the Internet of Things (IoT) has been represented traditionally as data streams, such as unbounded data sequences, or events representing notifications about occurrences of facts. Stream Reasoning (SR) (Della Valle et al., 2009) examines the ways in which Stream Processing technologies and Sematic Web can be combined to enable real-time decision making throughout multiple sources of data. SR studies ways of exploiting data streams time ordering to conduct temporal and deductive reasoning spontaneously.

Data as well as events in IoT results from various types of sensors. Such heterogeneity prevents the performance of queries throughout such data sources. The domain complexity represents another obstacle in this field.

To further explain the domain, a simple example is provided: our intent is to detect the existence of fire in a forest, without the existence of any direct method. The forest has various sensors for detecting the presence of Carbon Monoxide (CO) as well as for measuring the temperature. In such a situation, a data stream indicates a timestamped number sequence that represent the average temperature in one spot of the forest, while an event refers to a notification about the detection of CO. Considering the fact that the events and data result from various types of sensors, such heterogeneity impedes to carry out queries throughout data sources. The domain complexity represents another complication. If there is a fire, there will be a rise in temperature, but the question is how abnormal temperatures and normal temperatures can be differentiated? Another question is what will happen if we have different spots in the forest? Such information indicates the background knowledge which decision-making systems must merge with live data for gaining an answer. Eventually with the

assumption that we detected CO and the presence of an event of abnormal temperature, they should be timely related.

This thesis presents an approach that addresses the heterogeneity problem within an IoT infrastructure, this involves solving data variety (produced from heterogeneous sources), and is able to combine data and background knowledge, deducting related information and conducting temporal reasoning by combining data streams from events and sensors. Deductive reasoning's temporal extensions can append time relations to the ontological language, thus, easily diverging into intractability. Further, semantic Complex Event Processing is restricted to an event's semantic description and does not focus on the processing. This thesis proposes an ontology based event processing approach that conducts the event processing over high-level concepts that results from deductive reasoning, while excluding time relations at the ontological level.

5.2 OCEM-IoT: Lifecycle

In the first Phase of the model, real-time sensor data will be semantically represented using the IoT-Ont ontology for an IoT domain, at this stage metadata will be attached to data in order to support information interoperability. Events are then abstracted by applying the ontological modelling and the usage of a semantic reasoner engine. The raw observations now are converted to meaningful events where the context of time and location are implemented.

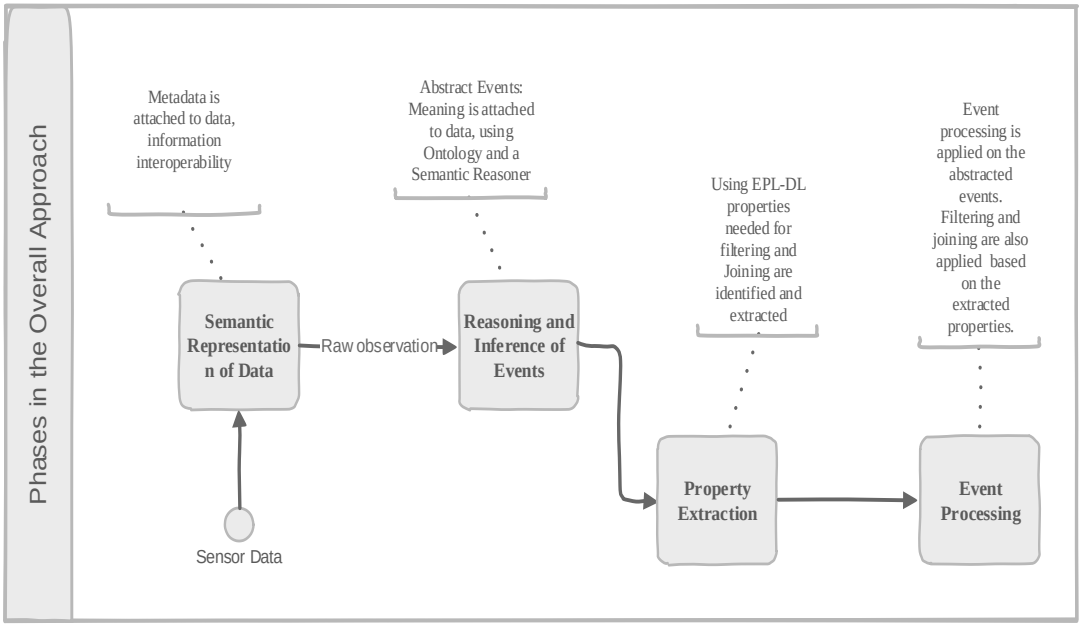

Figure 5-1 OCEM-IoT Lifecycle

The overall approach is illustrated in Figure 5-1. By using descriptive logic event processing syntax, properties relevant for filtering and/or joining are to be extracted. The abstracted events alongside the extracted properties are then forwarded to the event processing phase where the actual event processing over abstracted events takes place, besides any event processing technique such as the application of filtering/ joining, pattern matching, etc. A more comprehensive description of each layer of the model is presented in Sections 5.3 and 5.4.

5.3 OCEM-IoT: Architectural Layers

OCEM-IoT architecture is presented in Figure 5-1. It consists of three main layers: 1) Event Abstraction Layer, 2) Property Extraction Layer and 3) Event Processing Layer. However, it is vital to provide a short discussion of the sensor data layer in which sensor data stream sources are placed, before continuing with the descriptions of the main layers.

The fast development of the Internet of Things and the widespread use of smart mobile phones, knowledge bases and wearable smart devices significantly contribute to the big volumes of generated data and to the heterogeneity of this data. This heterogeneity refers primarily to time-related aspects (e.g., data streams/databases or static knowledge), sensor type (e.g., physical sensors such humidity or temperature and virtual sensors such as social media), various data formats, origin (e.g., public entity, private organisations, and individuals), and so forth. Semantic techniques have therefore been utilised to address the aforementioned heterogeneity issues.

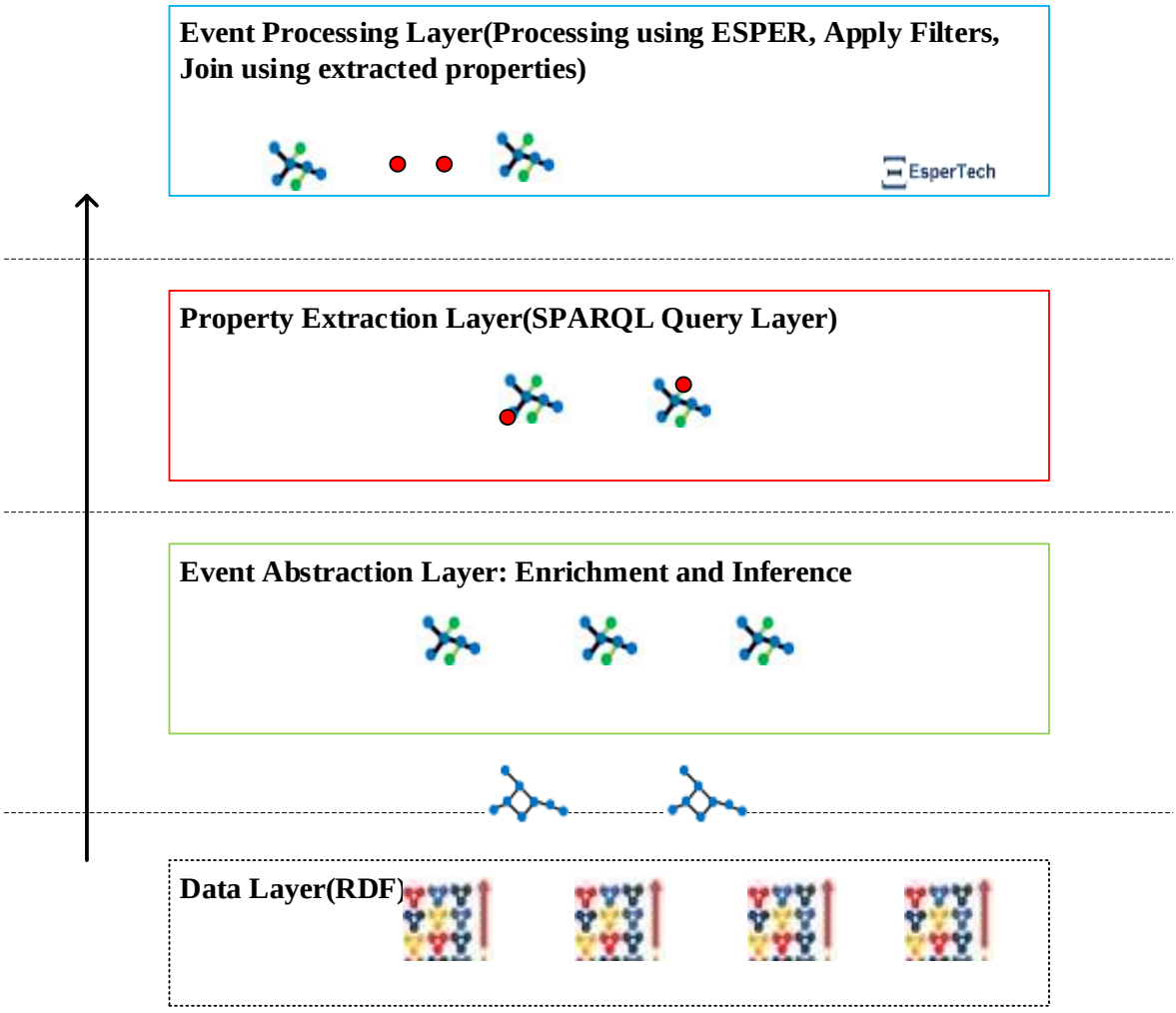

Figure 5-2 OCEM-IoT Architecture

A generic data model is required to guarantee data integration. The Semantic Web community has adopted the usage of RDF representation model to overcome the heterogeneity for static data. However, in this research, RDF is sufficient for representing the background static knowledge but not to represent sensor data streams, which require the usage of RDF Streams. This research assumes the receipt of data in the form of RDF stream without delving into various mapping languages used in this field. Regarding the temporal aspect for events, a point-based time semantics is utilised, where any event *E* is defined as a pair *(G,t)*, *G* representing the RDF graph comprising of the event statements and *t* representing the respective timestamps.

5.4 Components of Architectural Layers

This section describes the model architecture for an event processing system based on ontological concepts; the model supports the usage of the EPS-DL syntax. Figure 5-2 shows the four layers of the model, each layer is concerned with a particular processing task of utilising the RDF Streams in order to generate results of an EPS-DL query.

5.4.1 Sensor Data Stream

Incoming events are assumed to be a pair (G, t) *G* representing the RDF graph and *t* representing the respective timestamps (RDF Stream in Figure 5-3).

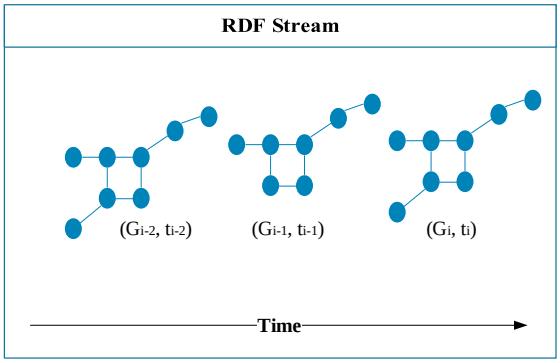

Figure 5-3 Sensor Data (RDF Graph)

RDF[26] has a plain data model, which makes it easy to manipulate and process by applications. The data model is independent of any specific serialisation syntax. RDF has a formal semantics, which offers a dependable framework for reasoning about the RDF expression meaning. It particularly supports strictly defined notions of entailment that offer a framework for the definition of reliable inference in RDF data.

The basic construct of an RDF expression is a set of triples, each triple comprises of a subject, a predicate and an object. An RDF graph is defined as a set of RDF triples. This can be presented by a node, and directed-line diagram, where each triple is represented as a node-line-node link (that is where the term "graph" comes from) See Figure 5-4.

[26] https://www.w3.org/TR/rdf-concepts

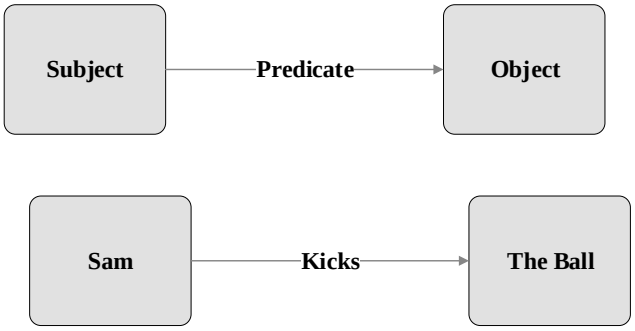

Figure 5-4 RDF Graph Data Model consists of Triples

Each RDF triple is a statement that represents the relation between the linked nodes (the things in IoT). The triple is composed of three parts:

- The subject.
- The object.
- The predicate (also referred to as a property) that denotes a relationship.

The *set* of nodes of an RDF graph is the *set* of subjects and objects of triples in the graph. The RDF triple assertion signifies that a relationship, indicated by the predicate, holds between the things denoted by subject and object of the triple. The RDF graph signifies the assertion of all the triples in the graph, so the meaning of an RDF graph is the conjunction (logical AND) of the statements associated to all the triples it includes. A more comprehensive overview of the RDF data model used with relation to our work was presented earlier in section 4.5.

5.4.2 Event Abstraction Layer (Enrichment and Reasoning)

In an IoT environment, it is continuously required to sense and respond to specific changes. In such an environment, the focus needs to be shifted from analysing raw data streams to analysing higher-level information they conceal, namely events. The inference time can be reduced by methods of inferring and integrating event-related information from accessible data sources.

The purpose of the Event Abstraction Layer is to bridge the gap between raw sensor data and applications used for event processing as shown in Figure 5-5.

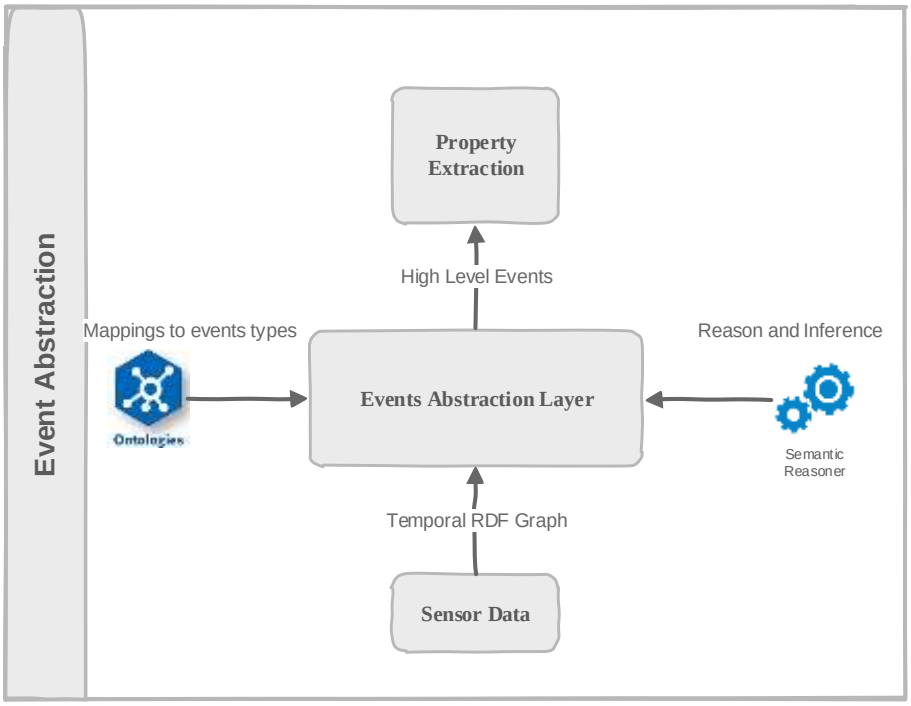

Figure 5-5 Event Abstraction Layer Analyses Data Stream and generates streams of Events

Event processing in an IoT environment does not have to deal with raw sensor data. The role of the Event Abstraction Layer is to consume temporal RDF streams of observations, and to handle mappings between event patterns and event types, and thus the inferred events are classified automatically depending on the domain knowledge provided. The Event Abstraction Layer also generates high-level events that are represented by the same model. Data integration consists in providing users with a uniform view on data residing at different sources (Lenzerini, 2002). The Event Abstraction Layer supports data integration by the representation of event-related information derived from various resources under a common event model.

The Event Abstraction Layer reasons over the incoming events to deduce higher-level concepts. The semantic reasoner processes the incoming RDF graphs after adding them to the ontology. This process is shown in Figure 5-6.

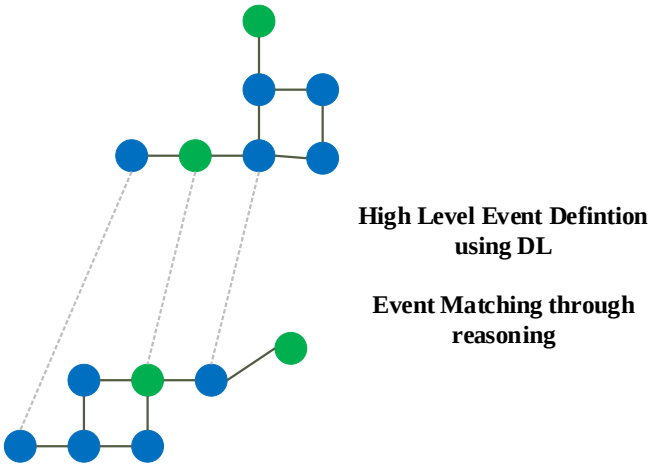

Figure 5-6 Event Definition and Inferred Events

DL reasoning is utilised, alongside the ontological definition of events, to materialise the incoming RDF graphs. If the reasoner, after a realisation step, does infer one of the predefined higher-level events, these higher-level events will be forwarded to the next layer.

The Event Abstraction Layer comprises of two primary elements: The IoT-Ont Ontology and the Semantic Reasoner. A brief introduction of each element is presented in the following sections. Chapter 4 introduced the ontology development for an IoT environment, which is used within the event abstraction layer.

5.4.3 Property Extraction Layer (Description logic event processing)

The Property Extraction Layer is responsible for the identification and extraction of the properties (variables) associated with filtering and joining, from the underlying RDF graph, as specified within the query. The proposed syntax Event Processing Syntax using Description Logic “EPS-DL” allows filter and joins to be specified over the predefined events. However,

in order to perform joins or filters we need to compare the values of the expressed variables in the EPS-DL query. This implies the need to access the higher-level ontological concepts from the underlying RDF graph that EPS-DL targets.

For this reason, an additional SPARQL-querying layer (A querying engine that supports the SPARQL RDF query language) is developed in order to access the higher-level event definition from the underlying RDF graph and to extract the required variables for joining and/or filtering. Figure 5-7 illustrates the property extraction layer with the use of a SPARQL query engine.

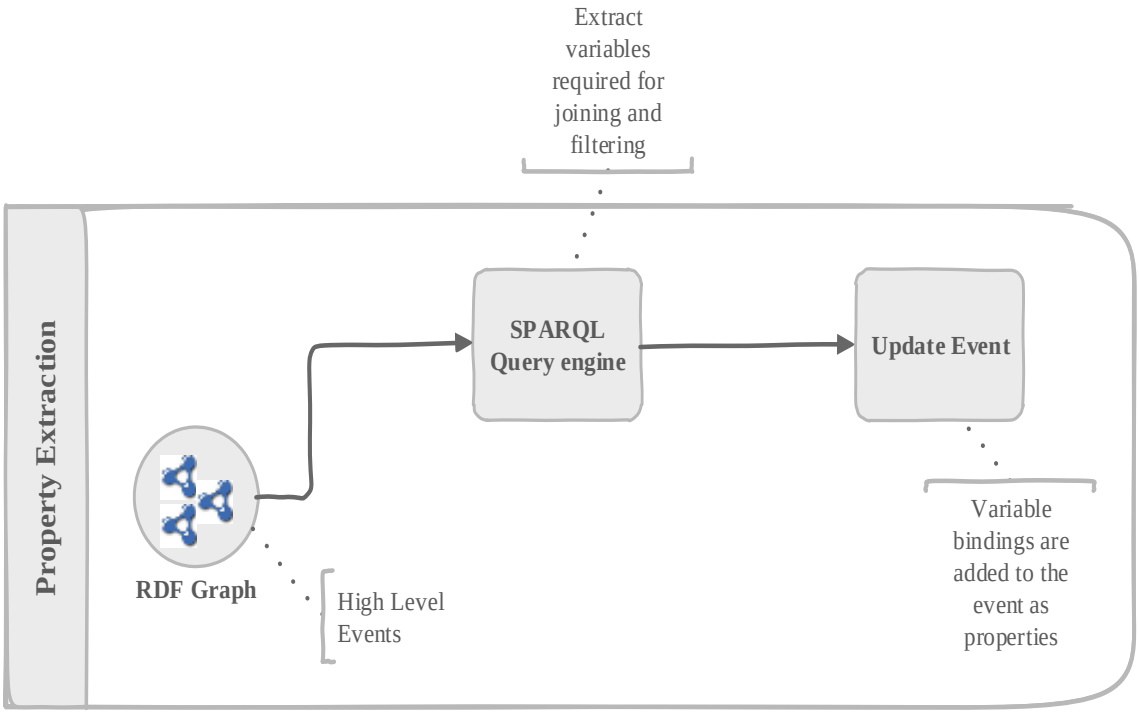

Figure 5-7 Property Extraction Layer

The EPS-DL filters can be simply translated to SPARQL queries. As shown in Listing 5-1.

Listing 5-1 Property extraction based on the translation of SPARQL query

```
SELECT ?temperatureSnsLoc ?v
WHERE { ? temperatureSnsLoc a : Location; : hasValue ?v}
```

Listing 5-1 presents one of the required queries for the property extraction of smoke detection. With regard to joins, for all the events sharing a variable, the value of this variable has to be the same; filters, however, must positively validate the provided condition (e.g. higher than a specific threshold). Listing 5-1 filters against a certain temperature threshold and then joins it to a specific location, in order to detect fire in a specific location. The variable bindings will be added to the event as properties after the execution of the query, while preserving the naming convention. This step can be omitted if there is no need to extract properties and no additional filtering is required.

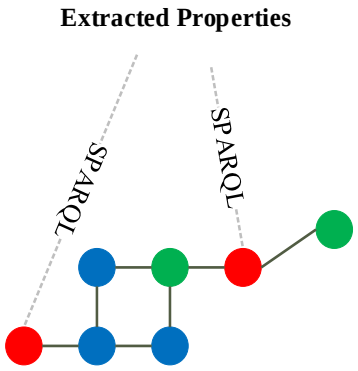

Figure 5-8 Extracting Properties using SPARQL

5.4.4 Event Processing Layer (ESPER)

Time-aware operators are used in Complex Event Processing (CEP) engines to detect patterns over unlimited incoming events sequences. Users specify reaction rules relating to invocating actions in response to situations and events where actions can be made. These rules actually define a pattern on the streaming data, e.g. X preceded-by Y, with the use of a declarative querying language.

The Event Processing Layer is responsible of applying the processing task on the abstracted events. EPS-DL applies filters and joins by using the extracted properties.

This is the last layer in OCEM-IoT architecture and it is the one that actually performs the event processing as illustrated in Figure 5-9 .

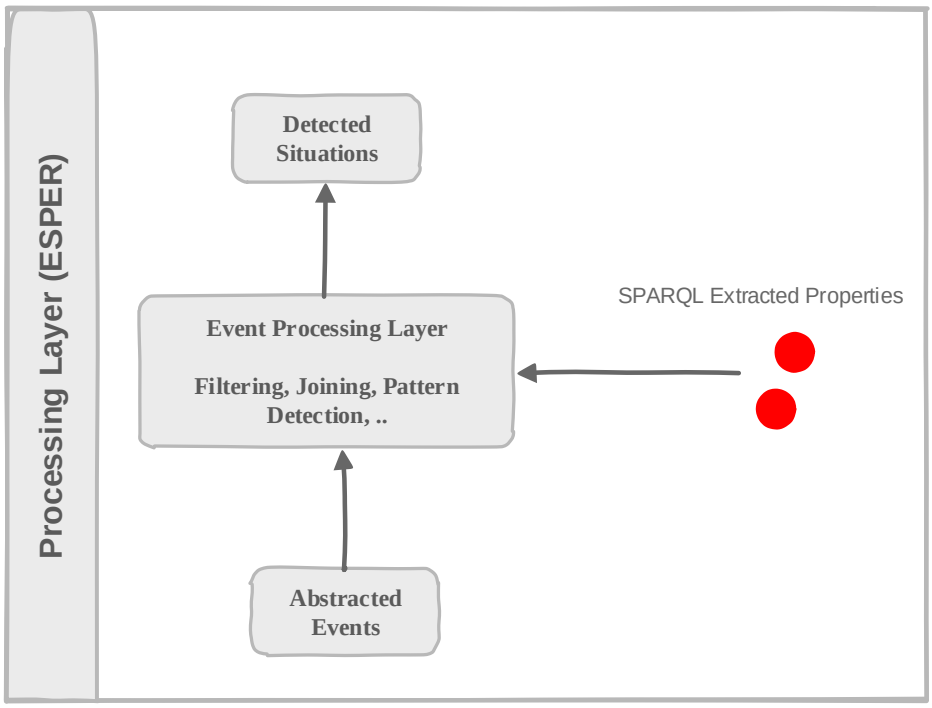

Figure 5-9 Event Processing Layer

The Event Processing Layer uses a complex event processor called ESPER, in order to apply event processing on the abstracted events and to perform the joining and filtering utilising the properties extracted earlier from the Property Extraction Layer.

In this chapter, we presented the overall model by combining the IoT-Ont and the event processing syntax EPS-DL. The ontology was employed within the event abstraction layer alongside the semantic reasoner, which is used to validate it and to infer events. In chapter 6, we provide a comprehensive evaluation of the model's components.

CHAPTER 6: EVALUATION

6.1 Introduction

This Chapter is divided as follows; Section 6.2 presents the IoT-Ont ontology evaluation. Section 6.3 presents the integration of the EPS-DL syntax with ontologies in an event-processing model and an evaluation for OCEM-IoT is presented in Sections 6.3 and 6.4.

6.2 IoT-Ont Evaluation

Once an ontology development process is finished, the evaluation of an ontology is a crucial step towards its application in real use cases. This evaluation involves the assessment of the ontology's usefulness with regard to the purpose it was developed for and the evaluation of the ontology's quality (ontology's conceptual coverage, clarity, etc.). This section presents the different methodologies applied in this research in evaluating the IoT-Ont ontology. This thesis does not claim that the developed IoT-Ont ontology is a complete one since the extensive domain it is meant to represent. It is a version that is meant to evolve and to be build upon by using various ontologies within the IoT domain. IoT-Ont aims to model key concepts and knowledge of the IoT domain into a practical, sharable and extensible ontology.

6.2.1 Introduction to Ontology Evaluation

Before publishing any ontology or building a software applications that rely on ontologies, there is an essential step of evaluating the contents of the ontology (definitions of its concepts, taxonomy and axioms). Evaluating ontologies is not an evidence of the lack of problems, however, it should make it safer to use. The main efforts towards evaluating ontology content were made by (Gómez-Pérez, 2001, Gómez-Pérez et al., 1996) in the framework of

METHONTOLOGY and by (Welty and Guarino, 2001) in the OntoClean method. A survey on evaluation methods and tools can be found in (Gómez-Pérez et al., 2006).

(Vrandečić, 2009) argues that ontology evaluation is an important and worthwhile task, hence it is the very step towards its application in real world scenarios. Omissions and mistakes in ontologies may lead to applications not realising the potential of exchanging data. In addition, ontology evaluation increases the availability and thus reusability of the ontology and decreases maintenance costs. Ontology evaluation assesses the quality of the ontologies and thus encourages their publication and reusability since the re-users' confidence in the quality of such ontologies increases.

According to (Gómez-Pérez et al., 2006) ontology evaluation requires:

- **Verification:** This refers to building the ontology correctly.
- **Validation:** This refers to whether the ontology definitions actually model the domain it was developed for. Ontology validation guarantees that the right ontology was developed. The aim is to show that the world model is compliant with the formal model.
- **Assessment:** This is the human judgment of the ontology; it is focused on judging the ontology from the users' points of view.

A popular ontology evaluation approach is evaluating the ontology with regard to a set of ontology design principles and criteria as it was evaluated in (Obrst et al., 2007, Gómez-Pérez, 2001, Gruber, 1995) (Vrandečić, 2009).

- The coverage of the modelled domain.
- The application and data sources it was developed to address.
- Completeness and consistency.

- Structure, syntax and vocabulary; and the representation language in which it is modelled.

In this thesis, ontology evaluation is based on the criteria identified by (Gómez-Pérez, 2001) this includes completeness, consistency, conciseness, and expandability.

- **Completeness:** All knowledge that is expected to be in the ontology is stated explicitly in it or could be inferred. In other words, how well the ontology covers the real world (IoT domain). Completeness comply to the minimal ontology commitment criteria where the ontology does **not** intend to describe **all** the knowledge involved in a domain, but only the one that is essential to conceptualise the domain.
- **Consistency**: Refers to the absence or existence of contradictory information in the ontology.
- **Conciseness**: Whether the ontology is free from any useless, redundant or unnecessary definitions.
- **Expandability**: Refers to the ability of adding new definitions, without the alteration of already stated semantic. (Modularity)

In this thesis, we distinguish between two types of consistency: the formal consistency and the logical consistency. Verification was held during the ontology implementation where the IoT-Ont ontology was checked for formal consistency by applying a semantic Reasoner during all the steps of the ontology's development, this feature is part of the Protégé software, where semantics reasoned can be applied to detect inconsistencies. Therefore, in this Chapter, consistency refers to the logical consistency.

6.2.2 Selection of Evaluation Methods

Various ontology evaluation approaches were considered in literature based on the type of the ontology being evaluated and the purpose of the evaluation. Brank has classified ontology evaluation approaches as follows: (Brank et al., 2005)

- Golden Standard Approach: Those involve comparing the ontology to a "golden standard" which might be an ontology itself. (Gomez-Perez, 1994, Hovy, 2002)
- Task-Based Approach: Those based on the usage of the ontology in an application and assesses the outcomes. (Porzel and Malaka, 2004)
- Corpus-driven approach: Those involving comparing the ontology with a source of data (e.g. a collection of documents) on the domain being modelled by the ontology. (Lozano-Tello and Gómez-Pérez, 2004)
- Human-Based assessment: Evaluation here is carried out by humans who attempt to assess how well the ontology complies with the predefined requirements, criteria, standards, etc. (Lozano-Tello and Gómez-Pérez, 2004)

The first approach (Golden Standard Approach) is not applicable due to the lack of a "golden standard" or upper-level IoT ontology (SSN, DUL, …etc). The second approach (Task-Based Approach) has been carried out through experimenting using a prototype system alongside the developed EPS-DL, presented in Section 6.3.

The third approach (Corpus-Driven Approach) was held over the course of the ontology's development when the evolving conceptual model was compared to the sources of knowledge, mainly to existing ontologies in IoT.

The fourth approach (Human-Based Approach) includes the usage of the ontology assessment questionnaire which has been distributed amongst Software Engineering, Cloud computing and

sensor network experts to evaluate the quality of the ontology. The Human-based assessment approach is presented in Section 6.2.3

6.2.3 Domain-Expert Evaluation

The evaluation process is conducted based on artefacts produced during the ontology conceptualisation. A general evaluation plan is presented in Table 6-1.

Table 6-1 Evaluation Plan

Evaluation Question	Evaluation Method	Evalauation Criteria
Is IoT-Ont ontology capable of representing the key concepts and its relations in an IoT environment?	Human Assesment (Survey Questionnaire)	• Completeness • Conciseness • Consistency • Expandability

During the evaluation process, the participants, 12 academic researchers and software engineers, received the list of modules and key concepts and their definitions, as well as a Class diagram containing the relationships between the concepts. Table 6-2 represents the modules definitions and Table 6-3 represents the concepts definitions. Both tables were sent to the domain experts.

Table 6-2 IoT-Ont Modules for Evaluation

Module Name	Description
SSN	Input data is described by the SSN/Sensors Module, the main classes of which come from SSN: ssn:Observation; ssn:Sensor. A generic sensor description is offered by ssn:Device together with its associated characteristics (for example, ssn:Deployment; ssn:OperatingRange)
Resources	the Resources Module characterises the system's interaction with physical objects. The main classes of this module are taken from SAN: san:Actuation; san:Actuator. Further, this module reuses some SSN classes, including ssn:Device, which are not specific to sensing. In addition, the Resources Module describes spatial coverage, device status (such as on/off) and RIFD tags.
Time	*OWL Time*[27] describes temporal relationships and properties. It also supports durations as well as time intervals; these characteristics are advantageous when describing complex event specifications as well as inaccurate measurement times
Geo	The WGS84[28] ontology which facilitates the system usage and is flexible enough for more advanced use cases. The WGS84[29] coordinates are in fact the standard model for outdoor scenarios using GPS.
IoT-Ont Domain/Event	The idea behind the event/domain module is the enrichment of the received stream and converting it in a meaningful event according to a specific-domain characteristics, An event could be a simple event such as a high-temperature event in a servers' room, where direct action could be needed as turning on the Air Condition. This is dependant on a domain ontology and is used here only to facilitate possible extension of the ontology. (i.e., Any event source of the system instantiates concepts from the domain ontology. Then, some static information about the event source are stored in the data properties defined by its relative concepts.)

[27] http://www.w3.org/TR/owl-time/
[28] http://www.w3.org/2003/01/geo/

The diagrams sent, contains the modules in Figure 6-1 and the concepts and relationships in Figure 6-2 are presented below.

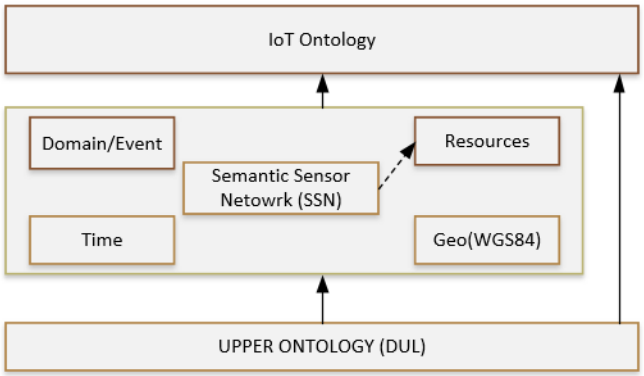

Figure 6-1 IoT-Ont Modules

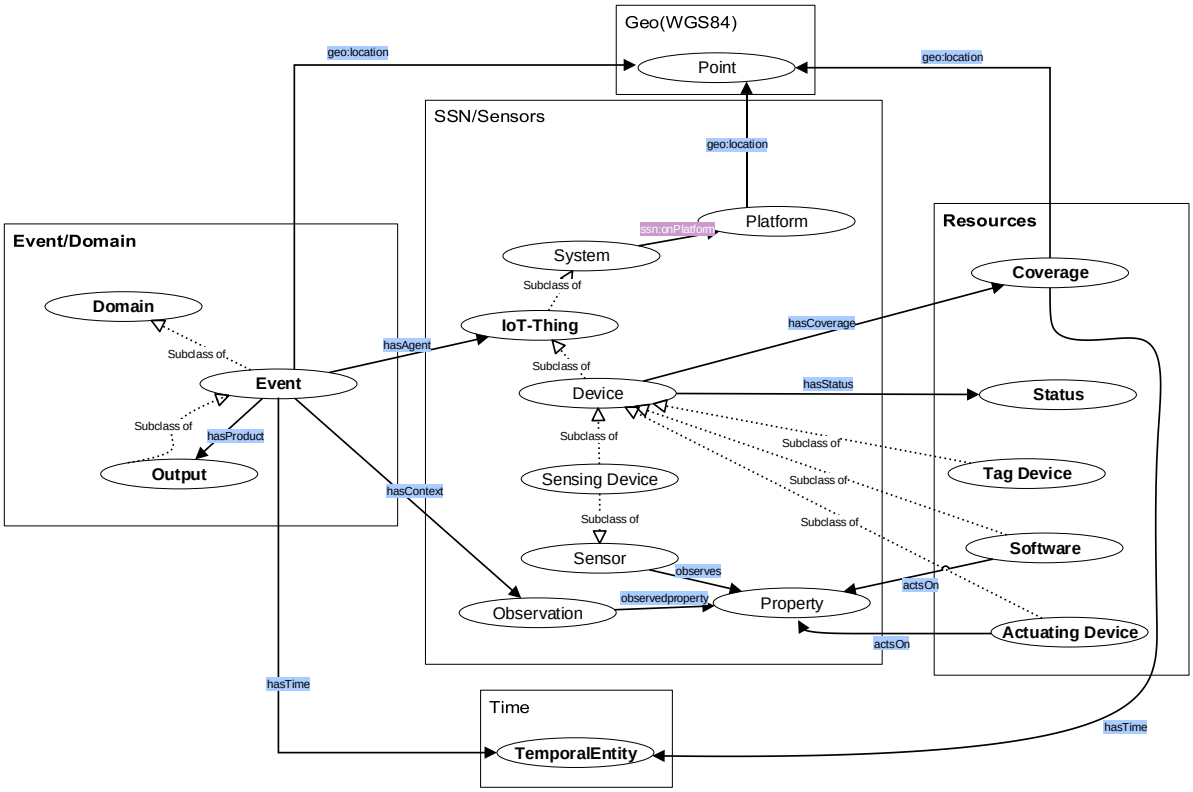

Figure 6-2 IoT-Ont Key Concepts and relationships

Table 6-3 IoT-Ont Glossary for Evaluation

Concept Name	Description
IoT-Ont Thing	The concept Iot-Ont Thing enables us to define events based on their features in the context of smart environments and relate it to one of the actors in an IoT environment. These actors are defined using a subclass of IoT-Thing.
IoT-Ont Device	IoT-Ont Device is based on the ssn:Device which describes an abstract device and inherits all the properties of the class ssn:System (subcomponents, platform to which a system is attached, deployment in which a system participates, operating and survival range). In IoT-Ont ssn:Device is categorised in four sub-classes SensingDevice, ActuatingDevice, TagDevice and Software. All physical sensor devices are represented by the class ssn:SensingDevice in the ontology. Instances of this class possess all properties of the classes ssn:Sensor and ssn:Device.
IoT-Ont temporal entity	The pattern Iot-Ont Temporal Entity represents temporal intervals between any two timestamps through an observation process. It also could represent any instant of time. The two main classes of this pattern are IoT-Ont time:Interval (expresses an interval between two specific time-date values) and Iot-Ont time:instant (expresses a temporal entity with zero extent or duration), which are subsumed by the class time:temporalentity.
IoT-Ont Point	IoT-Ont follow a practical approach of using a simple representation of location using GPS. This is based on the class WGS84:Point, whose members are points. Points can be described using the 'lat', 'long' and 'alt' properties, as well as with other RDF properties defined elsewhere. For example, we might use an externally defined property such as 'bornNear' or 'withinFiveMilesFrom', or perhaps other properties for representing lat/long/alt in non-WGS84 systems. The 'lat' and 'long' properties take literal (i.e. textual values), each in decimal degrees. The 'alt' property is decimal metres about local reference ellipsoid.
IoT-Ont Sensing Device	Sensing Device is a subclass of SSN:Device and represents an IoT-Ont Thing that implements sensing.
IoT-Ont Sensor	This concepts is based on SSN:sensor it is an IoT-Ont Thing that can do (implements) sensing. A sensor is any entity that can follow a sensing method and thus observe some Property of a FeatureOfInterest. Sensors may be physical devices, computational methods, a laboratory setup with a person following a method, or any other thing that can follow a Sensing Method to observe a Property.
IoT-Ont Actuating Device	Represents a SSN:Device that can actuate over an object. It is a subclass of SSN:Device which in turn is a subclass of IoT-Ont Thing.

IoT-Ont Tag Device	Tag is a subclass of SSN:Device and could be a QR code, RFID or barcode.
IoT-Ont Software	IoT-Thing Software is a subclass of SSN:device and represents a piece of software or a service that runs in an IoT node. Can also be an abstracted device, issued from the composition of physical devices and abstract processing.
IoT-Ont Coverage	Represents the 'physical space' covered by an IoT-Ont Thing (i.e. a temperature sensor inside a room has a coverage of that room).
IoT-Ont Status	Represents the current status of an IoT-Ont Thing. Currently, this is limited only to represent whether the IoT-Ont Thing is on/off, this could be extended in the future.
IoT-Ont Property	Property refers to a feature of an IoT-Thing that is the interest of an observation process in an IoT environment. These properties differ based on the category of objects (e.g., mobile or non-mobile objects) besides the measurement criteria (e.g., location of objects) depending on which the observation process is conducted. The concept Iot-Ont Property is used to more tightly couple the representation of IoT-Ont Thing with some features or properties, which are measurable by sensors.
IoT-Ont Event	The IoT-Ont Event concept represents a bridge between the IoT-Ont and a domain-specific ontology that could be easily added to the ontology.
IoT-Ont Domain	A domain is a concept that represents one of the domain applications in an IoT environment(Agriculture, HealthCare, Military, etc.). The main usage of this class is to provide more context to the observed data and thus provide data enrichment based on data properties.
IoT-Ont Observation	An DUL:Observation is a Situation in which a Sensing method has been used to estimate or calculate a value of a Property of a FeatureOfInterest. Links to Sensing and Sensor describe what made the Observation and how; links to Property and Feature detail what was sensed; the result is the output of a Sensor; other metadata details times and so forth.

6.2.4 Structure of the Survey

The structure of the ontology evaluation is based on the criteria identified by (Gómez-Pérez, 2001) such as consistency, conciseness, completeness and expandability, which is the most cited research for ontology evaluation. The IoT-Ont evaluation survey was divided into seven sections as follows:

1. **Introduction**: The goals of the evaluation were presented to the participants, and an option to receive the evaluation results was given.
2. **(Conciseness) Description of Modules:** The participants evaluated the description of the modules present in Table 6.2 and Figure 6.1. If the participants strongly disagreed or disagreed with any description, they could indicate which concepts should have the description changed. For that, a list of all modules were presented with checkboxes to select the ones that had to be reviewed. At the end of the section, a space for additional comments was given to the participant.
3. **(Conciseness) Description of Concepts:** The participants evaluated the description of the terms present in the Table 6.3 and illustrated in Figure 6.2. If the participants strongly disagreed or disagreed with any description, they could indicate which concepts should have the description changed. For that, a list of all concepts was presented with checkboxes to select the ones that had to be reviewed. At the end of the section, a space for additional comments was given to the participant.
4. **Completeness represented by Ontology Coverage:** The participants evaluated if the concepts in the ontology correctly represented the Internet of Things domain. If the participants strongly disagreed or disagreed with any concept, they could indicate which concepts should have been corrected in the ontology. For that, a list of all concepts was presented with checkboxes to select the ones that had to be reviewed. At the end of the section, a space for additional comments was given to the participant.
5. **(Consistency) Relationships:** The participants evaluated if the relationships between concepts were correctly mapped. If the participants strongly disagreed or disagreed with any concept, they could indicate which relationships should have been corrected in the ontology. For that, a list of the relationships was presented with checkboxes to select

the ones that had to be reviewed. At the end of the section, a space for additional comments was given to the participant.

6. **Expandability (Modularity):** New terms and modules could be added without having to revise the existing structure of the ontological model. This refers to the capacity of re-using IoT-Ont or part of its modules to build a domain specific ontology within IoT.
7. **Additional Information:** The participants could give their option about the ontology and explain how he or she was planning to use the ontology. Finally, a text area was provided to the participant so that they could give any additional comment regarding the ontology or the evaluation itself.

A summary of survey sections, questions, and possible answers is presented in Table 6-4. The complete survey structure can be found in Appendix A alongside the ethical approval needed for this type of research in Appendix B.

Table 6-4 Survey Questions

Quality Criteria	Question/Statement	Possible Answers	Comments
Introduction	*"This evaluation survey is aimed at expert researchers in this field and on software engineers"* *The purpose of this survey is to evaluate the modules and concepts represented and how they are related in IoT-Ont, an ontology for IoT.* *The survey contains 4 pre-survey questions regarding the participants expertise and 8 ontology evaluation questions, and your participation is anonymous. If you want to be informed about the final result, please enter a valid e-mail address below.".*	Enter E-mail	N/A
Ontology Coverage	*"All relevant concepts related to the Internet of Things domain have been represented in the ontology.".*	Strongly Disagree, Disagree, Undecided, Agree, Strongly Agree	*"If you strongly disagree or disagree, please explain which concepts should be corrected or included in the ontology."*
Description of Modules	*"The ontology correctly describes all key modules related to Internet Of Things.".*	Strongly Disagree,	*If you strongly*

		Disagree, Undecided, Agree, Strongly Agree	*disagree or disagree with any description, please explain why."*
Description of Concepts	*"The ontology correctly describes all concepts related to Internet Of Things.".*	Strongly Disagree Disagree Undecided Agree Strongly Agree	*If you strongly disagree or disagree with any description, please explain why."*
Relationships	*"All relevant relationships related to the Internet of Things concepts have been correctly mapped in the ontology."*	Strongly Disagree, Disagree, Undecided, Agree, Strongly Agree	*"If you strongly disagree or disagree, please explain which relationships should be corrected or included in the ontology."*
Expandability	"New terms and modules can be introduced without the need to revise the existing structure of the ontological model."	Strongly Disagree, Disagree, Undecided, Agree, Strongly Agree	*"If you strongly disagree or disagree, please explain what design issue makes it difficult to expand the current model."*
Additional Information	*"In your opinion, is an ontology for Internet of Things necessary?"*	Yes, No	*If not, please elaborate your answer.*
Additional Information	*"Please use this space for any additional comments or suggestions in regards to IoT-Ont."*	Free text	N/A

The scale adopted for evaluation is of 1-5, where 5 = strongly agree and 1 = strongly disagree, to validate the following criteria, this is based on the evaluation scale proposed by (Gruber, 1995, Obrst et al., 2007, Vrandečić, 2009).

6.2.5 Results

It was challenging step to collect responses from domain experts due to the limited number of experts in Software Engineering who are welling to participate and in particular those with experience in the IoT domain. It took more than 4 months to gather 12 response. The problem

of limited number of participants faces many researchers in their ontology evaluation process (García et al., 2006) (Al-Yahya, 2005). However, even though the sample is small in size, it is considered acceptable to judge domain ontology. (Velardi et al., 2005)

The participants invited to participate in the survey questionnaire were computer scientists and software engineers with interests mainly in IoT, Data knowledge representation and cloud computing, and PhD graduates in software engineering and networking. Table presents the participants' educational backgrounds and their current technology interests and expertise.

Table 6-5 Participants' Backgrounds and Technology of Interest

	Educational Background	**Technology Interest/ Work Experience**
Person 1	PhD in Data Science	IoT
Person 2	PhD in Software Engineering	Software Reference Architectures
Person 3	PhD in Computer Science	Cloud Computing
Person 4	PhD in Software Engineering	Data knowledge representation
Person 5	MSc Robotics	Robotics and Sensors
Person 6	MSc Artificial Intelligence	Speech Recognition
Person 7	MSc Artificial Intelligence	Human to Machine Interface
Person 8	MSc Software Engineering	Cloud Computing
Person 9	BSc Computer Science	Cloud Computing
Person 10	BSc Computer Science	Data Modelling in Cloud Computing
Person 11	BSc Computer Science	Data Modelling in Cloud Computing
Person 12	BSc Computer Science	IoT

After receiving the ontology artefacts and modules, the architectural model and the survey questionnaire, the participants could analyse the content provided to answer the survey and give their opinion. The results of the evaluation are presented and discussed according to the survey sections.

Results: Description of Modules

In Figure 6-3, a summary of the evaluation regarding the description of modules is presented. 67% of the participants said that the ontology described correctly the key modules in an IoT environment, 8% strongly agreed to the modules descriptions, and 25% of the researchers declared to be undecided about the descriptions.

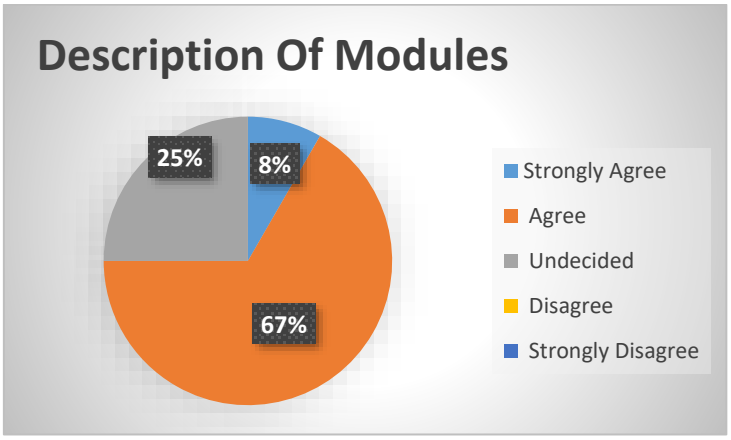

Figure 6-3 Experts' Opinions on Modules Description

Results: Description of Concepts

In Figure 6-4, a summary of the evaluation regarding the description of concepts is presented. 83% of the researchers have agreed or strongly agreed to the description of concepts provided. 17% were undecided about the descriptions.

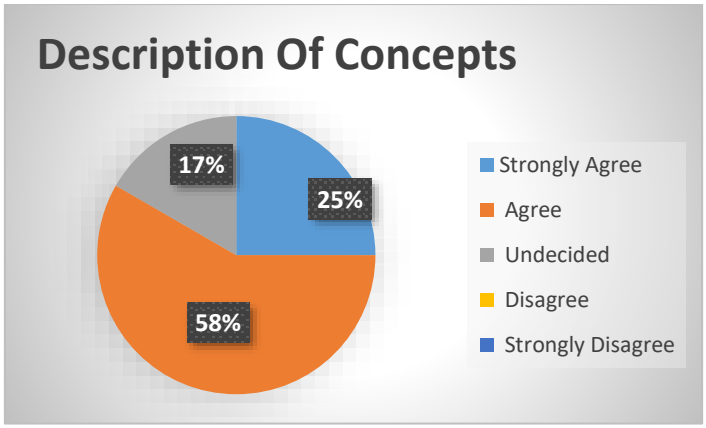

Figure 6-4 Experts' Opinions on Concepts Description

Results: Ontology Coverage

In Figure 6-5, a summary of the evaluation regarding the ontology coverage is presented. 75% of the participants said that all relevant concepts related to IoT domain were presented in the ontology, 25% of the participants have declared undecided.

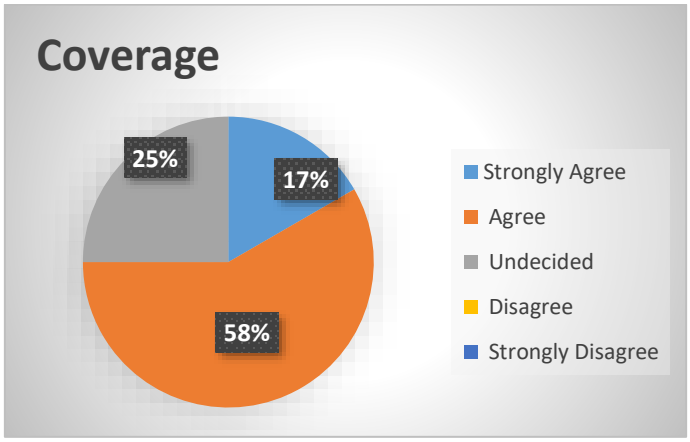

Figure 6-5 Experts' Opinions on Ontology Coverage

Results: Ontology Relationships

In Figure 6-6, a summary of the evaluation regarding the ontology relationships is presented. 83% said that all relevant relationships related to IoT concepts were correctly mapped in the ontology, while 17% of the researchers were undecided.

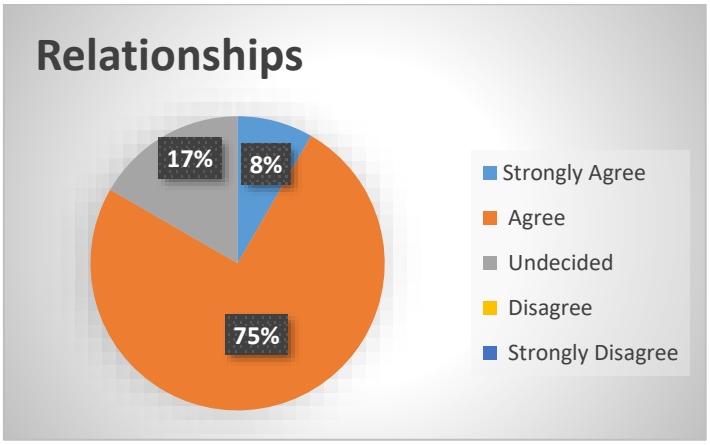

Figure 6-6 Experts Opinion On Ontology Relationships

Results: Ontology Expandability

In Figure 6-7, a summary of the evaluation regarding the ontology exapandability is presented. 83% of the participants have declared that the ontology is exapandable, 17% of which has strongly agreed. 16% of the participants were undecided.

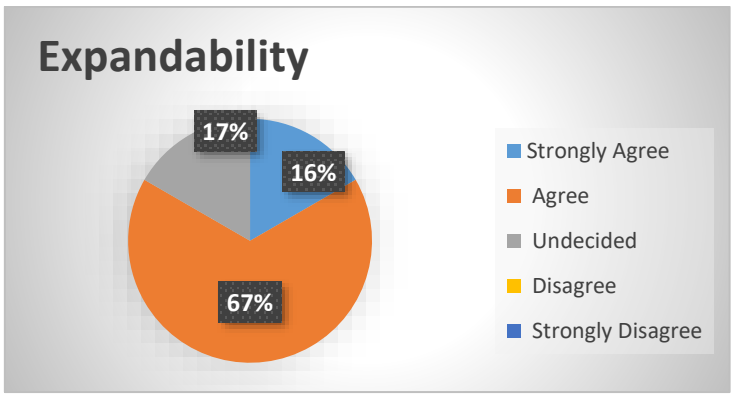

Figure 6-7 Experts' Opinions on Ontology Expandability

Results: Additional Information

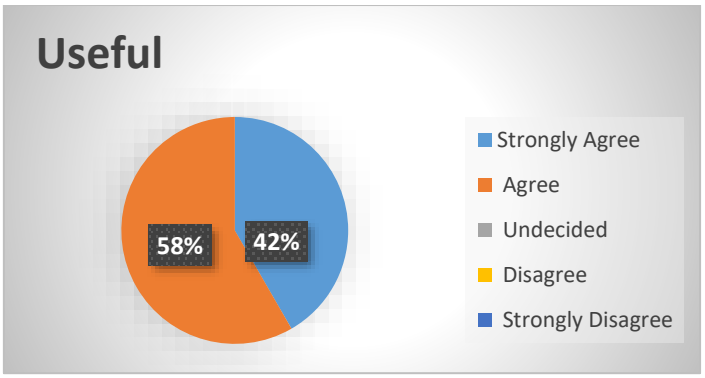

Figure 6-8 Experts' opinion on the usefulness of the ontology

The last parameter was a simple question whether the domain expert that this ontology could be useful in an IoT domain and whether, all the participants have acknowledged it as useful.

In order to validate the results obtained from the survey questionnaire, the SPSS software was used to check the integrity and the reliability of the results. Cronbach's alpha index is used to examine the internal consistency of the results (Santos, 1999), as this measurement is commonly used to verify and analyse the reliability of Likert-type question results, the preferred alpha index value is > 0.7. (Creswell, 2009, Shull et al., 2007).

Reliability Statistics

Cronbach's Alpha	N of Items
.714	6

Figure 6-9 Cronbach Alpha Result (6 Questions feedback)

The calculated Cronbach's alpha index for our results is 0.714 based on the 6 questions presented to the participants, this could be considered a good result, reflecting high inter-correlated results; Figure 6-9 shows the Cronbach's Alpha result for the dataset provided by the participants.

6.3 EPS-DL Evaluation

6.3.1 Introduction

An experimental evaluation of EPS-DL is also carried out to improve its functioning capability for event processing in an IoT environment. In order to do this, we followed the procedure used by researchers and developers of existing event processing languages (C-SPARQL, EP-SPARQL, MASSIF, SPARQLstream, CQELS) (Zhang et al., 2012) by using a set of queries developed by the World Wide Web Consortium[30] especially for stream reasoning evaluation (called the SR BENCH). The queries provided by the SR Bench aims to assess the capabilities of RDF/SPARQL stream engines with regard to essential features from Semantic Web research areas combined in one read-world application scenario. This indicates the capabilities of a system to handle a wide variety of distinct types of queries by using Semantic Web technologies, where querying, sharing, reasoning and interlinking, are applied on highly dynamic streaming RDF data. This benchmark was developed to aid both users and researchers to compare RDF/SPARQL stream engines in common application scenarios used in our daily life, i.e., querying and deriving information from weather stations.

The metrics used to evaluate EPS-DL are the performance metrics based on the evaluation process (Throughput and Latency), followed by existing event processing languages. (C-SPARQL, EP-SPARQL, MASSIF, SPARQLstream, and CQELS). This is illustrated in more details in Section 6.3.4.

6.3.2 Evaluation Using the SRBench

As we have seen in the literature, there is a growing interest in RDF/SPARQL streaming engines, though each one of them has its specific extensions set and very distinct approaches

[30] https://www.w3.org/wiki/SRBench

of implementation. Therefore, for the purpose of comparing the performance and functionality of these systems, a streaming RDF/SPARQL Benchmark (SRBench) was first developed in 2012 this benchmark has served as a reference for testing and has since been continuously developing (Zhang et al., 2012). To our knowledge there have not been produced any other benchmark for streaming data engines in the literature apart from the SRBench.

The SRBench is a streaming RDF/SPARQL benchmark, which evaluates the capabilities of streaming RDF engines when addressing diverse query types that applies Semantic Web technologies such as querying, sharing, interlinking, and reasoning, on highly dynamic RDF data streaming. SRBench refers to a general-purpose benchmark designed mainly for comparing RDF/SPARQL streaming engines. The design of SRBench was founded on the principle of dealing with the following challenges: The necessity of an appropriate streaming RDF dataset, the lack of a single SPARQL-based streaming query language and the definition of a specific set of features. (Calbimonte, 2013)

This section introduces the SRBench, which will be used for both the functional evaluation of EPS-DL in section 6.3.3, and the performance evaluation in Section 6.3.4.

Benchmark Data set. The design of a streaming RDF/SPARQL benchmark), a streaming RDF/SPARQL benchmark's design necessitates a data set which is selected cautiously and is realistic, relevant, interlinkable, and semantically valid (Zhang et al., 2012). Further, the data set must enable the queries to be formulated such that they feel natural and put forth a brief yet complete set of challenges which should be met by streaming RDF engines.

A concise set of features. The application of Semantic Web technologies on streaming data can provide explicit semantics to data for searching and reusing, enabling reasoning using ontologies, and facilitating the integration with other data sets. A comprehensive query set is provided by the benchmark, which evaluates the ability of a system to process such distinctive

features on highly dynamic streaming data (considering the arrival rate and also the amount of streamed data), possibly along with static data. (Calbimonte, 2013)

The SRBench has provided the definition of 17 queries which have been created and selected so that they provide important insights that are applicable to RDF streaming systems, and are useful in many domains, e.g., notion of time bounded queries (e.g., data in the latest X time-units); notion of continuous queries (i.e., queries evaluated periodically); data summarisation in the queries (e.g., aggregates); using raw-data to provide high abstractions ; and combining streams with contextual static data. (Zhang et al., 2012)

No standard query language. Standards have not yet been developed for streaming data processing, nor for streaming SPARQL extensions. Hence, in a streaming benchmark, it is important to specify the queries in a language agnostic way, while adhering to a precise semantics. This challennge was addressed by the SRBench, since it provided a descriptive definition for each query in the benchmark. (Zhang et al., 2012)

SRBench Queries

The queries defined by the SRBench[31] include diverse features from RDF and SPARQL processing, more advanced SPARQL 1.1 features, interlinking with external static data, and inherent streaming data features included in CEP. The features taken into consideration are discussed below.

Graph pattern matching. Is a significant feature of SPARQL queries, it comprises of features such as the basic graph pattern matching operators '.' (representing a natural join AND) as well as FILTER, along with the most complicated operators UNION and OPTIONAL (Arenas and Pérez, 2012) SRBench queries includes all these operators.

[31] https://www.w3.org/wiki/SRBench

Query forms. SPARQL's three major query forms are supported by the SRBench including SELECT, CONSTRUCT, and ASK. DESCRIBE, which is another query form, returns an RDF graph that represents the resources found. This form is not utilised in the SRBench queries as it is largely dependent on implementation, which highly complicates the verification of the query results. (Zhang et al., 2012)

SPARQL 1.1. The additions to the SPARQL language included in the SPARQL 1.1 W3C Proposed Recommendation presented various new characteristics that includes; subqueries, negation, aggregates, Property Paths projection expressions and assignment.

Reasoning. SRBench involves queries which enable exploiting reasoning if it is provided by the processing engine. Presently, the queries include reasoning over the rdfs:subClassOf, rdfs:subPropertyOf and owl:sameAs properties. Both systems can implement and execute queries provided by SRBench, with or without mechanisms for inference, while the differences can be noticed in the results of the query. Despite that SPARQL is not a reasoning language, it can be utilised for querying ontologies if they are RDF-encoded. Consequently, for systems that lack reasoning, such obstacle can be addressed by expressing explicitly some reasoning tasks through extra graph patterns with Property Path over the ontologies. (Calbimonte, 2013)

Streaming. Current streaming extensions of SPARQL tend to utilise streaming data operators inspired by DSMS and CEP continuous query languages such as CQL.

6.3.3 Functional Evaluation

RDF/SPARQL stream query engines are new and are mainly in their early development phase. A functional evaluation of these systems is conducted in this section, aiming at finding out if the functionalities provided by these systems are sufficient for realistic streaming query

processing, if there are any missing key characteristics or if there are different operators that differentiate one system from another.

The evaluation proposed in this research comprises of four systems: OCEM-IoT, C-SPARQL, CQELS, and SPARQLstream. These models were chosen since they are widely cited in literature and they provide implementations and results within the SRBench. The implementing queries are available in the SRBench page[32]. The evaluation results are shown in Table 6-6.

Table 6-6 Evaluation Results

Model	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17
OCEM-IoT	X	X	X	X	X	X	X	X	X	X	X	F	F	F	F	F	F
C-SPARQL	X	F	F	X	X	X	F	X	F	X	X	F	F	F	F	F	F
CQELS	X	F	F	X	X	X	F	X	F	X	X	F	F	F	F	F	F
SPARQLstream	X	F	F	F	F	X	X	F	F	F	F	F	F	F	F	F	F

The X indicates that the model is capable of processing a specific query. If a certain model fails to process a particular query, the F letter refers to its failure. The results of the evaluation demonstrate that most fundamental SPARQL features are supported by all systems i.e., (Q1) The CONSTRUCT and SELECT modifiers used for graph pattern matching. Interestingly, neither C-SPARQL nor CQELS nor SPARQLstream support a simple solution modifier such as ASK (Q3), however this is covered by the OCEM-IoT (Q3).

Notably, the property paths is not sufficiently supported by any of the implementations (Q12-Q17), and thus these models fail to execute 6 queries implemented in SRBench. The Property Path expressions is an important feature of SPARQL, since it offers flexible methods of navigating through RDF graphs and makes it easier to reason over different graph patterns. For future work, this should addressed.

[32] https://www.w3.org/wiki/SRBench

Finally, the current implementations provide very little support for the reasoning process. CQELS does not currently tackle the reasoning problem at all. C-SPARQL supports little reasoning based on simple RDF entailment; however, this is not included in the published implementation. EPS-DL has managed to provide reasoning for (Q2), however it failed for the queries that had complex property paths (Q12-Q17). The main conclusion of this evaluation is that there is yet no best single system available, and even though the EPS-DL model does support few more queries than C-SPARQL, SPARQLstream , and CQELS, all of them still need covering features of SPARQL 1.1. Overall, we have shown that EPS-DL and the other approaches are capable of querying streaming data, but there is still a big gap to be filled in terms of the implementation.

6.3.4 Performance Evaluation

Throughput and Latency are the basic performance metrics used for evaluation of such models (Benarbia et al., 2018, Anicic, 2012). As presented in (Etzion et al., 2011b), for these two metrics there are various definitions. For instance, throughput could be measured in the following ways: input throughput (measures the number of input events that the system can "digest" within a given time interval), processing throughput (measures the total processing times divided by the number of event processed within a given time interval), and output throughput (measures the number of events that were emitted to event consumers within a given time interval). For testing the input throughput was adopted in this work, where the percentage of event instances that are processed in patterns is high.

With regard to latency, the latency for each event is measured. For events that don't create derived events directly, the time that the system takes to finish processing them is measured (Etzion et al., 2011b).

To evaluate the OCEM-IoT approach, EPS-DL was implemented in Java using Eclipse. The test cases presented subsequently were performed on a workstation with Intel(R) Core i5-4210M, 2.6GHz, 4GB RAM, running on Windows 10. The RDF knowledge base is loaded automatically into Java by the query engine, which also compiles RDF stream triples into Java.

To run tests, an event stream generator was implemented, which provides random time series data. Two queries were used to evaluate the performance, which is calculated using the two metrics mentioned above (Throughput and latency). Query 1 and Query 6 from the SRBench were used for this test since all the models studied were capable of performing these queries.

Throughput and Latency Comparison

Figure 6-10 shows the throughput results comparison between the three models, the Sparql-stream provides better throughput in these two queries, however Sparqlstram provides no reasoning at all. Both C-Sparql and EPS-DL supports reasoning, EPS-DL has improved results over C-SPARQL.

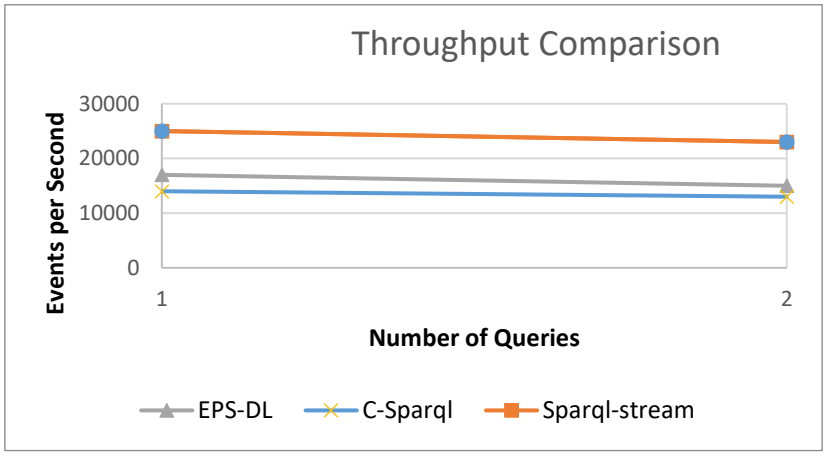

Figure 6-10 Throughput Comparison

Figure 6-11 shows the latency results comparison between the three models, again Sparql-stream provides better results, with minimum latency even when the number of inputted events

increase, however Sparql-stream provides no reasoning capabilities yet. EPS-DL again has improved results over C-Sparql.

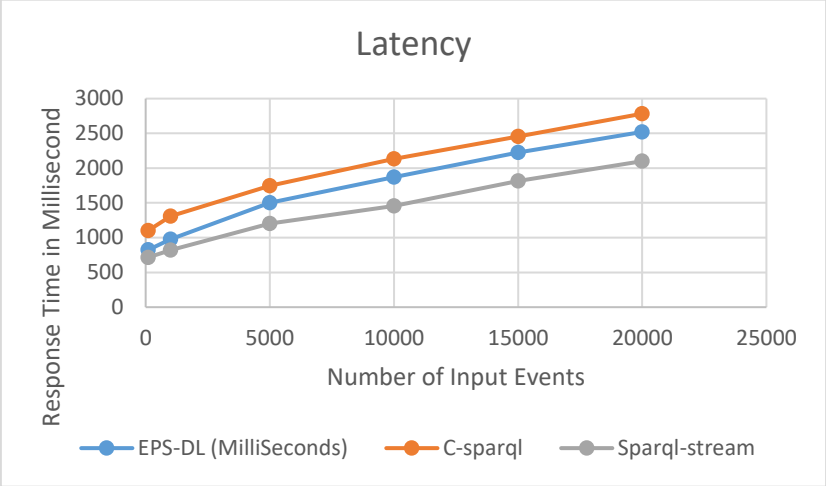

Figure 6-11 Latency Comparison

6.4 Comparison of Relevant Models

Table 6-7 summarises the similarities and differences between the related works and the OCEM-IoT model. The table highlights the contribution of the model through the requirements shown in Section 4.4. OCEM-IoT merges the semantic event declaration and with processing. Moreover, it supports computing temporal inference over the higher-level concepts produced by a deductive reasoning method. This model supports all the traditional event-processing operators, whereas state of the art approaches support some of these operators. For example, EP-SPAQL, which is one of the most cited modesl, supports only one event processing operator (The sequence operator SEQ), and the MASSIF model supports none of the event processing operators at all.

One of the studied models (ODECP) does not support filters, whereas joins are not supported by two of the studied models (MASSIF AND ODECP). The approach includes optional time aware modifiers like also and not, which enables to define more expressive patterns.

Table 6-7 Comparison of existing CEP Models

	Semantic Event Declaration	**Inference**	**Event processing Operators**	**Filters**	**Joins**
EPL	Relational	None	All	Yes	Yes
MASSIF (Bonte et al., 2017)	DL Axioms	OWL 2DL	None	Yes	No
EP-SPARQL (Barbieri et al., 2010)	RDF BGP	RDFs	SEQ	Yes	Yes
C-SPARQL (Barbieri et al., 2010)	RDF	RDFs	WINDOW	Yes	Yes
ODECP (Taylor and Leidinger, 2011)	OWL	RDFs	SEQ,AND,OR	No	No
OCEM-IoT	DL Axioms	OWL 2DL	All	Yes	Yes

In this chapter we have presented the evaluation method used for the two components of the OCEM-IoT model, the IoT-Ont has received positive feedback from the reviewers, and will be published online to allow collaboration on a generic IoT ontology. The EPS-DL syntax has advances over many languages used in event processing, it implements all the time operators used in complex event processing engines, and it has succeeded in running more queries in the SRBench, than the most cited languages in this field, moreover it has good throughput and latency results, in the running example using Q1 and Q6 of the SRbench, EPS-DL has improved results over one of the most cited languages for stream reasoning.

OCEM-IoT can play an important role in a weather monitoring and actuating centre, where big amounts of data can be enriched with relevant weather ontologies alongside the core IoT-Ont, and then providing an analysis of the current irrigation needs by implementing the relevant

EPS-DL queries on the high level concepts and events. Another domain which could benefit efficiently from OCEM-IoT is a wearable health device, which for example in blood pressure patients, can read the blood pressure, heart rate and temperature of the patient and then by analysing and processing the readings triggers the patient to act either by taking the required medication or seeking medical attention.

However, this is merely an initial step towards an event processing system based on ontological concepts, and lots of work should be done in this field. Chapter 7 provides the conclusions and future research work.

CHAPTER 7: CONCLUSIONS AND FUTURE WORK

7.1 Introduction

This thesis originates from the research on ontology-based event query techniques, the extensions for sensor data streams operators and the use of description logic to capture data and extract relevant properties sources. It aims to make sense of the raw sensor data in real time of multiple, heterogeneous, gigantic and inevitably noisy sensor data streams.

This final chapter presents a summary of the work and describes the key contributions. It also illustrates the possible areas for future work. The thesis contributed by developing OCEM-IoT, a model for accessing, enriching and querying data streams from heterogeneous sources, by the use of ontologies for representing the data captured by sensors. Achievements of this work include the development of an ontology for IoT (IoT-Ont) and validation of its modules, concepts and relationships. In addition, the proposal of an architecture to bridge the gap between sensor data and event processing applications. OCEM-IoT supports all the traditional event-processing operators, whereas state of the art approaches support some of these operators. OCEM-IoT performs better with regard to latency and throughput, than one of the most cited models in literature C-SPARQL.

OCEM-IoT tackled the research questions in section 1.2, by developing the IoT-Ont ontology, which represents the semantic properties of sensor data and enrich them with contect, such as time and location. Moreover OCEM-IoT implements an event processing syntax which can make use of the ontology as a data model to continuously query sensor data.

In this chapter, the objectives of the thesis will be revised and the means of realising them will be illustrated in Section 7.1. Section 7.2 presents the ideas and suggestions for the future development.

7.2 Review of the Research Objectives

This section introduces the research objectives and reviews the means of achieving them.

7.2.1 Objective 1

- **Comparison of existing IoT ontologies:**

In order to achieve this objective, existing ontologies for IoT have been researched and compared against requirements for devising an IoT ontology. This has been presented in Section 2.5.4 and in Chapter 3. A comparison table was provided by tables 3-1 and 3-2. The review has concluded that all the existing ontologies proposed cover only some of the necessary requirements, they are incapable of providing a comprehensive solution to the problems of heterogeneity and interoperability in IoT technologies

7.2.2 Objective 2

- **Review of current methodologies in software engineering addressing the ontology development process.**

This objective aims to study the methodologies used for developing ontologies. This objective has been carried out comprehensively in Chapter 3. The review showed that all the methodologies provide general guidelines for the development process, it also showed that competency questions is one of the most successful and cited techniques in literature, this was followed later on the development process.

7.2.3 Objective 3

- **An architectural model: An ontology based event processing model**

This objective aims at developing a model that interconnects the sensor data with event processing applications (such as complex event processing engines). This objective was achieved in Chapter 5. The model bridges the gap between semantic event representation of high-level concepts and its employment in complex event processing.

7.2.4 Objective 4

- **Development of an Ontology for IoT.**

To achieve this objective, after carrying a review on the existing IoT ontologies, conceptual and functional requirements were gathered in Chapter 4, this process was based on the-state-of-the-art ontologies and upper ontologies (such as DUL and SSN), we defined the basic modules and concepts of the ontology. This illustrated in details in Chapter 4.

7.2.5 Objective 5

- **A time-aware event processing syntax based on description logic to facilitate event processing (EPS-DL).**

To achieve this objective, we implemented a simple syntax based on the SPARQL language, this syntax uses description logic for reasoning over events and implements the time operators and modifiers used in complex event processing. The development process of this syntax is explained in details in Chapter 4.

7.2.6 Objective 6

- Evaluating the proposed ontology using a survey questionnaire.

To achieve this objective, we have carried out a survey questionnaire with domain experts. Results obtained from the participants provided evidence that the ontology does represent the IoT environment by providing a modular design which is easy to be reused and built upon.

7.2.7 Objective 7

- Evaluating the event processing syntax EPS-DL

To achieve this objective, we first compared the syntax to the-state-of-the-art languages used for stream reasoning and event processing, the comparison was based on the development requirements presented in Section 4.4, and the comparison Table 4-3 is available in section 4.6. The proposed syntax has also been evaluated using both functional and performance evaluation. The results showed the advances of EPS-DL against other languages, it provided evidence that EPS-DL can run more queries in the SRBench than other languages and provided improved throughput and lower latency than the well cited language C-Sparql.

7.3 Future Work

The research opportunities that emerged during the work on this thesis represent perspectives of future research that can contribute to the areas of IoT and event processing. In the future, we plan to:

1. IoT-Ont ontology should be evaluated within an industrial environment, in order to provide more evidence of its success.
2. IoT-Ont ontology should be published online in order to allow collaboration development, since the IoT field is very complex, this should be a contant dynamic process. The ontology could be applied to industrial use cases within an IoT environment, such as smart cities and smart architecture.

3. IoT-Ont could be extended to various domains within IoT, in order to better capture the specific domain knowledge, such as agriculture, health, etc.
4. Further test EPS-DL against the queries it failed to run in the SRBench, and work towards its success.
5. Implement OCEM-IoT within a specific use cases and evaluate it in comparison with existing models.

REFERENCES

ABADI, D. J., CARNEY, D., CETINTEMEL, U., CHERNIACK, M., CONVEY, C., LEE, S., STONEBRAKER, M., TATBUL, N. & ZDONIK, S. 2007. Aurora: a new model and architecture for data stream management. *VLDB Journal,* 12**,** 12039.

ABOWD, G. D. & MYNATT, E. D. 2000. Charting past, present, and future research in ubiquitous computing. *ACM Transactions on Computer-Human Interaction (TOCHI),* 7**,** 29-58.

ADI, A., BOTZER, D. & ETZION, O. 2000. Semantic event model and its implication on situation detection. *ECIS 2000 Proceedings***,** 2.

AL-YAHYA, M. 2005. *A New Model for Identifying and Describing Question/Answer Resource Semantics for Distributed Access.* Citeseer.

ALI, M. I., ONO, N., KAYSAR, M., GRIFFIN, K. & MILEO, A. A semantic processing framework for IoT-enabled communication systems. International Semantic Web Conference, 2015. Springer, 241-258.

ANICIC, D. 2012. Event processing and stream reasoning with ETALIS.

ANICIC, D., FODOR, P., RUDOLPH, S. & STOJANOVIC, N. EP-SPARQL: a unified language for event processing and stream reasoning. Proceedings of the 20th international conference on World wide web, 2011. ACM, 635-644.

ANICIC, D., RUDOLPH, S., FODOR, P. & STOJANOVIC, N. 2012. Stream reasoning and complex event processing in ETALIS. *Semantic Web,* 3**,** 397-407.

ARASU, A., BABCOCK, B., BABU, S., CIESLEWICZ, J., DATAR, M., ITO, K., MOTWANI, R., SRIVASTAVA, U. & WIDOM, J. 2016. Stream: The stanford data stream management system. *Data Stream Management.* Springer.

ARENAS, M. & PÉREZ, J. Federation and Navigation in SPARQL 1.1. Reasoning Web International Summer School, 2012. Springer, 78-111.

ARPÍREZ, J. C., CORCHO, O., FERNÁNDEZ-LÓPEZ, M. & GÓMEZ-PÉREZ, A. 2003. WebODE in a nutshell. *AI magazine,* 24**,** 37.

ATEMEZING, G., CORCHO, O., GARIJO, D., MORA, J., POVEDA-VILLALÓN, M., ROZAS, P., VILA-SUERO, D. & VILLAZÓN-TERRAZAS, B. 2013. Transforming meteorological data into linked data. *Semantic Web,* 4**,** 285-290.

ATZORI, L., IERA, A. & MORABITO, G. 2010. The internet of things: A survey. *Computer networks,* 54**,** 2787-2805.

AUFAURE, M.-A., CHIKY, R., CURÉ, O., KHROUF, H. & KEPEKLIAN, G. 2016. From Business Intelligence to semantic data stream management. *Future Generation Computer Systems,* 63**,** 100-107.

AVANCHA, S., PATEL, C. & JOSHI, A. Ontology-Driven Adaptive Sensor Networks. MobiQuitous, 2004. 194-202.

BAARS, H. & ERETH, J. From Data Warehouses to analytical atoms-the Internet of Things as a centrifugal force in Business Intelligence and Analytics. ECIS, 2016. ResearchPaper3.

BALAJI, B., BHATTACHARYA, A., FIERRO, G., GAO, J., GLUCK, J., HONG, D., JOHANSEN, A., KOH, J., PLOENNIGS, J. & AGARWAL, Y. Brick: Towards a unified

metadata schema for buildings. Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments, 2016. ACM, 41-50.

BANERJEE, P., FRIEDRICH, R., BASH, C., GOLDSACK, P., HUBERMAN, B., MANLEY, J., PATEL, C., RANGANATHAN, P. & VEITCH, A. 2011. Everything as a service: Powering the new information economy. *Computer,* 44**,** 36-43.

BARBIERI, D. F., BRAGA, D., CERI, S., VALLE, E. D. & GROSSNIKLAUS, M. 2010. C-SPARQL: a continuous query language for RDF data streams. *International Journal of Semantic Computing,* 4**,** 3-25.

BARNAGHI, P., MEISSNER, S., PRESSER, M. & MOESSNER, K. Sense and sens' ability: Semantic data modelling for sensor networks. Conference Proceedings of ICT Mobile Summit 2009, 2009.

BARNAGHI, P., TÖNJES, R., HÖLLER, J., HAUSWIRTH, M., SHETH, A. & ANANTHARAM, P. Citypulse: Real-time iot stream processing and large-scale data analytics for smart city applications. Europen Semantic Web Conference (ESWC), 2014.

BARNAGHI, P., WANG, W., HENSON, C. & TAYLOR, K. 2012. Semantics for the Internet of Things: early progress and back to the future. *International Journal on Semantic Web and Information Systems (IJSWIS),* 8**,** 1-21.

BAUER, M., BAQA, H., BILBAO, S., CORCHERO, A., DANIELE, L., ESNAOLA, I., FERNÁNDEZ, I., FRÅNBERG, Ö., CASTRO, R. G. & GIROD-GENET, M. 2019. Semantic IoT Solutions-A Developer Perspective.

BELLAVISTA, P., CARDONE, G., CORRADI, A. & FOSCHINI, L. 2013. Convergence of MANET and WSN in IoT urban scenarios. *IEEE Sensors Journal,* 13**,** 3558-3567.

BENARBIA, S., ALAOUI, N. & BENNANI, S. Complex event processing distributed architecture for Massive Open Online Courses. 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT), 2018. IEEE, 273-276.

BENDADOUCHE, R., ROUSSEY, C., DE SOUSA, G., CHANET, J.-P. & HOU, K. M. Extension of the semantic sensor network ontology for wireless sensor networks: The stimulus-WSNnode-communication pattern. 5th International Workshop on Semantic Sensor Networks in conjunction with the 11th International Semantic Web Conference (ISWC), 2012. 16 p.

BERGAMASCHI, S., CASTANO, S., VINCINI, M. & BENEVENTANO, D. 2001. Semantic integration of heterogeneous information sources. *Data & Knowledge Engineering,* 36**,** 215-249.

BERMUDEZ-EDO, M., ELSALEH, T., BARNAGHI, P. & TAYLOR, K. IoT-Lite: a lightweight semantic model for the Internet of Things. Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences, 2016. IEEE, 90-97.

BIKAKIS, A. & ANTONIOU, G. Defeasible contextual reasoning in ambient intelligence: theory and applications. OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", 2010. Springer, 89-89.

BITTNER, T., DONNELLY, M. & WINTER, S. 2005. Ontology and semantic interoperability. *Large-scale 3D data integration: Challenges and Opportunities***,** 139-160.

BLÁZQUEZ, J., FERNÁNDEZ, M., GARCÍA-PINAR, J. M. & GÓMEZ-PÉREZ, A. 1998. Building ontologies at the knowledge level using the ontology design environment.

BOHLEN, M. H., BUSATTO, R. & JENSEN, C. S. Point-versus interval-based temporal data models. Data Engineering, 1998. Proceedings., 14th International Conference on, 1998. IEEE, 192-200.

BONINO, D., ALIZO, M. T. D., ALAPETITE, A., GILBERT, T., AXLING, M., UDSEN, H., SOTO, J. A. C. & SPIRITO, M. Almanac: Internet of things for smart cities. Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on, 2015. IEEE, 309-316.

BONTAS, E. P. & MOCHOL, M. Towards affreuse-oriented methodology for ontology engineering. Proc. of 7th International Conference on Terminology and Knowledge Engineering (TKE 2005), 2005. Citeseer.

BONTE, P., ONGENAE, F., DE BACKERE, F., SCHABALLIE, J., ARNDT, D., VERSTICHEL, S., MANNENS, E., VAN DE WALLE, R. & DE TURCK, F. 2017. The MASSIF platform: a modular and semantic platform for the development of flexible IoT services. *Knowledge and Information Systems*, 51**,** 89-126.

BRANK, J., GROBELNIK, M. & MLADENIĆ, D. 2005. A survey of ontology evaluation techniques.

BROWN, P. J. 1995. The stick-e document: a framework for creating context-aware applications. *Electronic Publishing-Chichester-,* 8**,** 259-272.

BYUN, H. E. & CHEVERST, K. 2004. Utilizing context history to provide dynamic adaptations. *Applied Artificial Intelligence,* 18**,** 533-548.

CALBIMONTE, J.-P. 2013. *Ontology-based access to sensor data streams.* Informatica.

CALBIMONTE, J.-P., SARNI, S., EBERLE, J. & ABERER, K. XGSN: An Open-source Semantic Sensing Middleware for the Web of Things. TC/SSN@ ISWC, 2014. 51-66.

CELDRÁN, A. H., CLEMENTE, F. J. G., PÉREZ, M. G. & PÉREZ, G. M. 2014. SeCoMan: A semantic-aware policy framework for developing privacy-preserving and context-aware smart applications. *IEEE Systems Journal,* 10**,** 1111-1124.

CHANDRASEKARAN, S., COOPER, O., DESHPANDE, A., FRANKLIN, M. J., HELLERSTEIN, J. M., HONG, W., KRISHNAMURTHY, S., MADDEN, S. R., REISS, F. & SHAH, M. A. TelegraphCQ: continuous dataflow processing. Proceedings of the 2003 ACM SIGMOD international conference on Management of data, 2003. ACM, 668-668.

CHANDY, M. K., ETZION, O. & VON AMMON, R. 10201 executive summary and manifesto--event processing. Dagstuhl Seminar Proceedings, 2011. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

CHATZIGIANNAKIS, I., HASEMANN, H., KARNSTEDT, M., KLEINE, O., KROLLER, A., LEGGIERI, M., PFISTERER, D., ROMER, K. & TRUONG, C. True self-configuration for the IoT. Internet of Things (IOT), 2012 3rd International Conference on the, 2012. IEEE, 9-15.

CHEN, H., FININ, T. & JOSHI, A. 2003. An ontology for context-aware pervasive computing environments. *The knowledge engineering review,* 18**,** 197-207.

CHEN, J., DEWITT, D. J., TIAN, F. & WANG, Y. NiagaraCQ: A scalable continuous query system for internet databases. ACM SIGMOD Record, 2000. ACM, 379-390.

CHIANESE, A., MARULLI, F., PICCIALLI, F., BENEDUSI, P. & JUNG, J. E. 2017. An associative engines based approach supporting collaborative analytics in the internet of cultural things. *Future generation computer systems,* 66**,** 187-198.

COHEN, W., RAVIKUMAR, P. & FIENBERG, S. A comparison of string metrics for matching names and records. Kdd workshop on data cleaning and object consolidation, 2003. 73-78.

COMPTON, M., BARNAGHI, P., BERMUDEZ, L., GARCÍA-CASTRO, R., CORCHO, O., COX, S., GRAYBEAL, J., HAUSWIRTH, M., HENSON, C. & HERZOG, A. 2012. The SSN ontology of the W3C semantic sensor network incubator group. *Web semantics: science, services and agents on the World Wide Web,* 17**,** 25-32.

COMPTON, M., HENSON, C., LEFORT, L., NEUHAUS, H. & SHETH, A. A survey of the semantic specification of sensors. Proceedings of the 2nd International Conference on Semantic Sensor Networks-Volume 522, 2009. CEUR-WS. org, 17-32.

CRESWELL, J. W. 2009. Research designs: Qualitative, quantitative, and mixed methods approaches. Thousand Oaks, CA: Sage Publications.

CRISTEA, V., DOBRE, C. & POP, F. 2013. Context-aware environments for the internet of things. *Internet of Things and inter-cooperative computational technologies for collective intelligence.* Springer.

CUGOLA, G. & MARGARA, A. 2012. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys (CSUR),* 44**,** 15.

D'AQUIN, M. 2012. Modularizing ontologies. *Ontology Engineering in a Networked World.* Springer.

DANIELE, L., DEN HARTOG, F. & ROES, J. Created in close interaction with the industry: the smart appliances reference (SAREF) ontology. International Workshop Formal Ontologies Meet Industries, 2015. Springer, 100-112.

DE HOOG, R. 1998. Methodologies for building knowledge based systems: achievements and prospects. *The Handbook of Applied Expert Systems*.

DE, S., ELSALEH, T., BARNAGHI, P. & MEISSNER, S. 2012. An internet of things platform for real-world and digital objects. *Scalable Computing: Practice and Experience,* 13**,** 45-58.

DELL'AGLIO, D., DAO-TRAN, M., CALBIMONTE, J.-P., LE PHUOC, D. & DELLA VALLE, E. A query model to capture event pattern matching in RDF stream processing query languages. European Knowledge Acquisition Workshop, 2016. Springer, 145-162.

DELLA VALLE, E., CERI, S., VAN HARMELEN, F. & FENSEL, D. 2009. It's a streaming world! Reasoning upon rapidly changing information. *IEEE Intelligent Systems,* 24.

DEY, A. K., ABOWD, G. D. & SALBER, D. 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human–Computer Interaction,* 16**,** 97-166.

EID, M., LISCANO, R. & EL SADDIK, A. A universal ontology for sensor networks data. Computational Intelligence for Measurement Systems and Applications, 2007. CIMSA 2007. IEEE International Conference on, 2007. IEEE, 59-62.

ENGLER, M., VRANDECIC, D. & SURE, Y. A tool for diligent argumentation: Experiences, requirements and design. Enabling Technologies: Infrastructure for Collaborative Enterprises, 2006. WETICE'06. 15th IEEE International Workshops on, 2006. IEEE, 370-375.

ETZION, O. 2010. Event processing: past, present and future. *Proceedings of the VLDB Endowment,* 3**,** 1651-1652.

ETZION, O., NIBLETT, P. & LUCKHAM, D. C. 2011a. *Event processing in action*, Manning Greenwich.

ETZION, O., RABINOVICH, E. & SKARBOVSKY, I. Non functional properties of event porcessing. Proceedings of the 5th ACM international conference on Distributed event-based system, 2011b. ACM, 365-366.

FERNÁNDEZ-LÓPEZ, M., GÓMEZ-PÉREZ, A. & JURISTO, N. 1997. Methontology: from ontological art towards ontological engineering.

FLURY, T., PRIVAT, G. & RAMPARANY, F. 2004. OWL-based location ontology for context-aware services. *Proceedings of the Artificial Intelligence in Mobile Systems (AIMS 2004)***,** 52-57.

FRANKLIN, D. & FLASCHBART, J. All gadget and no representation makes jack a dull environment. Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments, 1998. 155-160.

GANGEMI, A. Ontology design patterns for semantic web content. International semantic web conference, 2005. Springer, 262-276.

GARCÍA, F., BERTOA, M. F., CALERO, C., VALLECILLO, A., RUIZ, F., PIATTINI, M. & GENERO, M. 2006. Towards a consistent terminology for software measurement. *Information and Software Technology,* 48**,** 631-644.

GAROFALAKIS, M., GEHRKE, J. & RASTOGI, R. 2016. *Data Stream Management: Processing High-Speed Data Streams*, Springer.

GLUHAK, A., KRCO, S., NATI, M., PFISTERER, D., MITTON, N. & RAZAFINDRALAMBO, T. 2011. A survey on facilities for experimental internet of things research. *IEEE Communications Magazine,* 49.

GOLAB, L. & ÖZSU, M. T. 2003. Issues in data stream management. *ACM Sigmod Record,* 32**,** 5-14.

GOMEZ-PEREZ, A. 1994. Some ideas and examples to evaluate ontologies. KSL. Stanford University.

GÓMEZ-PÉREZ, A., FERNÁNDEZ-LÓPEZ, M. & CORCHO, O. 2003. Ontological Engineering. Advanced Information and Knowlege Processing. Springer To follow.

GÓMEZ-PÉREZ, A., FERNÁNDEZ-LÓPEZ, M. & CORCHO, O. 2006. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*, Springer Science & Business Media.

GÓMEZ-PÉREZ, A., FERNÁNDEZ, M. & VICENTE, A. D. 1996. Towards a method to conceptualize domain ontologies.

GÓMEZ-PÉREZ, A. 2001. Evaluation of ontologies. *International Journal of intelligent systems,* 16**,** 391-409.

GOODWIN, C. & RUSSOMANNO, D. J. An ontology-based sensor network prototype environment. Proceedings of the Fifth International Conference on Information Processing in Sensor Networks, 2006. 1-2.

GRANDI, F. T-SPARQL: A TSQL2-like Temporal Query Language for RDF. ADBIS (Local Proceedings), 2010. Citeseer, 21-30.

GRAY, A. J., GARCÍA-CASTRO, R., KYZIRAKOS, K., KARPATHIOTAKIS, M., CALBIMONTE, J.-P., PAGE, K., SADLER, J., FRAZER, A., GALPIN, I. & FERNANDES, A. A. A semantically enabled service architecture for mashups over streaming and stored data. Extended Semantic Web Conference, 2011a. Springer, 300-314.

GRAY, A. J., SADLER, J., KIT, O., KYZIRAKOS, K., KARPATHIOTAKIS, M., CALBIMONTE, J.-P., PAGE, K., GARCÍA-CASTRO, R., FRAZER, A. & GALPIN, I. 2011b. A semantic sensor web for environmental decision support applications. *Sensors,* 11**,** 8855-8887.

GRUBER, T. R. 1995. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies,* 43**,** 907-928.

GRÜNINGER, M. & FOX, M. S. 1995a. Methodology for the design and evaluation of ontologies.

GRÜNINGER, M. & FOX, M. S. 1995b. The role of competency questions in enterprise engineering. *Benchmarking—Theory and Practice.* Springer.

GUTIERREZ, C., HURTADO, C. A. & VAISMAN, A. 2007. Introducing time into RDF. *IEEE Transactions on Knowledge and Data Engineering,* 19.

GYRARD, A., BONNET, C. & BOUDAOUD, K. Domain knowledge Interoperability to build the semantic web of things. W3C Workshop on the Web of Things, 2014a. 1-5.

GYRARD, A., BONNET, C. & BOUDAOUD, K. Enrich machine-to-machine data with semantic web technologies for cross-domain applications. Internet of Things (WF-IoT), 2014 IEEE World Forum on, 2014b. IEEE, 559-564.

GYRARD, A., SERRANO, M. & ATEMEZING, G. A. Semantic web methodologies, best practices and ontology engineering applied to Internet of Things. Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on, 2015. IEEE, 412-417.

HACHEM, S., PATHAK, A. & ISSARNY, V. 2014. Service-oriented middleware for large-scale mobile participatory sensing. *Pervasive and Mobile Computing,* 10**,** 66-82.

HALLER, S., SERBANATI, A., BAUER, M. & CARREZ, F. A domain model for the internet of things. Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCom), IEEE International Conference on and IEEE Cyber, Physical and Social Computing, 2013. IEEE, 411-417.

HAN, S. N., LEE, G. M. & CRESPI, N. Towards automated service composition using policy ontology in building automation system. Services Computing (SCC), 2012 IEEE Ninth International Conference on, 2012. IEEE, 685-686.

HENSON, C., SHETH, A. & THIRUNARAYAN, K. 2012. Semantic perception: Converting sensory observations to abstractions. *IEEE Internet Computing,* 16**,** 26-34.

HIRMER, P., WIELAND, M., BREITENBÜCHER, U. & MITSCHANG, B. Dynamic ontology-based sensor binding. East European Conference on Advances in Databases and Information Systems, 2016. Springer, 323-337.

HOVY, E. 2002. Comparing sets of semantic relations in ontologies. *The semantics of relationships.* Springer.

HRISTOZOVA, M. & STERLING, L. Experiences with Ontology Development for Value-Added Publishing. OAS, 2003. 17-24.

HULL, R., NEAVES, P. & BEDFORD-ROBERTS, J. Towards situated computing. Wearable Computers, 1997. Digest of Papers., First International Symposium on, 1997. IEEE, 146-153.

IEEE 1996. IEEE Standard for Developing Software Life Cycle Processes. *IEEE Std 1074-1995*, i.

INDRA. 2017. *SOFIA2* [Online]. Available: www.sofia2.com [Accessed].

JANOWICZ, K. & COMPTON, M. The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology. SSN, 2010.

JAYASEKARA, S., KANNANGARA, S., DAHANAYAKAGE, T., RANAWAKA, I., PERERA, S. & NANAYAKKARA, V. 2015. Wihidum: Distributed complex event processing. *Journal of Parallel and Distributed Computing,* 79**,** 42-51.

JIN, J., GUBBI, J., MARUSIC, S. & PALANISWAMI, M. 2014. An information framework for creating a smart city through internet of things. *IEEE Internet of Things Journal,* 1**,** 112-121.

KANG, J. & PARK, S. 2013. Context-aware services framework based on semantic web services for automatic discovery and integration of context. *International Journal of Advancements in Computing Technology,* 5.

KIM, J.-H., KWON, H., KIM, D.-H., KWAK, H.-Y. & LEE, S.-J. Building a service-oriented ontology for wireless sensor networks. Computer and Information Science, 2008. ICIS 08. Seventh IEEE/ACIS International Conference on, 2008. IEEE, 649-654.

KIM, S. I. & KIM, H. S. Ontology based location reasoning method using smart phone data. Information Networking (ICOIN), 2015 International Conference on, 2015. IEEE, 509-514.

KLYNE, G. & CARROLL, J. J. 2006. Resource description framework (RDF): Concepts and abstract syntax.

KORTUEM, G., KAWSAR, F., SUNDRAMOORTHY, V. & FITTON, D. 2010. Smart objects as building blocks for the internet of things. *IEEE Internet Computing,* 14**,** 44-51.

KOSTELNIK, P., SARNOVSK, M. & FURDIK, K. 2011. The semantic middleware for networked embedded systems applied in the internet of things and services domain. *Scalable Computing: Practice and Experience,* 12**,** 307-316.

KOUBARAKIS, M. & KYZIRAKOS, K. Modeling and querying metadata in the semantic sensor web: The model stRDF and the query language stSPARQL. Extended Semantic Web Conference, 2010. Springer, 425-439.

KUZNETSOVA, E., LI, Y.-F., RUIZ, C. & ZIO, E. 2014. An integrated framework of agent-based modelling and robust optimization for microgrid energy management. *Applied Energy,* 129**,** 70-88.

KYRIAZIS, D. & VARVARIGOU, T. 2013. Smart, autonomous and reliable Internet of Things. *Procedia Computer Science,* 21**,** 442-448.

LEBLANC, D., BUTTERS, J. & COOK, C. 2017. Multi-language support for dynamic ontology. Google Patents.

LENZERINI, M. Data integration: A theoretical perspective. Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, 2002. ACM, 233-246.

LIU, L., PU, C. & TANG, W. 1999. Continual queries for internet scale event-driven information delivery. *IEEE Transactions on Knowledge and Data Engineering,* 11**,** 610-628.

LOZANO-TELLO, A. & GÓMEZ-PÉREZ, A. 2004. Ontometric: A method to choose the appropriate ontology. *Journal of database management,* 2**,** 1-18.

LUCKHAM, D. C. 2011. *Event processing for business: organizing the real-time enterprise*, John Wiley & Sons.

MADDEN, S., SHAH, M., HELLERSTEIN, J. M. & RAMAN, V. Continuously adaptive continuous queries over streams. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, 2002. ACM, 49-60.

MANYIKA, J. 2015. *The Internet of Things: Mapping the value beyond the hype*, McKinsey Global Institute.

MEI, Y. & MADDEN, S. Zstream: a cost-based query processor for adaptively detecting composite events. Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, 2009. ACM, 193-206.

MÜLLER, H., CABRAL, L., MORSHED, A. & SHU, Y. From RESTful to SPARQL: A Case Study on Generating Semantic Sensor Data. SSN@ ISWC, 2013. 51-66.

NACHABE, L., GIROD-GENET, M. & EL HASSAN, B. 2015. Unified data model for wireless sensor network. *IEEE Sensors Journal,* 15**,** 3657-3667.

NAGIB, A. M. & HAMZA, H. S. 2016. SIGHTED: a framework for semantic integration of heterogeneous sensor data on the Internet of Things. *Procedia Computer Science,* 83**,** 529-536.

NAGOWAH, S. D., STA, H. B. & GOBIN-RAHIMBUX, B. An Overview of Semantic Interoperability Ontologies and Frameworks for IoT. 2018 Sixth International Conference on Enterprise Systems (ES), 2018. IEEE, 82-89.

NGUYEN, A.-T., REITER, S. & RIGO, P. 2014. A review on simulation-based optimization methods applied to building performance analysis. *Applied Energy,* 113**,** 1043-1058.

NIITSUMA, M., YOKOI, K. & HASHIMOTO, H. Describing human-object interaction in intelligent space. Human System Interactions, 2009. HSI'09. 2nd Conference on, 2009. IEEE, 395-399.

NORDRUM, A. 2016. The internet of fewer things [news]. *IEEE Spectrum,* 53**,** 12-13.

NOVIKOV, A. M. & NOVIKOV, D. A. 2013. *Research methodology: From philosophy of science to research design*, CRC Press.

NOY, N. F. & MCGUINNESS, D. L. 2001. Ontology development 101: A guide to creating your first ontology. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, Stanford, CA.

OBRST, L., CEUSTERS, W., MANI, I., RAY, S. & SMITH, B. 2007. The Evaluation of Ontologies: Toward Improved Semantic Interoperability. Chapter in: Semantic Web: Revolutionizing Knowledge Discovery in the Life Sciences, Christopher JO Baker and Kei-Hoi Cheung, Eds. Springer.

PASCOE, J. Adding generic contextual capabilities to wearable computers. Wearable Computers, 1998. Digest of Papers. Second International Symposium on, 1998. IEEE, 92-99.

PATKOS, T., CHRYSAKIS, I., BIKAKIS, A., PLEXOUSAKIS, D. & ANTONIOU, G. A reasoning framework for ambient intelligence. Hellenic Conference on Artificial Intelligence, 2010. Springer, 213-222.

PATON, N. W. & DÍAZ, O. 1999. Introduction. *Active Rules in Database Systems.* Springer.

PERERA, C., ZASLAVSKY, A., CHRISTEN, P. & GEORGAKOPOULOS, D. 2014. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials,* 16**,** 414-454.

PERRY, M., JAIN, P. & SHETH, A. P. 2011. Sparql-st: Extending sparql to support spatiotemporal queries. *Geospatial semantics and the semantic web.* Springer.

PFISTERER, D., ROMER, K., BIMSCHAS, D., KLEINE, O., MIETZ, R., TRUONG, C., HASEMANN, H., KRÖLLER, A., PAGEL, M. & HAUSWIRTH, M. 2011. SPITFIRE: toward a semantic web of things. *IEEE Communications Magazine,* 49**,** 40-48.

PINTO, H. S. & MARTINS, J. P. A methodology for ontology integration. Proceedings of the 1st international conference on Knowledge capture, 2001. ACM, 131-138.

PINTO, H. S., STAAB, S. & TEMPICH, C. DILIGENT: Towards a fine-grained methodology for distributed, loosely-controlled and evolvInG. Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), 2004. 393.

PORZEL, R. & MALAKA, R. A task-based approach for ontology evaluation. ECAI Workshop on Ontology Learning and Population, Valencia, Spain, 2004. Citeseer, 1-6.

PRESSMAN, R. S. & MAXIM, B. R. 2015. *Software engineering : a practitioner's approach,* New York, McGraw-Hill Education.

PRUD, E. & SEABORNE, A. 2006. SPARQL query language for RDF.

RAZZAQUE, M. A., MILOJEVIC-JEVRIC, M., PALADE, A. & CLARKE, S. 2016. Middleware for Internet of Things: A survey. *IEEE Internet of Things Journal,* 3**,** 70-95.

RIVERA, J. & GOASDUFF, L. 2014. Gartner says a thirty-fold increase in internet-connected physical devices by 2020 will significantly alter how the supply chain operates. *Gartner*.

RODDEN, T., CHEVERST, K., DAVIES, K. & DIX, A. Exploiting context in HCI design for mobile systems. Workshop on human computer interaction with mobile devices, 1998. Glasgow, 21-22.

RUSSOMANNO, D. J., KOTHARI, C. & THOMAS, O. Sensor ontologies: from shallow to deep models. System Theory, 2005. SSST'05. Proceedings of the Thirty-Seventh Southeastern Symposium on, 2005. IEEE, 107-112.

SANTOS, J. R. A. 1999. Cronbach's alpha: A tool for assessing the reliability of scales. *Journal of extension,* 37**,** 1-5.

SCHARFFE, F., EUZENAT, J. & FENSEL, D. Towards design patterns for ontology alignment. Proceedings of the 2008 ACM symposium on Applied computing, 2008. ACM, 2321-2325.

SCHILIT, B., ADAMS, N. & WANT, R. Context-aware computing applications. Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on, 1994. IEEE, 85-90.

SCHLENOFF, C., HONG, T., LIU, C., EASTMAN, R. & FOUFOU, S. A literature review of sensor ontologies for manufacturing applications. Robotic and Sensors Environments (ROSE), 2013 IEEE International Symposium on, 2013. IEEE, 96-101.

SCHREIBER, G. & AKKERMANS, H. 1999. Knowledge Engineering and Management: The CommonKADS Methodology. Massachusetts. MIT Press.

SCHREIER, U., PIRAHESH, H., AGRAWAL, R. & MOHAN, C. Alert: An architecture for transforming a passive DBMS into an active DBMS. Proceedings of the 17th International Conference on Very Large Data Bases, 1991. Morgan Kaufmann Publishers Inc., 469-478.

SHETH, A. 2016. Internet of things to smart iot through semantic, cognitive, and perceptual computing. *IEEE Intelligent Systems,* 31**,** 108-112.

SHI, Y., LI, G., ZHOU, X. & ZHANG, X. 2012. Sensor ontology building in semantic sensor web. *Internet of things.* Springer.

SHULL, F., SINGER, J. & SJØBERG, D. I. 2007. *Guide to advanced empirical software engineering,* Springer.

SIRIN, E., BULKA, B. & SMITH, M. Terp: Syntax for OWL-friendly SPARQL Queries. OWLED, 2010.

SOLDATOS, J., KEFALAKIS, N., HAUSWIRTH, M., SERRANO, M., CALBIMONTE, J.-P., RIAHI, M., ABERER, K., JAYARAMAN, P. P., ZASLAVSKY, A. & ŽARKO, I. P. 2015. Openiot: Open source internet-of-things in the cloud. *Interoperability and open-source solutions for the internet of things.* Springer.

SONG, Z., BIES, A., STRASSEL, S., RIESE, T., MOTT, J., ELLIS, J., WRIGHT, J., KULICK, S., RYANT, N. & MA, X. From light to rich ERE: annotation of entities, relations, and events. Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation, 2015. 89-98.

STAAB, S., STUDER, R., SCHNURR, H.-P. & SURE, Y. 2001. Knowledge processes and ontologies. *IEEE Intelligent systems,* 16**,** 26-34.

STRANG, T. & LINNHOFF-POPIEN, C. A context modeling survey. Workshop on advanced context modelling, reasoning and management, UbiComp, 2004. 34-41.

SU, X., RIEKKI, J., NURMINEN, J. K., NIEMINEN, J. & KOSKIMIES, M. 2015. Adding semantics to internet of things. *Concurrency and Computation: Practice and Experience,* 27**,** 1844-1860.

SUÁREZ-FIGUEROA, M. C. 2010. *NeOn Methodology for building ontology networks: specification, scheduling and reuse.* Informatica.

SUÁREZ-FIGUEROA, M. C., GÓMEZ-PÉREZ, A. & FERNÁNDEZ-LÓPEZ, M. 2012. The NeOn methodology for ontology engineering. *Ontology engineering in a networked world.* Springer.

TAPPOLET, J. & BERNSTEIN, A. Applied temporal RDF: Efficient temporal querying of RDF data with SPARQL. European Semantic Web Conference, 2009. Springer, 308-322.

TAYLOR, K. & LEIDINGER, L. Ontology-driven complex event processing in heterogeneous sensor networks. Extended Semantic Web Conference, 2011. Springer, 285-299.

TAYLOR, K. L. & LEIDINGER, L. 2015. Ontology-driven complex event processing. Google Patents.

TEIXEIRA, T., HACHEM, S., ISSARNY, V. & GEORGANTAS, N. Service oriented middleware for the internet of things: a perspective. European Conference on a Service-Based Internet, 2011. Springer, 220-229.

TERRY, D., GOLDBERG, D., NICHOLS, D. & OKI, B. 1992. *Continuous queries over append-only databases,* ACM.

UNION, I. C. 2012. Internet of Things Global Standards Initiative.

USCHOLD, M. 1996. Building ontologies: Towards a unified methodology. *Technical report-university of Edinburgh artificial intelligence applications institute AIAI TR*.

VÁZQUEZ SALCEDA, J., ÁLVAREZ NAPAGAO, S., TEJEDA GÓMEZ, J. A., OLIVA FELIPE, L. J., GARCIA GASULLA, D., GÓMEZ SEBASTIÀ, I. & CODINA BUSQUET, V. Making smart cities smarter using artificial intelligence techniques for smarter mobility. SMARTGREENS 2014: proceedings of the 3rd International Conference on Smart Grids and Green IT Systems, 2014. SciTePress, IS7-IS11.

VELARDI, P., NAVIGLI, R., CUCHIARELLI, A. & NERI, R. 2005. Evaluation of OntoLearn, a methodology for automatic learning of domain ontologies. *Ontology Learning from Text: Methods, evaluation and applications,* 123.

VERMESAN, O. & FRIESS, P. 2014. *Internet of things-from research and innovation to market deployment*, River publishers Aalborg.

VRANDEČIĆ, D. 2009. Ontology evaluation. *Handbook on ontologies.* Springer.

WANG, C., CHEN, N., HU, C., YAN, S. & WANG, W. A general sensor web resource ontology for atmospheric observation. Geoscience and Remote Sensing Symposium (IGARSS), 2011 IEEE International, 2011. IEEE, 3436-3439.

WANG, W., DE, S., CASSAR, G. & MOESSNER, K. 2013. Knowledge representation in the internet of things: semantic modelling and its applications. *automatika,* 54**,** 388-400.

WANG, X., ZHANG, X. & LI, M. 2015. A survey on semantic sensor web: Sensor ontology, mapping and query. *International Journal of u-and e-Service, Science and Technology,* 8**,** 325-342.

WANT, R. 2004. Enabling ubiquitous sensing with RFID. *Computer,* 37**,** 84-86.

WELTY, C. & GUARINO, N. 2001. Supporting ontological analysis of taxonomic relationships. *Data & Knowledge Engineering,* 39**,** 51-74.

WITT, K. J., STANLEY, J., SMITHBAUER, D., MANDL, D., LY, V., UNDERBRINK, A. & METHENY, M. Enabling Sensor Webs by utilizing SWAMO for autonomous operations. 8th NASA Earth Science Technology Conference, 2008. 263-270.

XUE, L., LIU, Y., ZENG, P., YU, H. & SHI, Z. An ontology based scheme for sensor description in context awareness system. Information and Automation, 2015 IEEE International Conference on, 2015. IEEE, 817-820.

YAN, B., HU, Y., KUCZENSKI, B., JANOWICZ, K., BALLATORE, A., KRISNADHI, A. A., JU, Y., HITZLER, P., SUH, S. & INGWERSEN, W. An Ontology For Specifying Spatiotemporal Scopes in Life Cycle Assessment. Diversity++@ ISWC, 2015. 25-30.

YUN, J., AHN, I.-Y., SONG, J. & KIM, J. 2019. Implementation of Sensing and Actuation Capabilities for IoT Devices Using oneM2M Platforms. *Sensors,* 19**,** 4567.

ZHANG, D., WANG, L., XIONG, H. & GUO, B. 2014. 4W1H in mobile crowd sensing. *IEEE Communications Magazine,* 52**,** 42-48.

ZHANG, N., CHEN, H., CHEN, X. & CHEN, J. 2016. Semantic framework of internet of things for smart cities: case studies. *Sensors,* 16**,** 1501.

ZHANG, Y., DUC, P. M., CORCHO, O. & CALBIMONTE, J.-P. SRBench: a streaming RDF/SPARQL benchmark. International Semantic Web Conference, 2012. Springer, 641-657.

APPENDIX A: SURVEY QUESTIONNAIRE

This Appendix contains all survey structure, separated by sections, containing all items present in the survey form.

IoT-Ont Ontology Evaluation

This evaluation survey is aimed at domain expert and researchers in Internet of Things. The purpose of this survey is to evaluate the concepts represented and how they are related in IoT-Ont, an ontology for Internet of Things, which has been developed in a PhD project.

The survey contains 7 questions regarding the ontology evaluation, 4 questions regarding the respondent background and knowledge. Your participation is totally anonymous, if you would like to be informed about the final evaluation result, please enter a valid email address below.

Email address (optional):

The results of the survey will be used to assess and evaluate the developed ontology in this research. It is assumed that respondent has some knowledge in the IoT domain.

Part A: Background Knowledge

1. Are you familiar with the Internet of Things or Wireless Sensor Networks?

   ☐ Yes
   ☐ Yes, but I don't consider myself an expert.
   ☐ No

2. Are you familiar with Software Engineering?

   ☐ Yes
   ☐ Yes, but I don't consider myself an expert.
   ☐ No

3. Are you familiar with Ontology Engineering?

   ☐ Yes
   ☐ Yes, but I don't consider myself an expert.
   ☐ No

4. Have you developed an Ontology before?

   ☐ Yes
   ☐ No

Part B: IoT-Ont Evaluation

Description of Concepts (Conciseness)

Please evaluate the description of each term presented in the attached Table titled IoT-Ont Description of Concepts.

1. The ontology correctly describes all concepts related to Internet of Things.

   ☐Strongly Agree
   ☐Agree
   ☐Undecided
   ☐Disagree
   ☐Strongly Disagree

Please evaluate the description of each term presented in the attached Table titled IoT-Ont Description of Modules.

2. The ontology correctly describes all modules related to Internet of Things.

☐Strongly Agree
☐Agree
☐Undecided
☐Disagree
☐Strongly Disagree

Ontology Coverage (Completeness)

Please evaluate if the concepts in ontology correctly represent the Internet of Things field.

3. All relevant concepts related to the Internet of Things domain have been represented in the ontology.

☐Strongly Agree
☐Agree
☐Undecided
☐Disagree
☐Strongly Disagree

Relationships (Consistency)

Please evaluate if the relationships between the concepts are correctly mapped.

4. All relevant relationships related to the Internet of Things concepts have been correctly mapped in the ontology.

☐Strongly Agree
☐Agree
☐Undecided
☐Disagree
☐Strongly Disagree

Expandability (Modularity)

Please evaluate if the ontology follows a modular ontology design, where modules and concepts could easily be added.

5. New terms and modules can be introduced without the need to revise the existing structure of the ontological model

   ☐Strongly Agree
   ☐Agree
   ☐Undecided
   ☐Disagree
   ☐Strongly Disagree

Additional Information

6. In your opinion, do you think an ontology for IoT is useful?

   ☐Strongly Agree
   ☐Agree
   ☐Undecided
   ☐Disagree
   ☐Strongly Disagree

7. Please use this space for any additional comments or suggestions in regards to IoT-Ont.

APPENDIX B: ETHICAL APPROVAL

University of Salford MANCHESTER

Research, Innovation and Academic Engagement Ethical Approval Panel

Doctoral & Research Support
Research and Knowledge Exchange,
Room 827, Maxwell Building
University of Salford
Manchester
M5 4WT

T +44(0)161 295 5278

www.salford.ac.uk/

31 August 2018

Amer Al Lahham

Dear Amer,

RE: ETHICS APPLICATION STR1718-53: Ontology-Based Context-Aware Model for Event processing in an IoT environment

Based on the information you provided, I am pleased to inform you that your application STR1718-38 has been approved.

If there are any changes to the project and/ or its methodology, please inform the Panel as soon as possible by contacting S&T-ResearchEthics@salford.ac.uk

Yours sincerely,

APHigham.

Dr Anthony Higham
Chair of the Science & Technology Research Ethics Panel