Contextualising Sensors with Linked Data

to Improve Relevancy, Data Quality and Network Adaptability

Myriam Leggieri

Submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy

PRIMARY SUPERVISOR:
Dr. John Breslin

SECONDARY SUPERVISOR:
Dr. Brian Davis

INTERNAL EXAMINER:
Dr. Michael Schukat

EXTERNAL EXAMINER:
Prof. Dr. Steffen Staab

Insight Centre for Data Analytics,
National University of Ireland, Galway
February 2015

Declaration

I declare that the work covered by this thesis is composed by myself, and that it has not been submitted for any other degree or professional qualification except as specified.

Myriam Leggieri

The work reported in this thesis was supported by the European Union (Grant no. ICT-258885 – SPITFIRE, Grant no. ICT-2011.1.3 – OpenIoT and Grant no. ICT-608682 – Vital) and by Science Foundation Ireland (SFI/12/RC/2289 and Grant no. SFI/08/CE/I1380 – Líon-2).

Abstract

A sensor taken by itself, with no indication of what it is sensing, with what unit of measurement and where, would keep working without anybody being able to interpret its output. The sensor needs to be "contextualised". In this thesis we propose a sensor data modeling based on the Linked Data principles. We support its uptake by releasing LD4Sensors, a Web application that facilitates manual (GUI) and automated (REST API) sensor annotation, storage and retrieval using our data model. A side gain of this approach consists of automating the (currently manual) setup of several sensor settings, which will be presented in a demonstrative application and architecture. Finally, we demonstrate the advantages of using our data model to improve sensor relevancy and enrich Web data, within two different applications. First, we use our data model to improve the relevancy prediction of sensors during Daily Activity Logging tasks. Currently, Task Logging is performed by classifying the previously collected sensor readings in order to identify which task/s they were measuring. This approach has two downsides. First, the task identification happens after all the data has been recorded. Second, selecting which of the available sensors to query is difficult. Usually, sensors are selected according to their energy consumption or location. However, a more fine-grained selection would improve the system efficiency by reducing the amount of data to record an process while at the same time, saving energy. As the amount of Internet-connected objects increases and as we move towards ubiquitous computing web, the selection of on-demand information sources has become a significant requirement. In this thesis, I demonstrate that using our model based on Semantic Technologies and Distributional Semantic techniques we can identify which sensors to use during Task Logging while predicting the task being sensed in real-time. I compare my results with the other state-of-the-art Task Logging techniques showing the improvement. Second, we use our data model to enrich Web content in order to bridge the traditional Web and the Sensor Web. Traditional Web content is long-lived, as it most of the times lacks of real-time information. Sensors deployed pervasively throughout Smart Cities have the potential to be the source of such real-time information. However, querying all the sensor data sources is costly for they are distributed and live streaming high volumes of data. We realised the G-Sensing application that, as a Mozilla Firefox add-on, displays sensors data related to Google search results that represent real places. We demonstrate the feasibility and extensibility of our approach and the advantages it brings to the final user.

Acknowledgements

I would like to thank Dr. Brian Davis for supervising me during the write-up of this thesis and the finalising of the work described in Chapter 5.

Thanks to Prof. Dr. Gregor Schiele, Dr. Michael Hausenblas and Dr. Alexandre Passant for supervising me in different stages of my PhD. They facilitated the work described in Chapter 5, Chapter 2 and Chapter 4 of this thesis, respectively.

Thanks to Dr. John Breslin for the continuous support, encouragement, inspiring ideas and help given to me throughout my PhD years. Thanks for also supervising me during the write-up of this thesis.

Thanks to Prof. Dr. Manfred Hauswirth for supervising me throughout my PhD years. Thanks for the direction given to me, suggestions, help, support, inspiring conversations and mind-opening ideas.

Thanks to Sarven Capadisli (with his post-its), Christian von der Weth (with his delicious dinners) and Henning Hasemann (with his emails), who helped me going through tough times and made these last years so much full of kind memories. Thanks to Alessandra Mileo, Laleh Kazemzadeh and Milena Caires, for their friendship and for never giving up after all the times I lazily declined to go out on a weekend. Thanks to Pasquale Minervini and Micaela Liguori who have always been there for laughing and chatting and being close to me, despite the miles of distance that separate us.

A special "thank you" to Michael Malone who is, for me, a most precious gift.

Dedicated to my parents,
Maria Caporusso and Gianfranco Leggieri to whom I owe everything and beyond.

Contents

List of Figures

List of Tables

List of Listings

Acronyms

OV Observation Value

LD4S Linked Data for Sensors

OGC Open Geospatial Consortium

SWE Sensor Web Enablement

URI Unique Resource Identifier

HTTP Hypertext Transfer Protocol

RDF Resource Description Framework

SPARQL Simple Protocol and RDF Query Language

UPnP Universal Plug and Play

TCP Transmission Control Protocol

IP Internet Protocol

UDP User Datagram Protocol

SOAP Simple Object Access Protocol

IoT Internet of Things

WSDL Web Services Description Language

DBMS Database Management System

HADP Human Active Database Passive model

DAHP Database Active Human Passive model

QoS Quality of Service

M2M Machine-to-Machine paradigm

WoT Web of Things

REST Representational State Transfer

OWL Web Ontology Language

RDB Relational DataBase

ETL Extract, Transform and Load process

LOD Linked Open Datasets

LD4S Linked Data for Sensors web service

Part I.

Prelude

Chapter 1.

Introduction

Plants *move* towards the sunlight. They don't have a brain or muscles. They use both differential growth of cells and circulation of water in or out of specific cells in order to *move* exposing their surfaces to the sun as much as possible. This is an example of sponatneous adaptation to the context. Living beings adapt according to the perception of the surroundings through their senses. This adaptation manifests as a consequence of a decision making process that - either consciously or unconsciously, specially based on instinct -triggers actions. The latter ultimately causes a change in the surroundings. Electronic devices are not living beings but through time they have been equipped with human-like capabilities such as memory, logic and, more recently, sensors to emulate human senses. When connected computational devices are interwoven with common artifacts. Processing, sensing, activation and communication have become embedded into devices and environments making computing part of our daily life in what we call *Pervasive Computing*.

Nonetheless sensors alone would generate meaningless measurements. We can automatically make sense of the sensors output only if we assign machine-understandable labels to their features. A series of real numbers could then become labelled as measurements of the luminosity of a plant at different times of the day. However, this information alone would be limited unless otherwise enriched with its context. Such sensors could be placed in the context of a biological experiment within a laboratory in which the luminosity of several other plants is measured in different locations; rather than in a botanic garden or private household. The process of adding contextual information is what we call *sensor contextualisation*.

Context is more formally defined as *any information that can be used to characterize 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, 2001].

A situation is the state of the real world at a certain moment or during an interval in time at a certain location. Context is identified by a name and includes a description of a type of situation by its characteristic features. A situation S belongs to a context C when all conditions in the description of C evaluate to true in a given situation S. It is assumed that for all situations that belong to the same context the sensory input of the characterizing features is similar [Schmidt, 2008]. Creating a description of a context involves solving similar problems for creating a query for information retrieval. To assess the quality of a description we use measures such a precision and recall from information retrieval. Based on these definitions, context can be regarded as a pattern, which can be used to match situations of the same type.

1.1. Problem Statement

In this Thesis we expose our approach for sensor contextualisation and how we facilitate its uptake. The growing number of Internet connected devices is contributing to a growth of available data and thus to information overload. It becomes important to detect the relevancy of such data which highly depends on context. A specific information can be more or less relevant for the final user depending on the situation in which the user is currently immersed. Our approach can be used to make better autonomous decisions on which information to filter out according to the context. For instance, if while cooking quick defrost meals sensors attached to the microwave may be relevant given that the microwave is likely going to be used, they wont be just as important while the electric oven is in use for more elaborate meals. These situations differ despite being both about the same activity, such as "cooking". Applied to this example, our solution automates the relevancy prediction of the sensors on the microwave and on any other appliance, while actions are being performed by the user and logged. We propose how to collect the contextual information via Linked Sensor Data and use it as a filter.

With the rapid development of electronic devices and service technologies, providing suitable services that can consider location, available devices as well as other user- related runtime contextual data is in high demand. Context-awareness becomes a key feature for providing adaptable services, for instance, when the best-suited services are required to be selected according to the relevant context information or when services are required

to adapt to context changes during their execution. Siri, Google Now, Voice Search and a plethora of new and promising mobile applications are evolving into personal assistants. They are starting to know us better, taking over the minutiae in our life, helping us to adjust dynamically to the surprises and changes of everyday life [Scoble and Israel, 2013]. The key to what makes these products so game changing is that, because they understand the context of what we are doing, they can predict what we want to do next with high accuracy. These anticipatory systems work tirelessly on our behalf. They stay focused when we are unable. Anticipatory system filters know when to alert us to an important change and shield us from low priority or irrelevant information. Given the continuous clutter and noise of data overload, the *when* becomes crucial to the relevancy of the *what*. Without knowing when to interrupt the user for what purpose, carefully personalised information may go unnoticed. The decision to present the right kind of information or take the right action and at the right time is all about context and situation.

Five forces are shaping what is now being called the *Age of Context* [Scoble and Israel, 2013]:

- Mobile. Cell phones now exceed people on the planet, wearable computing is booming, data costs are dropping, and the number of application downloads are ever increasing.
- Social media. Almost 1.5 billion people are on social networks, and businesses are using them to get closer to the customers, present a more accessible image of themselves and learn.
- Data. The size of the Internet is expanding at an exponential rate which has seen the emergence of the concept of Big Data[1]. But its little bits of data delivered to us exactly when we want them (thanks to search) that are really impacting our lives.
- Sensors. Sensors in technology can emulate three of the five human senses: sight, touch, and hearing. Sensors can talk to us and to each other.
- Location-based services. Our location is one of the most important parts of our context.

Context-aware computing as a field is not new. For instance, in 1991 Mark Weiser wrote about ubiquitous computing: "*The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it*". A lot of progress has been made in understanding the users, interest

[1]The term Big Data describes a massive volume of both structured and unstructured data, so large that it's difficult to process using traditional database and software techniques.

graphs and social graphs and to a less extent, content graphs. The devices are now equipped with so many types of sensors that carry rich data on the users real time context. However, while power drain is a concern for continuous sensing of all the pertinent data, that is not the only factor to constrain contextual personalisation. Today iPhone and Android smartphone users still have to interrupt their own lives to support their devices: entering preferences for different modes of operations; checking in with the device when eating at a restaurant or shopping at a store; and launching applications before going jogging or working out. Context-awareness promises to move us closer to the time when our devices proactively support our lives rather than the other way around: requiring us to feed our devices with every input they need. Hence, we will see a new landscape of applications and systems that will improve peoples lives without intruding on their activities.

For instance, knowing that the user has not moved from his seat or has stayed close to a fixed location, means the smartphone would not need to turn on the GPS at all to maintain location services. So, a context-aware power manager can turn off the GPS and make the assumption that available Wi-Fi connections have not changed. In this way, context-awareness allows a prolonged battery lifetime without any intrusion into the user experience. As another example, much of the worlds knowledge and research is domain-specific, which can create barriers for persons looking to learn about findings in a discipline that they are approaching for the first time. We believe adding context to original resources can make challenging new concepts understandable. From low level context such as movement patterns to high level situations such as talking to important people, such inferences will advance the field of personalisation to take it to new heights.

After a decade of context-aware application explorations such as *supporting reminders* and *ringer manipulation*[2] use cases, this is not mainstream just yet. What is the breakthrough the field is lacking? In reality, the practical difficulty of getting high quality labelled data makes it very challenging to perform contextual inferences with high confidence. Consequently, it is also challenging to select the most appropriate time to provide certain information without involving the final user. Many applications become less useful or even cause the detriment of the user due to such low confidence during the inference process. Hence, once there is a need for any kind of ongoing or even somewhat ongoing user interaction, it becomes less about automation and is more intrusive to the user. So, we are stuck in the conundrum of requiring large amounts of user input to perfect these inferences, while we must perfect the inferences to obtain user buy-in.

[2]Adapting the ringtone volume in smartphones

Our approach to sensor contextualisation fundamentally relies on applying *Linked Data* to sensors. The term *Linked Data* refers to a set of best practices for publishing and interlinking structured data on the Web. These best practices were introduced by Tim Berners-Lee in his Web architecture note Linked Data[3] and have become known as the Linked Data principles. These principles are:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
- Include links to other URIs, so that they can discover more things.

The basic idea of Linked Data is to apply the general architecture of the World Wide Web [70] (URIs as identification mechanism, HTTP as access mechanism and HTML as content format) to the task of sharing structured data on global scale.

The Linked Data life-cycle is represented in Fig 1.1. Unstructured data or information adhering to other structured or semi-structured formalisms must be mapped to the RDF data model (Extraction). The RDF data needs then to be stored, indexed and queried efficiently (Storage and Querying).

Clients must be able to create new RDF data or to correct and extend existing ones (Authoring). Links between resource representing the same or related entities, are established (Linking). In case of lack of structure and schema information on the resource instances, the RDF data should be enriched with higher-level structures (Enrichment). Strategies for assessing the quality of data published for the Data Web (Quality Analysis) must be devised [Scheglmann et al., 2013]. Once problems are detected, strategies for repairing these problems and supporting the evolution of Linked Data are required (Evolution and Repair). Finally, clients have to be provided with browsing, searching and exploring functionality over the Linked Data with a usable and efficient approach (Search, Browsing and Exploration).

In particular, the Linking phase is the foundation of Linked Data. In Tim Berners-Lee's vision, *the Semantic Web*[4] *isn't just about putting data on the Web. It is about*

[3] http://www.w3.org/DesignIssues/LinkedData.html

[4] The term Semantic Web indicates a movement aimed at converting the current Web dominated by unstructured and semi-structured documents into a "Web of Data" that can be processed by machines.

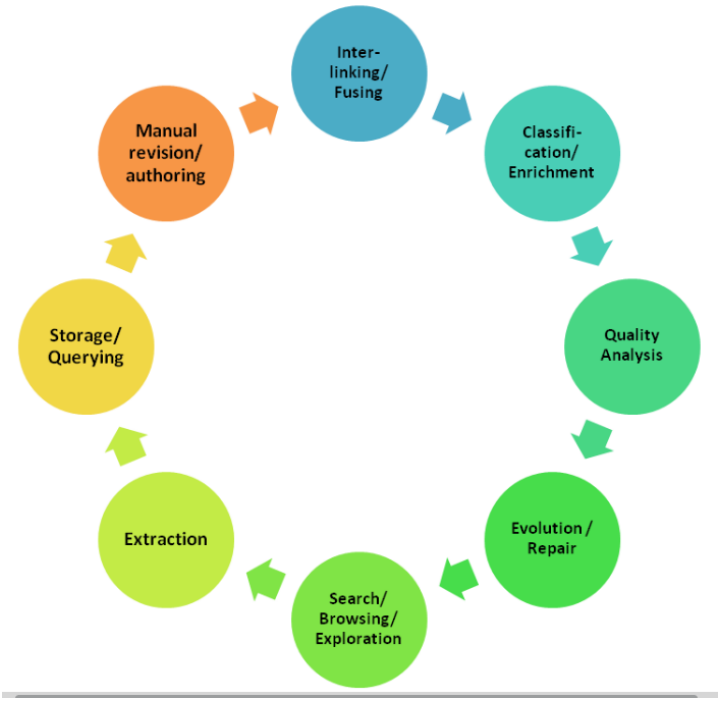


Figure 1.1.: Linked Data life-cycle.

making links, so that a person or machine can explore the Web of Data[5]. A client holding its own data can link it to other, related information by browsing the link-based structure of such Web of Data. According to the original definition, links are aimed at creating browsable graphs. Formally, a graph G is called *browsable* if, for the URI of any node in G, if I look up that URI I will be returned information which describes the node, where describing a node means: returning all statements where the node is a subject or object; and describing all blank nodes attached to the node by one arc.

When data is stored in two documents, this means that any RDF statement which relates things in the two files must be repeated in each. However, a set of of completely browsable data with links in both directions can be completely consistent only if coordination is applied, especially if different authors or different programs are involved. Such coordination is not applicable to the scale of the Web of Data, thus the Linking phase necessary brings compromises over data consistency. Tim Berners-Lee suggests to decide when to create links and in which direction, relying on common sense and practicality.

The *Pursuit of Context* is the fusion point between the current Web and its evolution into a structured Web of Linked Data. In fact, let's consider *context* as applied to terms

[5]The Web of Data as opposed to the Web of Documents, refers to the approach of publishing information on the Web that are structured and thus can be automatically processed by machines.

within documents. Terms are usually disambiguated by their *context*, i.e., the other terms surrounding the term of interest, or the overall topic of the document. In today's primarily Document centric Web, we cannot unravel *context* from existing Web content without incorporating powerful disambiguation technology into an *Entity Extraction* process. For pursing any entity extraction and disambiguation endeavour we need a lookup backbone that exposes *Named Entities* and their relationships to *Subject Matter Concepts*. Thus, when looking at the broad subject of the Semantic Web, we can also look at *context* as the vital point of confluence for the Data oriented (Linked Data) and the "Linguistic Meaning" -oriented perspectives. *Context* may ultimately be the foundation for the fourth *Web Interaction Dimension* where practical use of AI leverages a Linked Data Web substrate en route to exposing new kinds of value [Idehen, 2008].

Context landlocked within literal values offers little gain endowed with platform-specific identifiers. The ability of Web users to discern or derive multiple perspectives from the base context will be lost, or severely impeded at best. The shape, form, and quality of the lookup substrate that underlies semantic tagging services, ultimately affects "context fidelity" matters such as Entity Disambiguation. Instead *context* through the use of ontologies in the Semantic Web, is provided through a series of reference subject concepts that can be related at the class level with external data and ontologies. The "context" here acts more at how to relate large datasets or data spaces to one another. Subject concepts provide some fixed reference points in a global knowledge space for saying two things "are about" (the predicate is isAbout) the same concept. In other words, a "context" is provided for saying that two disparate resources are asserted to be about the same topic or subject.

We bridge the notion of context as applied to documents and terms to the notion of context as the situation in which the user's actions sensed by sensors occur (these two notions are described above). In particular, we use the first in order to define the latter via using Linked Sensor Data. This Thesis core contribution consists of demonstrating how this approach can lead to identify the relevancy of a sensor for a given task, adapt the network to changes in mobile environments and improve the sensor discovery for average users. At the same time, we also propose a method to guarantee the uptake of our own solution.

1.2. Research Questions

In this Thesis we address the challenge of identifying what information sensor systems should collect, how to process that information, correlate and cross-reference with multiple other sources, and produce relevant responses. This involves taking possibly millions of nodes of what we call Big Data and place it in a small context that Scoble et al. [Scoble and Israel, 2013] refer to as *Little Data*. However, far from suggesting a unique approach to create context in any scenario, we only aim to improve the decision making for our own proof-of-concept applications. Our research focus is driven by the research questions introduced below.

Sensor output can assume different meanings and trigger different actions depending on the context in which it occurred. For instance, a temperature of 4 degrees Celsius may be too low for a private household and require to trigger the action of switching on the heather. However, the same temperature observed in a food storage room would be just enough and could maybe require to trigger actions to get it even lower. Moreover correlation between different data must be created under specific criteria for correctness. It will then be difficult to assess such correctness because of a lack of a common framework. How can information from different sources correctly correlate and cross-reference?How can we evaluate the correctness of such correlation? This led to the following first research question.

Q 1. Context. *How can contextual information be used to enrich sensor data?*

Contextual information often lies on platforms which use ad-hoc solutions for sensor communication. Early research in ad-hoc wireless sensor networks focused on networks in which all nodes use identical software and hardware. This homogeneous architecture is attractive because it is resilient to individual failures. However, that does not match the case of real and especially mobile deployments, where networks are necessarily heterogeneous. This heterogeneity hinders the communication between sensors and, ultimately, the discovery and correlation of sensor resources from different platforms. Creating correlations between data from distinct platforms requires the system to communicate properly with the other external platforms. The system has to correctly interpret the data stored in the external platform to decide whether or not to create a correlation with it. This involves defining an approach for collecting, modelling and processing specific metadata for the purpose of cross-platform communication. Which information about sensors should we collect to enable cross-platform communication? How should such

information be modelled to enable cross-platform communication? How should such information be processed to enable cross-platform communication? Sensor data includes the value sensed, the unit of measurement, the observed property (phenomenon) and the date and time of the observation. Such information is usually annotated. How can the annotation process be improved for average users in terms of learnability - facilitate the accomplishment of basic tasks for users who deal with the proposed semantic annotation for the first time - efficiency - facilitate the task performance after the users have learnt the proposed annotation - memorability - facilitate the re-establishing of proficiency even after a period of not using the proposed annotation - errors - prevent the users to make severe errors and allow for the errors to be recovered easily - satisfaction - make the proposed annotation process a pleasant experience? Also, for such readings to be reusable across platforms, we need to define a different approach by addressing the second research question.

Q 2. Communication. *How can sensors communicate across different platforms without ad-hoc solutions?*

The contextual information that we identify and correlate across different platforms would cause inefficiency due to its size. Correlations should be created cautiously. If too many of them existed, it would be difficult to interpret them and expensive to browse and store the graph. We need a criteria to select only the relevant source for a specific task. Here, we then focus on a well defined task since relevancy is task-dependent. Also, given the amount of information available, the selection must be automated and evaluated for correctness. How can we automate the selection of relevant sources of information to define the Little Data? How can we evaluate it? The system should detect which information to filter out as required by the third research question.

Q 3. Relevancy. *How to identify which sensors are more relevant sources of information to define a specific small context scope - the Little Data - of interest?*

With the proliferation of *Smart Cities*, more and more live data sources such as webcam feeds and physical sensor information are publicly accessible over the Web. However, these sources are typically decoupled from normal websites, and are therefore not within the scope of traditional online search using Web search engines. A user's location-related information needs often refer to data which is only valid for very specific and short time frames. As a result, they are typically not maintained on a Web page, let alone indexed by a search engine. Sensors could provide the missing information to

traditional Web content and Web search engines, enriching the quality of the available data. We investigate how to realise this vision, with the following research question.

Q 4. Quality. *How can contextualised sensors improve the quality of traditional Web content?*

1.3. Main Contributions

In order to answer the research questions detailed above, the thesis describes 5 complementary directions for contextualising sensors: cross-platform communication between sensors; interlinking with contextual information; filtering of relevant contextual information for a given task; improving the data quality for traditional Web content; supporting the automated adaptability to changes in mobile sensor networks. The experiments that led us to these contributions produced or were initiated by sensor data and metadata automatically annotated as Linked Sensor Data. We collected this data in a dataset that we published on DataHub (a powerful data management platform) with a GNU Free Documentation License.

For a complete and detailed list of my own personal contribution to each of the following subjects, please refer to Appendix A.

C 1. Interlinking with contextual information. *Search for external data eligible to be linked and evaluate the linking.*

Since part of the metadata can be context-related, we define and implement an algorithm which finds external data to link based on client-defined criteria. Such service is also provisioned by LD4S. We also enable the evaluation and rating of the created link. This part of the work initiated from annotated and linking Pachube (now called Xively) sensor streams with LOD resources [Leggieri et al., 2011a] and continued with a generalisation to any sensor metadata [Leggieri et al., 2015b]. The finalised algorithm and its evaluation are described in [Leggieri et al., 2013a].

C 2. Cross-platform communication between sensors. *Collection, modelling and processing of sensor data and metadata from distinct platforms.*

We identified which information to collect, modelled it using ontologies and process it as OWL. To demonstrate semantic annotations as a solution to the challenges of enabling seamless sensor communication mentioned above, we describe the design process and

implementation of *Linked Data for Sensors (LD4S)*, a web service generating linked data for sensors. Given the learning curve for semantic technologies, this tool has received attention as a facilitator for the uptake of semantic approaches by developers from different domains. We have written about developing LD4S in [Leggieri et al., 2013a] and in more detail in [Leggieri et al., 2015b].

C 3. Filtering of relevant contextual information for a given task. *Automate the relevancy-based selection and evaluation of contextual information performed during the interlinking phase.*

Since relevancy is highly dependent on the task at hand, we focus on a specific task: daily activity logging. It is an increasingly popular task due to monitoring interests coupled with the pervasiveness of tracking technologies in the everyday life. We design and implement an algorithm that predicts which sensors are going to provide relevant data in the close future for the activity that is currently being logged. The evaluation reported outstanding results which we detail in [Leggieri et al., 2015a].

C 4. Improving the data quality for traditional Web content. *Bridging the gap between Sensor Web and traditional Web in which the latter lacks of short-lived but extremely interesting information provided to the average user.*

Relying on LD4S as a backend, we developed *G-Sensing* as a frontend, a browser plugin that injects live data from sensors into Google search results. We evaluated its performances and suitability of our approach in [Leggieri et al., 2015b].

1.4. Thesis Outline

Our approach can be used by systems to autonomously take better decisions. We demonstrate our claims of improved decision making by building and evaluating a system that predicts the relevancy of a sensor with respect to an activity logging task. Here we demonstrate how better decisions are taken on selecting which sensors to query to accomplish the activity logging task. Another sample application is a system that automatically recognises and identifies a new sensor entering the context and a system that suggests the best transportation route. Figure 1.2 shows a mapping between our contributions and the core chapters in which we describe them.

The remainder of the thesis is structured as follows:

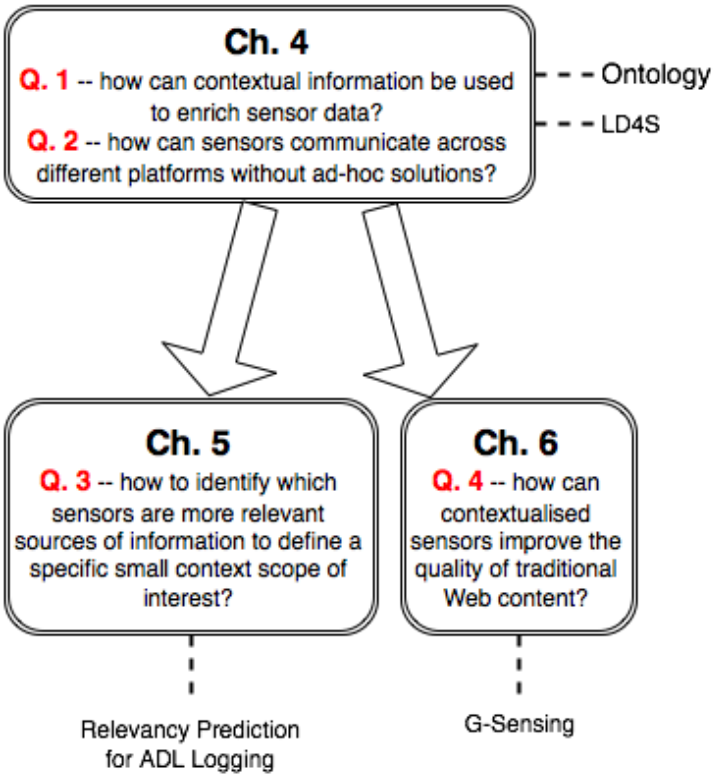


Figure 1.2.: Mapping between core chapters, research questions and main contributions.

Part II — Foundations

We begin by setting the foundations for our work. This part is comprised of two chapters.

Chapter 2 describes the terminology and introduces some of the related work about the Sensor Web and the Semantic Sensor Web with Linked Data principles.

Chapter 3 contains an extensive survey of the *Internet of Things (IoT)* phenomenon, from the visionaries that inspired it, to discussing its architectures, and its applications, the common points and what they do differently. This chapter also describes in more detail the SPITFIRE framework to which this thesis mainly contributed.

Part III — Core

The core of the thesis presents our work, divided in five chapters: the first two reflect the approaches used to solve our two main research questions, and the last three demonstrate how our approach leads to increased relevancy and quality of the data.

Chapter 4 describes our solution to the first and second research question **Q 1.** — how can contextual information be used to enrich sensor data?
Q 2. — how can sensors communicate across different platforms without ad-hoc solutions?

Chapter 5 describes a system that uses of the work from the first two chapters for automatically filtering contextual data according to its relevancy for the task of daily activity logging. This is our solution to the third research question **Q 3.** — how to identify which sensors are more relevant sources of information to define a specific small context scope - the *Little Data* - of interest?

Chapter 6 describes a system that uses of the work from the first two chapters to improve the quality of traditional Web content. This addresses the research question **Q 4.** — how can contextualised sensors improve the quality of traditional Web content?

Part IV — Conclusion

Chapter 7 contains the conclusion of the work, reiterating the contributions and how they answer our research questions. We discuss some of our results and the insights we gathered from the work. We also outline ideas for future research.

Part II.

Foundations

Chapter 2.

Background

In this Chapter we will depict the evolution of the Web from its origins to the Sensor Web, Internet and Web of Things (IoT and WoT). We will particularly highlight the latter, detailing its vision, architecture and main challenges. A survey on the research efforts in that field will follow. In parallel to the Sensor Web and Web of Things phenomenon, the Semantic Sensor Web has developed. It has always proposed itself as a solution to the interoperability issue thanks to the adoption of Semantic Web technologies. However, for several reasons that we will analyse, it hasn't yet become mainstream in real sensor deployments. We dedicate Chapter 3 to the Semantic Sensor Web since this is the approach that we adopt and since semantic technologies have unique features that deserve to be better detailed.

The first attempt to let two computers communicate with each other through a network [Olifer and Olifer, 2005] took place in the 1980s. The TCP/IP stack was introduced and commercial use of the Internet started. The World Wide Web (WWW) became available in 1991 which made the Internet more popular and stimulate the rapid growth. At around the same time, advances in micro-electro-mechanical systems (MEMS) technology, wireless communication and electronics enabled the development of low cost, low power and multi functional miniature devices called *sensor nodes*. A sensor is defined from an engineering point of view as a "device that converts a physical, chemical, or biological parameter into an electrical signal" [Bermudez et al., 2009]. Sensors are able to sense, compute and communicate wireless in short distance (up to 150 m). Common examples include sensors for measuring temperature (i.e., a thermometer), wind speed (an anemometer) conductivity, or solar radiation.

An aggregation of sensors attached to a single *platform* is called a "sensor system" [Stasch et al., 2009], e.g., temperature and humidity sensors attached to a weather

station. A sensor or a sensor system may be abstracted as a *sensor resource*. A number of spatially distributed and communicating sensor resources is considered a "sensor network" [van Zyl et al., 2009].

Sensor nodes enabled ubiquitous sensing capabilities that led to the rise of the *Ubiquitous Computing* discipline. Ubiquitous Computing's goal is the creation of a smart environment, defined by Weiser in his ground-breaking paper [Weiser, 1991] as "the physical world that is richly and invisibly intervowen with sensors, actuators, displays and computational elements, embedded seamlessly in the everyday objects of our lives and connected through a continuous network".

Due to advances in sensor technology, sensors are becoming more powerful, cheaper and smaller in size, which has stimulated large scale deployments. As a result, today we have a large number of sensors already deployed and it is predicted that the numbers will grow rapidly over the next decade [Sundmaeker et al., 2010a].

Given the large number of sensor manufacturers and differing accompanying protocols, integrating diverse sensors into observation systems is not straightforward. This interoperability issue is one of the major challenges along with scalability. The prototypes and system solutions developed in the past two decades always dealt with a limited number of data sources, e.g., physical (hardware) or virtual (software) sensors. Such solutions can not scale with the current growing trend in the amount of sensors available.

The Sensor Web (Section 2.1), Semantic Sensor Web (Chapter 3), Internet of Things (Section 2.2) and Web of Things do not represent solutions to the scalability and interoperability issues by themselves. They are mere labels that identify the current growing trend of intercommunicating ubiquitous devices as depicted in Figure 2.1 while each focusing on a different aspect and set of technologies. We will explain the phenomenon behind each of those labels in the next sections.

Figure 2.1.: Evolution of the Internet in five phases from a single M2M communication to the connection of people and things.

.

2.1. Interoperability and Sensor Web

A coherent infrastructure is needed to treat sensors in an interoperable, platform-independent and uniform way. The concept of the *Sensor Web* reflects such a kind of infrastructure for sharing, finding, and accessing sensors and their data across different applications. *Semantic Web* technologies offer an improvement over the XML-based Sensor Web solutions. They all particularly address the interoperability issue. Here we mention briefly the Semantic Web in the context of the interoperability issue. However, we will expose a broader and more detailed analysis of the Semantic Sensor Web paradigm in Chapter 3.

First described by Delin et al. in 1999 [Delin et al., 1999], a Sensor Web was considered as an autonomously organised wireless sensor network which can be deployed to monitor environments. As a smart macro instrument for coordinated sensing [Delin, 2001], Delins Sensor Web concept consists of sensor nodes which not only collect data, but also share their data and adjust their behaviour based on that data. Thereby, the term *Web* within Delins *Sensor Web* relates to the intelligent coordination of the network rather than the World Wide Web (WWW) [Teillet, 2008]. Later, the meaning of "Sensor Web" changed and it was more and more seen as an additional layer integrating sensor networks with the WWW and applications [Gibbons et al., 2003a, Shneidman et al., 2004, Moodley and Simonis, 2006].

Today, the notion of "*Sensor Web*" has been largely influenced by the developments of the Open Geospatial Consortium's Sensor Web Enablement (SWE) initiative. The Open Geospatial Consortium (OGC) started the SWE working group in 2003 in a standardisation effort to address the scaling issue of sensor networks. Due to the large variety of sensor protocols and sensor interfaces, most applications were still integrating sensor resources through proprietary mechanisms, instead of building upon a well-defined and established integration layer. This manual bridging between sensor resources and applications leads to extensive adoption effort, and is a key cost factor in large-scale deployment scenarios [Aberer et al., 2006]. SWE developed a suite of standards which can be used as building blocks for a Sensor Web. SWE defines [Botts et al., 2006] the term Sensor Web as "Web accessible sensor networks and archived sensor data that can be discovered and accessed using standard protocols and application programming interfaces". It is defined as an infrastructure which enables an interoperable usage of sensor resources by enabling their discovery, access, tasking, as well as eventing and alerting within the Sensor Web in a standardised way. Thus, the Sensor Web is to sensor

resources what the WWW is to general information sources: an infrastructure allowing users to easily share their sensor resources in a well-defined way [Nittel, 2009]. It hides the underlying layers, the network communication details, and heterogeneous sensor hardware, from the applications built on top of it. The Sensor Web is an integration of three layers as shown in Figure 2.2 Sensor Layer, Communication Layer and Information Layer:

- Sensor Layer. Sensors can measure physical, chemical, and biological properties. They can be classified as "in situ" or "remote" according to the target that is being sensed. In situ sensors have higher accuracy and better resolution; while remote sensors have better spatial resolution.
- Communication Layer. This layer includes controls for transmitting data between the Sensor Layer and the Information Layer. Examples are satellites, radio networks or cellphones. In the Sensor Web this layer is hybrid, forwarding data to the Information Layer through the Internet.
- Information Layer. This layer stores, disseminate, exchange and analyse sensor resources such as sensors, their locations and their measurements. In the Sensor Web interoperability within the Information Layer is critical to enable a seamless access to sensor resources.

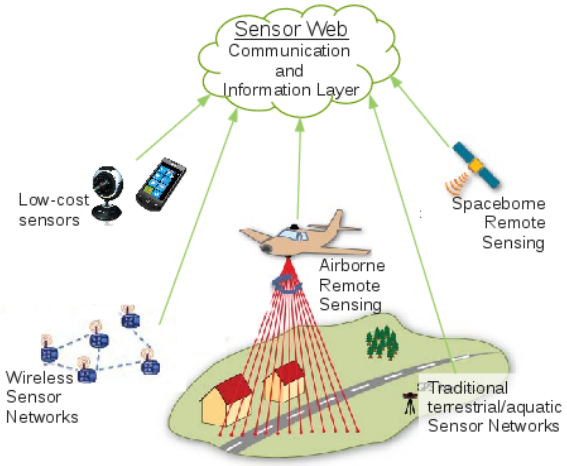


Figure 2.2.: Sensor Web Layers.

Platforms . The communication of smart things[1] has been studied for decades. Several different technologies and standards have been proposed in this area. Different manufacturers produces smart things which are mostly isolated from each other, i.e., they can not communicate with each other because of a lack of shared or standardise communication protocols and languages. There is a lack of interoperability between smart things that has to be enabled. Intercommunication would be only part of the steps required for such enablement. Here we review some of the major technologies which have so far attempted to address the interoperability issue.

UPnP (Universal Plug and Play) is a suite of networking protocols extended from the idea of the original Plug and Play to a networked system context. It was promoted by the UPnP forum2 mainly for personal networks devices to discover each others presence and further to establish connections on the network. UPnP is based on established protocols and standards, such as TCP/IP[2], UDP[3], HTTP[4], HTTPU (HTTP over UDP), SOAP[5], WSDL[6], etc. Currently, UPnP[7] is the most popular solution for personal network implementation. However, UPnP has several drawbacks [Duquennoy et al., 2009]:

- There is no authentication protocol proposed for UPnP. Any devices are allowed to configure the other devices of the personal network, without any user control, resulting in a critical security issue when the smart things are available on the Internet.
- UPnP is not strictly standardised as some UPnP devices are based unstandardised protocols such as HTTPU, restricting its universal interconnection somehow.
- UPnP is inapplicable to some resource-constrained devices because it normally uses a lot of heavy protocols (e.g., SOAP, WSDL, etc.) involving complex processing.

[1] Smart things is a synonym for Internet-connected-objects, i.e., objects that are connected to the Internet.

[2] Transmission Control Protocol/Internet Protocol (TCP/IP) refers the suite of communications protocols used to connect hosts on the Internet.

[3] User Datagram Protocol (UDP) a connectionless protocol that, like TCP, runs on top of IP networks. It is used primarily for broadcasting messages.

[4] HyperText Transfer Protocol (HTTP) is the underlying protocol used by the World Wide Web. It defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

[5] Simple Object Access Protocol (SOAP) is a lightweight XML-based messaging protocol used to encode the information in Web service request and response messages before sending them over a network

[6] Web Services Description Language (WSDL) is an XML-formatted language used to describe a Web service's capabilities as collections of communication endpoints capable of exchanging messages.

[7] Universal Plug and Play (UPnP) is a networking architecture that provides compatibility among networking equipment, software and peripherals of the 400+ vendors that are part of the Universal Plug and Play Forum.

JXTA is a set of protocols to connect heterogeneous devices into the same network for peer-to-peer application design. Unfortunately, they have neither been standardised nor widely adopted by the industry.

One trend is integrating the devices into the Web. It has been found that the web severs can be built on highly constrained devices occupyingonly a few KBs in size [Duquennoy et al., 2009, Agranat, 1998, Lin et al., 2004]. This is relevant in a smart things scenario where object are not only connected to the Web but also directly providing services through their own local web servers over the Web. Using the free, open, flexible, and scalable Web as the universal platform to integrate smart devices outperforms all other solutions mentioned earlier in terms of easiness, flexibility, customisation and security. This idea has attracted much attention from both academia and industry, especially with the emergence of the IoT. Web browsers are available on almost any platform, from computers to PDAs, smart phones, and tablets, and have become the de facto standard user interface to a variety of applications. The Web-enabled applications can be accessed from any location provided there is an Internet connection. Applied to embedded systems, web technologies can offer platform-independent interfaces such that the end-user does not need to install specific software and drivers for different devices. Also, developers do not have to tediously develop different software and drivers targeting different platforms. The Web provides a one-for-all solution. Furthermore, devices become programmable providing great opportunities to create more innovative and powerful applications. However development - especially composition - of applications that run on top of those physical devices is still a cumbersome process as it requires extensive expert knowledge (e.g. specific APIs in a specific programming language) about all different physical devices. This more or less constrains development of smart things based services. Fortunately, existing web technologies (e.g. mashup), which previously targeted for cyber-world web services can be reused for application development with the participation of physical smart things provided that they can be abstracted as web services. By reusing existing web technologies, the expenses of additional infrastructure and overall implementation time can be minimised. These technologies can promote the progress of IoT significantly.

We can classify the systems realised so far within the Sensor Web area in 3 categories:1. streaming query systems, 2. sensor middleware/querying sensor networks 3. hierarchical and receptor-based systems. Each category is described in the following sections.

Streaming Query Systems

Streaming Query Systems are applications that monitor continuous streams of data. For example military applications that monitor readings from sensors worn by soldiers, financial analysis applications that monitor streams of stock data reported from various stock exchanges, and audio-visual departments that monitor the location of borrowed equipment. These applications substantially differ from conventional business data processing. Monitoring applications are very difficult to implement in traditional DBMSs[8] because basic computational model is wrong. Traditional DBMSs have a Human Active Database Passive model (HADP) while monitoring applications often require a database active human passive model (DAHP). HADP means DBMS is a passive warehouse storing a large collection of data elements and that humans initiate queries and transactions on this warehouse. In DAHP model the DBMS get data from external sources rather than from humans issuing transactions and alert humans when abnormal activity is detected.These applications require storing some history of values reported in a stream. They are trigger oriented, have to deal with incomplete information, and have real time requirements. These systems handles large numbers of continuous queries over high-volume and highly variable data streams. The main Streaming Query Systems from the state of the art are: 1. **Aurora* and Medusa**: two distributed stream processing systems [Cherniack et al., 2003]. Aurora* is designed for a single administrative domain. It addresses QoS[9] and dynamic operator repartitioning to achieve load balancing and fault-tolerance. Medusa arranges the single-site Aurora data stream processors in a loosely federated network mediated by agoric principles, i.e., it uses economic principles to manage and share the load. The communications infrastructure is an overlay network, layered on top of the underlying Internet substrate. 2. **PIER**: a peer-to-peer information exchange and retrieval query engine that scales up to thousands of participating nodes [Huebsch et al., 2003, Huebsch et al., 2005]. Built on top of a distributed hash table, it allows to query Internet data in situ, without the need for database design, maintenance or integration. 3. **Sophia** is a framework which collects and reasons over data from distributed sets of sensors [Wawrzoniak et al., 2004]. A declarative programming environment evaluates logic distributed statements about the system.

Query streaming systems can be further classified as 1. Centralised. In the recent years, in other application domains such as network monitoring or telecommunications,

[8] A Data Base Management System (DBMS) is a collection of programs that enables you to store, modify, and extract information from a database.

[9] Quality of Service (QoS) is a networking term that specifies a guaranteed throughput level.

data stream processing has received huge attention. Because of that, a rich set of query languages and query processing approaches for data streams exist. They were initially designed for centralised architectures. 2. Distributed. Stream processing systems operate in a distributed fashion because stream oriented systems are inherently geographically distributed and support scalable load management and higher availability. 3. Distributed and Declarative. In this approach it can be viewed as a multi-user distributed expression evaluator in which sensors and actuators form the edge level devices. This approach has several advantages in managing and controlling a complex, federated, and evolving network. First, a declarative logic language provides a natural way to express the kinds of statements that are common to this application domain, through temporal and positional logic rules, facts and expressions. Second, distributed evaluation of such logic expressions provides many opportunities for performance optimisation yielding an efficient system [Wawrzoniak et al., 2004].

Sensor Middleware/ Querying sensor networks

Sensor Web requires a method to change the behaviour of a sensor network dynamically. This saves power consumption in the network. In network-level abstractions, a sensor network is treated as a whole and is regarded as a single abstract machine. The entire network is considered as a virtual database system. Sensor networks are often for collecting sensing data, the database approach is one solution. Database systems allow users to issue queries in a declarative SQL-like language. Database abstraction provides a simple and easy-to-use interface. However, it is suitable only for describing query operations to a sensor network. Although Cougar [Bonnet et al., 2001] and TinyDB [Madden et al., 2003] extend SQL so that users can express continuous sensing tasks, they are still not expressive enough to cover all sorts of sensor network applications. They are described as follows, along with other major middlewares from the state of the art: 1. **TinyDB**: query processing system to extract information from a TinyOS sensor network using power-efficient in-network algorithms [Madden et al., 2003]. Its core service is data aggregation. The data aggregation is distributed and executed in the sensor network in a time and power-efficient manner. 2. **Cougar**: a middleware that assigns tasks to sensor networks using declarative queries [Bonnet et al., 2001]. It makes the node-level implementation transparent to the user and tries to reduce the amount of data to be collected for energy-efficiency by pre-selecting subsets of the nodes. 3. **SINA**: made of modules that run on each sensor node to provide adaptive organisation of sensor information and to facilitate querying, event monitoring and task-

ing [Srisathapornphat et al., 2001]. Sensor nodes are automatically clustered in order to support energy efficiency and operation scalability. The kernel uses a spreadsheet-based storage system for querying and monitoring. Each logical datasheet is uninque and represents a sensor node attribute. Each sensor node maintains a whole datasheet, so one can view the sensor network as a collection of datasheets. 4. **MiLAN** is a framework that processes queries based on QoS requirements [Heinzelman et al., 2004]. QoS is defined by the level of certainty about an attribute, based on the assumption that each sensor can measure some basic attributes with predefined reliability. In response to a query, MiLAN creates an execution plan, which specifies the source nodes and the routing tree that satisfy the QoS requirement while maximising energy efficiency.

Hierarchical and Receptor-Based Systems

Hierarchical and receptor-based systems aim at managing and querying the data produced by sensors, both physical and virtual. These systems have assumed topologies similar to the high fan-in systems but there are significant differences. Theses are the system in which large number of receptor exist at the edge of the network that collect raw data readings. These heterogeneous edge devices produce data which is aggregated locally with data from other nearby devices. That data will be further aggregated within a larger area, and so on. This kind of arrangement is called *high-fan-in* system [Franklin et al., 2005]. This hierarchical bowtie shape arises because of **two** reasons. **First**, data cleaning, filtering and aggregation close to the edges will save the bandwidth and processing costs. **Second**, many of target applications naturally have a hierarchical structure. The main platforms under this category can be described as follows:

1. **IrisNet** is a framework where a set of receptors feed into a core of sensor nodes running a distributed database [Gibbons et al., 2003b]. It introduces a two tire architecture called Sensing Agents (SA) and Organising Agents (OA). SAs implement a generic data acquisition interface which allow to access different sensors. OAs are organised in groups. Each group includes all the OAs particularly selecting those sensor data relevant to a specific service.

2. **SensorMap** is a web portal for real-time real-world sensor data [Nath et al., 2006]. It archives sensor data via a web service and allows to search for them, optimising the

query efficiency by indexing static sensor metadata. A GUI lets users query data sources and view results on a map.

3. **Mobile Web Services Framework** is a web service framework based on a mobile sensor network for ubiquitous environment monitoring [Kim et al., 2008]. Web services complying with the Service Oriented Architecture process and store the transmitted sensor data.

4. **Sensor Web Language**(SWL) is an extensible, object oriented language supporting robust message passing among various components [Nickerson and Lu, 2004]: a) sensor nodes b) gateway nodes c) communication computers d) LINUX server and web browser.

5. **OSRE** is a declarative application for a sensor-rich environment. Given a declarative query that expresses a goal for the environment to report back or react to, the system generates a workflow of semantic services. Then it creates a task graph where each service is assigned to a node in the network according to the service workflow, the requirements of the network resources and their availability. Each node executes the assigned services and sends back the result to the user or to the cache.

Service Oriented Sensor Web is a middleware and programming environment for creating, and accessing sensor services through the Web [Xingchen and Rajkumar, 2007]. Sensors are exposed on the Web as SOA web services.

6. **Sensor Web Enablement** (SWE) consists of a set of standard services to build a unique framework for discovering and interacting with web-connected sensors and for assembling and utilising sensor networks on the web [Botts et al., 2006]. It is defined by the Open Geospatial Consortium (OGC). SWE focuses on developing standards to enable the discovery, exchange, and processing of sensor observations, as well as the tasking of sensor systems. The vision is to define and approve the standards foundation for "plug-and-play" Web-based sensor networks.

All of the above services are useful for different aspects of sensor data processing, and this may be done in different ways based on the underlying scenario. For example, the discovery of the appropriate sensors is a critical task for the user, though it is not always easy to know a-priori about the nature of the discovery that a user may request. For example, a user may be interested in discovering physical sensors based on specific criteria such as location, measurement type, semantic meta- information etc., or they may be interested in specific sensor related functionality such as alerting [Jirka et al., 2009a].

Either goal may be achieved with an appropriate implementation of the SML module [Broering et al., 2011a, Jirka et al., 2009a]. Thus, the specific design of each module will dictate the functionality which is available in a given infrastructure.

To realise the Sensor Web vision, SWE incorporates models for describing sensor resources and sensor observations. Furthermore, it defines web service interfaces levering the models and encodings to allow accessing sensor data, tasking of sensors, and alerting based on gathered sensor observations. The SWE specifications provide the functionality to integrate sensors into Spatial Data Infrastructures (SDI). The integration of sensor assets into SDIs makes it possible to couple available sensor data with other spatio-temporal resources (e.g., maps, raster as well as vector data) at the application level, which maximises the information effectiveness for decision support. Due to this integration, Sensor Webs and the geosensors they comprise represent a real-time link of Geoinformation Systems (GIS) into the physical world. Thereby, geosensors are defined as sensors delivering an observation with georeferenced location [Stasch et al., 2009].

The **Semantic Sensor Web** (SSW) is a framework for providing enhanced meaning for sensor observations to enable situation awareness [Sheth et al., 2008]. It enhances the SWE standards by adding semantic annotations to the SWE sensor languages. This enhanced access to sensor data and bridges the gap between the primarily syntactic XML-based SWE metadata and the RDF/OWL-based metadata standards from the Semantic Web. It provides an environment for enhanced query and reasoning within the sensor domain by incorporating OGC and W3C standardisation efforts into a SSW. We will discuss further in Chapter 3 the SSW evolution into the Semantic Web of Things, as it constitutes the foundation of this thesis contribution. The Internet of Things is the key that started such evolution and we describe it in the following Section 2.2.

2.2. Scaling and the Internet of Things

Internet of Things (IoT) systems are expected to deal with billions of sensors that are all connected to the Internet. This is in contrast with the limited-scope systems developed during the past decade. In such a situation, the available information must be filtered. Context-awareness plays a critical role to support such necessary data filtering process.

Kevin Ashton [Ashton, 2009] firstly coined the term "*Internet of Things*" (IoT) in 1998. Then, the MIT Auto-ID centre presented their IoT vision in 2001 [Brock, 2001]

while the International Telecommunication Union (ITU) in 2005 formally introduced the IoT as a new discipline [Union, 2005].

The IoT vision promises to create a world where all the objects (also called smart objects [Kortuem et al., 2010]) around us are connected to the Internet and communicate with each other with minimum human intervention [Le-Phuoc et al., 2009]. The ultimate goal is to create a better world for human beings, where objects around us know what we like, what we want, and what we need and act accordingly without explicit instructions [Dohr et al., 2010]. The process of machines communicating with one another, is also referred to as the *Machine-to-Machine* (M2M) paradigm. This requires tremendous data-centric capabilities, which is the primary medium of communication between the different entities. Therefore, the ability to securely and privately collect, manage, index, query and process large amounts of data is critical.

Definitions . The IoT vision is very broad and the research is still in its infancy. The IoT has different interpretations due to the diversity of the communities involved in inherently cross-disciplinary efforts between sensor networking, data management and the World Wide Web. Such diversity also reflects in the technologies developed [Sundmaeker et al., 2010a]. Therefore, there are no any standard definitions for IoT. Some definitions given by researchers are as follows:

- A pervasive-oriented definition [Giusto et al., 2010, Atzori et al., 2010] depicts the IoT as "the pervasive presence around us of a variety of things or objects such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, mobile phones, etc. which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals".
- A things-centered definition [Lu and Neng, 2010] focuses on inter-objects communication stating that "things have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts.
- A syntactic definition [of the ETP EPOSS, 2005] describes "the semantic origin of the expression is composed by two words and concepts: Internet and Thing, where Internet can be defined as the world-wide network of interconnected computer networks, based on a standard communication protocol, the Internet suite (TCP/IP), while Thing is an object not precisely identifiable. Therefore, semantically, Internet

of Things means a world-wide network of interconnected objects uniquely addressable, based on standard communication protocols".

- A connection between objects and humans is highlighted by [Guillemin and Friess, 2009]: "the Internet of Things allows people and things to be connected Anytime, Anyplace, with Anything and Anyone, ideally using Any path/network and Any service". The latter [Guillemin and Friess, 2009] probably offer a better depiction of IoT's broad vision.

- According to Cluster of European research projects on the Internet of Things [Sundmaeker et al., 2
"*things* are active participants in business, information and social processes where they are enabled to interact and communicate among themselves and with the environment by exchanging data and information sensed about the environment, while reacting autonomously to the real/physical world events and influencing it by running processes that trigger actions and create services with or without direct human intervention".

- According to Forrester [Belissent, 2010], a smart environment "uses information and communications technologies to make the critical infrastructure components and services of a city's administration, education, healthcare, public safety, real estate, transportation and utilities more aware, interactive and efficient.

All the definitions above are valid and can be summarised by defining the IoT as a worldwide network based on the current Internet and its suite of protocols and applications - in which any kind of object is inter-connected and remotely accessible.

Application Domains . The IoT interconnection and communication between everyday objects enables new scenarios in several application domains. Such domains can be classified in **three** categories based on their focus [Atzori et al., 2010, Sundmaeker et al., 2010a]: industry, environment and society.

Examples for the industry category include automotive, supply chain management [Chaves and Dec
aerospace, aviation, transportation and logistics [Chen et al., 2010]. Among the enabled applications in the environment domain there are recycling, agriculture and breeding [Burrell et al., 2004, Lin, 2011], disaster alerting. Scenarios for the society category include healthcare [Wang et al., 2011], home or office automation [Chong et al., 2011], telecommunication, ticketing, entertainment. The same scenarios were differently classified by Asin and Gascon [Asin and Gascon, 2012] into **twelve** categories: smart animal

farming, smart cities, retail, logistics, smart environment, smart water, smart metering, security and emergencies, smart agriculture, eHealth, industrial control and building automation.

The magnitude of each application is better described by the following statistics. It is estimated that there about 1.5 billion Internet-enabled PCs and over 1 billion Internet-enabled mobile phones today. These two categories will be joined with Internet-enabled devices (smart objects [Kortuem et al., 2010])) in the future. By 2020, there will be 50 to 100 billion devices connected to the Internet [Sundmaeker et al., 2010a]. According to BBC Research [Forecasting, 2011], the global market for sensors was around $56.3 billion in 2010. In 2011, it was around $62.8 billion. The global market for sensors is expected to increase to $91.5 billion by 2016, at a compound annual growth rate of 7.8%.

Web of Things . Part of the research efforts aim at reusing existing Web technologies and standards to realise the IoT vision. This trend has been labelled the *Web of Things* (WoT). In particular, Web services have resulted to be indispensable for creating interoperable applications on today's Internet. We could abstract over smart things equipped with embedded web servers to consider the things themselves as web services seamlessly integrated into the existing Web. The WoT envisions a collection of web services that can be discovered, orchestrated and executed. This approach exposes the synchronous functionality of smart objects through a REST interface. The REST interface defines the notion of a resource as any component of an application to which it is worth being uniquely identified and linked. On the Web, the identification of resources relies on Uniform Resource Identifiers (URIs), and representations retrieved through resource interactions contain links to other resources [Guinard et al., 2011]. This means that applications can follow links through an interconnected web of resources. Similar to the web, clients of such services can follow these links in order to find resources to interact with. Therefore, a client may explore a service by browsing it, and the services will use different link types to represent different relationships.

WoT enlarges the types of Web services provided from offering only cyber-world services to also include physical-world services. The reuse and adaptation of existing Web technologies also yields to higher flexibility, customisation and productivity.

2.3. Common Vision

Each of the visions described so far - under the different labels of Sensor Web, Semantic Sensor Web, IoT and WoT - have a slightly distinct emphasis on different parts of the data-centric pipeline. However, as identified by Atzori et al. [Atzori et al., 2010, Atzori et al., 2010], IoT can be realised under **three** main visions around which most of the research related to the union of Sensors and Web is focused. These are: 1. internet-oriented (middleware) 2. things-oriented (sensors) 3. semantic-oriented (knowledge).

1. **Things-oriented Vision**. The Electronic Product Code (EPC) [of Business, 2009] and the RFID technology for unique product identification and tracking [Welbourne et al., 2009] are the key enablers. However, more sophisticated sensor technology is usually required in conjunction with RFID in order to collect and transmit useful information about the objects being tracked. For instance, Intel's Wireless Identification and Sensing Platform (WISP) [Corp., 2014] is powered by standard RFID readers but enriched to measure additional physical quantities such as temperature.

2. **Internet-oriented Vision**. Inline with the reuse of Web technologies supported by WoT [Guinard and Trifa, 2009] and by the IPSO alliance [Alliance, 2014], this vision mainly focuses on the IP protocols for enabling Internet-connected smart objects. Since, each of the IoT devices would require its own IP address, the aims is to adapt the internet infrastructure to accommodate the constantly growing number of *things* which require connectivity. For instance, the new protocol IPv6 has been designed to provide a much larger addressable IP space.

3. **Semantic-oriented Vision**. The key enabler for this vision are standardised resource descriptions. These are meant to address the data management and interoperability issues brought by the heterogeneous resources available through the WoT. This vision depicts a separation between the meanings of data and the actual data itself. The semantic meaning of objects is stored separately from the data about the object itself and from the the management tools.

This type of delineation is required due to the interdisciplinary nature of the subject. However, the usefulness of IoT can be unleashed only in an application domain where the three paradigms intersect. In this thesis we rely on the achievements reached within the Internet-oriented vision. As we describe our contributions, we will also specify which Internet technologies we rely on. Furthermore, since our approach is heavily based on Semantic Web technologies, we will dig deeper into the semantic-oriented vision in

Chapter 3. Finally our contributions focus on activity tracking via objects monitoring, which is part of the things-oriented vision.

2.3.1. Trends

The popularity of different paradigms varies with time. However, we can notice that since IoT was firstly coined, its search volume has consistently increased in parallel with a decrease for the Wireless Sensor Networks' trend [Inc., 2013]. The Google's search forecast predicts such a trend to continue. In the rest of this chapter we will refer to the research area related to Sensors and WWW as **IoT**, because of its popularity.

During the past decade, IoT has gained significant attention from both academia and industry because of the capabilities [Institutes, 2011, Atzori et al., 2010] that it will provide. IoT is one of the emerging technologies in IT. It has been forecasted that IoT will take 510 years for market adoption [Inc., 2012].

2.3.2. Architecture

We identified **three** IoT components which enable seamless Ubiquitous Computing:

1. hardware: sensors, actuators and embedded communication hardware
2. middleware: on demand storage and computing tools for data analytics
3. presentation: innovative and usable visualisation and interpretation tools which can be widely accessed on different platforms and designed for different applications.

In this section, we discuss a few enabling technologies in these categories which will make up the above components. First we describe sensor networks which could be consider the building block of IoT since they enable Internet-Connected Objects (ICOs). Then we focus on the addressing schemes used to let the ICOs communicate, followed by the state of the art techniques to store, process and visualise the collected data.

Sensor Networks

Prior to the IoT, Sensor Networks (SNs) were used in limited domains to achieve specific purposes, such as environment monitoring [Mainwaring et al., 2002], agricul-

ture [Burrell et al., 2004], medical care [Malan et al., 2004], event detection [Rooney et al., 2006], structural health monitoring [Rocha et al., 2009], etc. They constitute the most essential component of the IoT. A SN is made of one or more sensor nodes which can be either homogeneous or heterogeneous. Sensor nodes and SNs themselves can communicate with each other through wired or wireless technologies and protocols. One such approach is through the Internet.

Most of the networks nowadays are wireless and called Wireless Sensor Networks (WSNs). There are several major wireless technologies used to build wireless sensor networks. For instance Bluetooth is used in Wireless Personal Area Networks (WPANs), Wi-Fi is used in Wireless Local Area Network (WLAN), WiMAX is used in Wireless Metropolitan Area Network (WMAN), 2G and 3G are used in Wireless Wide Area Network (WWAN) and GPS used in satellite networks. Sensor networks also use two types of protocols for communication: non-IP based (e.g., Zigbee and Sensor-Net) and IP-based protocols (e.g., NanoStack, PhyNet, and IPv6). The components that make up the WSN monitoring network include:

- WSN hardware (hardware category). Typically a node (WSN core hardware) contains sensor interfaces, processing units, transceiver units and power supply. Almost always, they comprise of multiple A/D converters for sensor interfacing and more modern sensor nodes have the ability to communicate using one frequency band making them more versatile [Akyildiz et al., 2002].

- WSN communication stack (presentation category). The nodes are expected to be deployed in an ad-hoc manner for most applications. Designing an appropriate topology, routing and MAC layer is critical for the scalability and longevity of the deployed network. Nodes in a WSN need to communicate among themselves to transmit data in single or multi-hop to a base station. Node drop outs, and consequent degraded network lifetimes, are frequent. The communication stack at the sink node should be able to interact with the outside world through the Internet to act as a gate- way to the WSN subnet and the Internet itself [Ghosh and Das, 2008].

- WSN Middleware (middleware category). A mechanism to combine cyber infrastructure with a Service Oriented Architecture (SOA) and sensor networks to provide access to heterogeneous sensor resources in a deployment independent manner [Ghosh and Das, 2008]. This is based on the idea of isolating resources that can be used by several applications. A platform-independent middleware for developing sensor applications is required, such as an Open Sensor Web Architecture

(OSWA) [Sang et al., 2010]. OSWA is built upon a uniform set of operations and standard data representations as defined in the Sensor Web Enablement Method (SWE) by the Open Geospatial Consortium (OGC).

- Secure Data aggregation (presentation category). An efficient and secure data aggregation method is required for extending the lifetime of the network as well as ensuring reliable data collected from sensors [Sang et al., 2010]. Node failures are a common characteristic of WSNs. Hence the network topology should have the capability to repair itself. Ensuring security is critical as the system is automatically linked to actuators and protecting the system from intruders is very important. Holohan et al. [Holohan and Schukat, 2010] propose to perform authentication using virtual certificate authorities for WSNs.

Sensor networks can also be further classified [Gluhak and Schott, 2007] according to the targets of the monitoring task: body sensor networks (BSN), object sensor networks (OSN) and environment sensor networks (ESN).

Addressing Schemes

The ability to uniquely identify "*things*" is critical to the success of IoT. This will not only allow us to uniquely identify billions of devices but also to control remote devices through the Internet. The most critical requirements are the uniqueness of the object identifier, reliability (robust responses in cases of technical failures for one or more of the nodes), persistence (long-term storage of the most critical information) and scalability (ability to extend to include a potentially high number of new devices). Every element that is already connected and those that are going to be connected must be identified by their unique identification, location and functionalities. The current IPv4 may support to an extent where a group of cohabiting sensor devices can be identified geographically, but not individually. The Internet Mobility attributes in the IPV6[10] may alleviate some of the device identification problems. However, the heterogeneous nature of wireless nodes, variable data types, concurrent operations and the confluence of data from devices exacerbates the problem further [Zorzi et al., 2010]. Persistent network functioning to channel the data traffic ubiquitously and relentlessly is another aspect of IoT. Although, the TCP/IP takes care of this mechanism by routing in a more reliable and efficient way,

[10] Internet Protocol version 6 (IPv6) is the latest version of the Internet Protocol (IP), the communications protocol that provides an identification and location system for computers on networks and routes traffic across the Internet. IPv6 uses a 128-bit address thus supporting the identification of a larger number of devices with respect to IPv4.

from source to destination. A gateway[11] (or proxy) is adopted to let sensor networks communicate with the World Wide Web or with other networks. Sensor network devices are resource constrained, the majority of the data processing and storage happens on the gateway, which then constitute a potential bottleneck. Furthermore, the scalability of the device address of the existing network must be sustainable. The addition of networks and devices must not hamper the performance of the network, the functioning of the devices, the reliability of the data over the network or the effective use of the devices from the user interface.

To address these issues, the *Uniform Resource Name* (URN) system is considered fundamental for the development of IoT. A URN creates replicas of the resources that can be accessed through the URL. With large amounts of spatial data being gathered, it is often quite important to take advantage of the benefits of metadata for transferring the information from a database to the user via the Internet [Honle et al., 2005]. IPv6 also gives a very good option to access the resources uniquely and remotely. Also critical to address is the deployment of a lightweight 6LoWPAN has been defined to let IPv6 be assigned to sensor network devices despite they run on a different protocol stack when compared to the Internet. At the node level each sensor will have a URN (as numbers) for sensors to be addressed by the gateway. The entire network now forms a web of connectivity from users (high-level) to sensors (low-level) that is addressable (through URN), accessible (through URL) and controllable (through URC).

2.4. Challenges and Platforms

The scientific challenges that must be overcome in order to realise the enormous potential of IoT are substantial and multidisciplinary [Akyildiz et al., 2002]. For instance, efficient heterogeneous sensing within a urban environment requires to simultaneously meet competing demands of multiple sensing modalities. This effects data storage, network traffic andenergy utilisation, in case of both continuous or random sampling within either fixed and mobile infrastructures [Harris and Shadbolt, 2005]. A generalised framework is required for data collection and modelling that effectively exploits spatial and temporal characteristics of the data, both in the sensing domain as well as the associated transform domains. For example, urban noise mapping needs an uninterrupted collection of noise levels using battery powered nodes using fixed infrastructure and participatory

[11] A network gateway is an inter-networking system capable of joining together two networks that use different base protocols.

sensing [Harris and Shadbolt, 2005] as a key component for health and quality of life services for its inhabitants.

Several challenges are due to the resource constraint nature of the devices involved in the IoT. For example, embedded tiny web servers are not as powerful as traditional web services. Intermittent connectivity caused by duty cycling may disrupt the availability of IoT-based web services. Heterogeneous networks are multi-service since they provide more than one distinct application or service at once. This implies not only multiple traffic types within the network, but also the ability of a single network to support all applications without QoS[12] compromise [Jirka et al., 2009a]. There are **two** application classes: 1. throughput and delay tolerant elastic traffic of (e.g. monitoring weather parameters at low sampling rates) 2. bandwidth and delay sensitive inelastic (real-time) traffic (e.g. noise or traffic monitoring) which can be further discriminated by data-related applications (e.g., high-vs-low resolution videos) with different QoS requirements. Therefore, a controlled, optimal approach to serve different network traffics, each with its own application QoS needs is required [Juels, 2004]. It is not easy to provide QoS guarantees in wireless networks, as segments often constitute gaps in resource guarantee due to resource allocation and management ability constraints in shared wireless media. Resource constraints in sensor networks create novel challenges for deep learning in terms of the need for adaptive, distributed and incremental learning techniques.

The low-power and lossy nature of WSNs is not properly supported and compensated by traditional Web protocols. When relying on conventional wireless sensing technology one or more nodes in the sensor network may function as gateways. The major advantage is that peer-to-peer communications among the nodes are possible with this kind of approach. However, this is significantly more expensive in large-scale applications and is limited by the battery life. Moreover most IP protocols cannot accommodate the sleep modes required by sensor motes in order to conserve battery life, thus further limiting their autonomy. Since the network connectivity is based on IP protocols, this would require the sensor devices to be constantly switched on. The energy requirements can be reduced in several different ways such as lower sampling or different transmission rates. However, such solutions can impact the timeliness and quality of the data available for the underlying applications. Tradeoffs are also possible during data trans- mission between timeliness and energy consumption (eg. real-time 3G vs. opportunistic WiFi). A variety of methods have

[12] Quality of service (QoS) refers to a network's ability to achieve maximum bandwidth and deal with other network performance elements like latency, error rate and uptime. Quality of service also involves controlling and managing network resources by setting priorities for specific types of data (e.g., video, audio, files) on the network.

been proposed in recent years, for calibrating these different tradeoffs, so that the energy efficiency is maximised with significantly compromising the data-centric needs of the application [Labs, 2012, Paek et al., 2010, Ra et al., 2010, Zhuang et al., 2010]. Examples of specific methods include energy-timeliness tradeoffs [Ra et al., 2010], adaptive sampling [Paek et al., 2010], and application-specific collection modes [Zhuang et al., 2010]. We note that the impact of such collection policies on data management and processing applications is likely be significant. Therefore, it is critical to design appropriate data cleaning and processing methods, which take such issues of data quality into consideration.

An energy efficient MAC protocol and appropriate routing protocols are critical for network efficiency. Several MAC protocols have been proposed for various domains with TDMA (collision free), CSMA (low traffic efficiency) and FDMA (collision free but requires additional circuitry in nodes) schemes available to the user [Juels et al., 2003]. However, they have not been yet standardised. Furthermore, networks must implement multi-path routing for self-adapting whenever sensors drop out, as it often happens. Multi-hop routing protocols are used in mobile ad hoc networks and terrestrial WSNs [Juels and Brainard, 2004]. Energy is the main consideration for the existing routing protocols.

The location of a sensor makes the sensor's observations meaningful. However, it is challenging to precisely determine the sensor's coordinates. The most advanced technologies so far detect latitude and longitude via GPS but with a degree of error. Recent solutions usually combine different location techniques to reduce the error range. For georeferencing features, clients should be able to provide not only the observed value but also spatial information. Given the amount of sensors and other data sources available in the IoT, the discovery of a sensor that meets specific user's requirements is a difficult task. Smart systems should support rich queries, which could include arithmetic, aggregation, and other database operators. Both people and things may need to discover the existence, functionality and information of their desired web services. For example, things require identities of smart things and web services within their environment in order to negotiate about shared goals to create a new mashup according to some requirement. Search engine is essential to WoT. Generally, as indicated in [Ostermaier et al., 2010], there are two fundamental approaches to construct a search engine for WoT. In the push approach, sensor outputs are proactively pushed to a search engine, which uses the data to resolve queries reactively. However, this method lacks of scalability in the smart things-based crowd- sourcing environment. It can be only applied to a system with limited number of devices. Alternatively, in the pull approach, only upon receiving a user

query, the search engine forwards it to the sensors to pull the relevant data. This method is scalable but challenged by the accuracy and timeliness. Here, we focus on the latter one. The increasing presence of Internet-Connected Objects and their pervasiveness into everyday life, make their discovery by search engine even more challenging. Furthermore, a key service for WoT will be the search engine that allows to search a physical-world service with certain properties. The traditional Web is dominated by static or slowly changing contents that are manually typed in by humans. The contents in WoT are rapidly changing because they are automatically produced by smart things. Thus, a search engine for WoT shall support searching rapidly changing content. This is a key challenge because existing search engines are based on the assumption that most web contents change slowly such that it is sufficient for the search engine to update an index at a low frequency. This is clearly impossible for the WoT where the states of many physical world devices changes are at frequency of minutes or even seconds. On the other hand, some Web content or service is significant only during a specified duration. In addition, future mashup shall be created dynamically on-demand according to the context. The source web services may need to be searched and obtained dynamically and in realtime. This issue becomes more challenging due to the dynamics of WoT, introduced by its features such as mobility and intermittent connectivity of smart things. The search engine for WoT will support real-time search of information and real-time discovery of web services. There has been some pioneer work on this issue. In [Ostermaier et al., 2010] the authors show how the existing web infrastructure can be used to support publishing of sensor and entity data. They implement a prototype of real-time search engine, called Dyser, which enables finding the real-world devices that exhibit a certain state at the time of the query. In [Roemer et al., 2010] the authors survey and clarify relevant existing approaches (e.g. Snoogle [Wang et al., 2010], Microsearch [Tan et al., 2008], MAX [Yap et al., 2005], etc.) according to query type, language, scope, accuracy and so on. Mayer et al. [Mayer and Guinard, 2011] present DiscoWoT, a semantic discovery service for Web-enabled smart things. DiscoWoT is based on the application with multiple discovery strategies to a representation of web resource, where arbitrary users can create and update the strategies at runtime using DiscoWoTs RESTful interface. The heterogeneity that characterise sensor devices and sensor networks affects all the OSI layers from network to application protocols. This leads an interoperability issue as already exposed, besides several challenges for integrating different data sources. A lack of integration would be very limiting for sensor discovery. During the information processing step, it is critical to consider and properly analyse the observed data according to their type and unit of measurement. Given the ubiquitous nature of sensors, the

observed data often carry sensitive information. For this reason sensor data sources are often closed. A tradeoff between the need for privacy and the anonymity for research purposes should be achieved, in order to exploit the potential of cross-dataset information. A significant number of IoT objects such as mobile phones can be connected by 3G and WiFi connectivity. However, the power usage of such systems is quite high. This family of devices usually belong to a participant who is responsible for maintaining the battery and other connectivity aspects of the sensing object which is transmitting the data. In such cases, however, the privacy of the transmitted data (eg. GPS location) becomes sensitive, and it is important to design privacy preservation paradigms in order to either limit the data transmission, or reduce the fidelity of the transmitted data. This is of course not desirable from the data analytics perspective, because it reduces the quality of the data analytics output. At the same time, this reduces the user-trust in the data analytics results. For security, the CoRE[13] working group has been exploring approaches to security bootstrapping that are realistic under the given constraints and requirements of the network. To ensure that any two nodes can join together, all nodes must implement at least one universal bootstrapping method. Security can be achieved using either session security or object security. Cipher suite will also be redesigned so as to be implemented with a minimal requirement. In [Kim, 2008] the authors present an analysis of security threats to the 6LoWPAN adaptation layer from the point of view of IP packet fragmentation attacks and proposes a protection mechanism against such attacks using time stamp and nonce options[14] that are added to the fragmentation packets at the 6LoWPAN adaptation layer. Allowing the information available on the Web poses a perceived privacy threat. The approach to use existing authentication service from third parties has been advocated. For example, Sensorpedia [Chen et al., 2007] relies on open data portability standards to ensure current and future interoperability with other web-based software applications. Some web service might be shared within restricted groups only. For example, home appliance web service may be accessible only to family members. Following the idea of levering existing social structure on Online Social Networks (OSNs) such as Twitter, Facebook, Linked-in and their APIs to define the access privilege of smart things, the authors in [Guinard et al., 2010] implement a prototype, called Social Access Controller(SAC), which is an authentication proxy between users and smart things. OSN-based methods can handle the access control between people and things but are unable to deal with the access control between things.

[13]The Constrained RESTful Environments (CoRE) working group aims at realizing the REpresentational State Transfer (REST) architecture in a suitable form for the most resource constrained devices.
[14]A nonce is a random number generated for one time use in a security operation.

Universal but distributed access control mechanism is expected to enable interoperation between things while preserving the privacy of the owners.

Sensor data is collected at a particularly high rate. A tradeoff is necessary among the need for historical records and the request for the latest information. Performant data storage systems are also required, to optimise the archiviation and retrieval of data streams. Historical recording also raises many privacy issues as the data collected can be used in both positive (e.g., for recommending services) and negative purposes(e.g., defamation). Digital forgetting could emerge as one of the key areas of research to address the concerns and the development of an appropriate framework to protect personal data [Gonzalez et al., 2006].

IoT systems should be reactive to the data they are sensing rather than perform a mere passive collection. An automatic reconfiguration of data collection and filtering is desirable, in synchronisation to the changes sensed in the surrounding environment. The ultimate goal of the IoT is to build an ecosystem that can provide user-oriented and environment-aware services. In other words, web services should be sensitive and responsive to the presence of people and the condition of environment. Ambient Intelligence(AmI) has been much addressed on stand-alone systems, such as wireless sensor and actuator networks. The sensor capable of recognising simple emergency situation may fire an alarm and the actuator can take an action accordingly. When it comes to AmI in IoT, new opportunities and challenges are exposed. Web services exposed by ubiquitous and pervasive devices will effect a larger community of users and such effect should be further investigated. One may easily find different public services on the Web and build private Web services using standard web-enabled devices in personal area network. The challenges first come from the heterogeneity and availability of smart things that provide web services. Unlike stand-alone systems where the devices are predetermined and configured according to the application requirement, some AmI applications in IoT may need to discover the required web services first. It is difficult to discover a web service via search engine, let alone basing the search on its QoS. Furthermore, this situation is exacerbated by the unexpected user requirements and environment (e.g. time, location, etc.). Recall that mashup technology is a key enabling technology of IoT. It can be expected that the mashups will be dominant in IoT. Another challenge of AmI in IoT is that the mashup shall intelligently adapt to user requirements and runtime environment. In other words, it shall be context-aware. In such a condition, dynamic mashup could be a good option. Other than developing static mashup by integrating existing web services together, rules about how to mashup services should be defined such that the basic web services are

dynamically added or deleted on-demand. The whole mashup process is transparent to the users and is automated. For example, to build a healthcare system for the elderly requires some private services to monitor and record their health conditions and some public services to know the environment information (e.g. temperature, humidity, light, traffic, etc.) about the places where they locate. The public services to be integrated shall be dynamically chosen according to their location. In an emergency condition, some services shall be automatically activated and integrated, e.g., the control service for automatic syringe might be activated and responded accurately according to the health condition and the environment parameters known from the other services. AmI of WoT is far more powerful and sophisticated than those examples. To fully explore the potential of smart things, more innovative solutions are expected to be proposed. Those solutions shall be able to orchestrate all available web services in a graceful manner and enable more intelligent user-oriented services. Some existing artificial intelligent concepts and technologies, such as Collective intelligence [Levy, 1999] and Semantic Web Services [Narayanan and McIlraith, 2002], may deserve revisiting in the hope of finding new efficient solutions feasible to smart things on the Web.

The primary networking issues for the internet of things arise during the data collection phase. At this phase, a variety of technologies are used for data collection, each of which have different tradeoffs in terms of capabilities, energy efficiency, and connectivity, and may also impact both the cleanliness of the data, and how it is transmitted and managed. Therefore, we will first discuss the key networking technologies used for data collection. This will further influence our discussion on data-centric issues of privacy, cleaning and management.

Sensor data are often redundant, incomplete or noisy because a large fraction of the readings are dropped, and there are cross-reads from multiple sensor readers. In these cases, a data cleaning step must be performed and a probabilistic uncertain modelling may be required [Deshpande et al., 2004]. This step is necessary also in case of privacy-preservation for an intentional reduction of data quality [Aggarwal, 2008]. Another reason for data to be noisy is its derivation from the conversion of one measured quantity into another one. This conversion may have different levels of precision. Changes in external conditions or ageing of sensors can introduce additional systematic errors, while periodic failure of sensors may lead to data incompleteness. An approach to reduce such errors consists of re-calibrating the sensor [Bychkovskiy et al., 2003] or performing data-driven cleaning and uncertainty modelling [Deshpande et al., 2004]. Uncertain probabilistic modelling has been the preferred solution across different con-

texts because of recent advances in the field of probabilistic databases [Aggarwal, 2008, Deshpande et al., 2004, Khoussainova et al., 2006, Aggarwal, 2009]. This solution relies on representing sensor data in a probabilistic format that reflects its errors and uncertainty. As a consequence, data mining using such sensor data achieves more effective results. However, all commercial solutions still use conventional (deterministic) representations of sensor data. Then, cleaning deterministic entities may require more direct solutions. In case of lost readings in RFID data, an alternative solution is to use a temporal smoothing filter [Gupta and Srivasatava, 2004, iAnywhere Inc., 2004]. Here, a sliding window over the RFID reader's data stream interpolates for lost readings from each tag within the time window. The idea is to provide each tag more opportunities to be read within the smoothing window. Since the window size is a critical parameter, a Statistical sMoothing for Unreliable RFid data (SMURF) has been proposed [Jeffrey et al., 2006c]. It consists of an adaptive smoothing filter for raw RFID data streams. This technique determines the most effective window size automatically, and continuously changes it over the course of the RFID stream. Several among such data cleaning methods use declarative queries over relational data streams to specify the cleaning stages [Jeffrey et al., 2006a, Jeffrey et al., 2006c, Jeffrey et al., 2006b].

Security is predicted to become a major concern for the IoT vision due to the sensitive and ubiquitous nature of the transmitted data. Also there can be many ways an IoT system could be attacked threatening data integrity and confidentiality [Hayes and Gutierrez, 2004, Jeffrey et al., 2006a]. Against outsider attackers, encryption ensures data confidentiality, whereas message authentication codes ensure data integrity and authenticity [Jeffrey et al., 2006c]. However, encryption does not protect against insider malicious attacks. One of the WSNs maintenance tasks is particularly vulnerable from a security perspective: the periodic installation or update of sensor applications. This is usually performed by remote wireless reprogramming which consists only of a data dissemination protocol that - without authentication - distributes code to all nodes in the network. Such method constitute an obvious security threat allowing for the installation of malicious software.

People-centric platforms provide low-cost information about the environment localised to the user and how the user experiences it [Harris and Shadbolt, 2005, Aggarwal, 2008, Kagal et al., 2003]. This forms a social currency and results in the provision of more timely data if compared with a fixed infrastructure sensor network [Kagal et al., 2004]. Here users can express their feedback along with relevant contextual information and the most apprpriate devices become mobile phones. However, this type of platforms rely on

user's desired participation, volunteering data, thus leading to the problem of missing samples and inconsistency. Consequently, the ability to produce meaningful data for any applications and policy decisions is limited.

Given the pervasiveness and broadness nature of the IoT, extracting useful information at different spatial and temporal resolutions is not straightforward. A proposed approach uses shallow learning methods where pre-defined events and data anomalies are extracted using supervised and unsupervised learning [Khoussainova et al., 2006]. The next level of learning involves inferring local activities by using temporal information of events extracted from shallow learning. Deep learning is applied to learn multiple layers of abstraction on complex events to interpret the given data [Khoussainova et al., 2006].

As new display technologies emerge, creative visualisation will be enabled. The evolution from CRT to Plasma, LCD, LED, and AMOLED displays has given rise to highly efficient data representation (using touch interface) with the user being able to navigate the data better than ever before. With emerging 3D displays, this area is certain to have more research and development opportunities.

Heterogeneous spatio-temporal data like IoT sensor data requires new visualisation schemes [Kim et al., 2005]. Their geo-relatedness and sparse nature lead to the necessity for a framework based on Internet GIS. An approach is to represent them in a 3D landscape that varies temporally [Kinoshita et al., 2004].

2.5. Context Awareness

Rogers proposes a human centric perspective over the IoT in which both human capabilities and the environment are exploited via human creativity [Rogers, 2006]. The interpretation of the sensor data we collect and the consequent decision making process, are highly dependent on context information. For instance, low values of temperature may be desirable in a fridge but not in a private household, consequently leading to trigger actions for either decreasing or increasing such values. Context-aware computing allows the storage of context information related to sensor data. It has already proven to be successful in other paradigms such as ubiquitous computing and Sensor Web, prior to the IoT.

Definition . The term *context-awareness*, also called *sentient*, was first introduced by Schilit and Theimer [Schilit and Theimer, 1994] in 1994. Later, it was defined by Ryan et al. focusing on computer applications and systems [Ryan et al., 1997].

Abowd and Mynatt define context by the minimum information necessary to describe it such as the content summarised by the five Ws: Who, What, Where, When, Why [Abowd and Mynatt, 2000]. Ahn and Kim consider context as a set of events with logical and timing relations among them [Ahn and Kim, 2006]. An event is an occurrence that triggers a condition in a target area and can be classified as either discrete or continuous. Given the sampling rate p two events are discrete if given their occurrence at time t and $t + p$ respectively, they are considered to be two separate event instances. Examples are a door opening or lights switching. Two events lasting for at least time p and occurring at time t and $t + p$ respectively, are considered continuous if they can not be considered as two separate events. For instance, raining or driving a car.

Dey et al. discussed and detailed the weaknesses of the numerous definitions given by researchers to the term *context* [Dey et al., 2001]. They claimed that the definitions provided by Schilit et al. and others can not be used to identify new context because they are not generalised but application domain-specific [Schilit and Theimer, 1994, Brown, 1996, Franklin and Flachsbart, 1998, Rodden et al., 1998, Hull et al., 1997, Ward et al., 1997, Schilit et al., 1994, Pascoe, 1998].

As an alternative, Dey et al. define context as "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" [Abowd et al., 1999]. They also define a context-aware system as a system that "uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task". We choose the latter definition [Abowd et al., 1999] as the most appropriate in this thesis since it allows to easily determine whether a data value can be classified as context information or not.

Sanchez et al. distinguish between raw data and context information [Sanchez et al., 2006]. Any data that is unprocessed and collected directly from the data source is considered raw sensor data. Processed data, eventually also checked for consistency and enriched with meta data, is considered context information. For example, GPS sensor readings would be considered raw sensor data while the information about the geographical location that they represent and which is derived by processing them, is considered context data.

Evaluating the quality of the context information (QoC) has been thoroughfully investigated [Bellavista et al., 2013]. QoC is defined based on three parameters: validity, freshness and precision [Bellavista et al., 2013]. These parameters can be used in context data conflicts resolution. They also depend on the quality of the physical sensor, of the context data and of the delivery process. Quality, validity, accuracy, cost and effort of acquisition, etc. vary significantly according to the techniques used. Given the large amount of data sources that can be used to retrieve the same data value in the IoT vision, deciding which source to use is an even more challenging task. This is problem raised by the Research Question **Q 3. Relevancy**: *How to identify which sensors are more relevant sources of information to define a specific small context scope - the Little Data - of interest?*. We address this question in Chapter 5.

We can identify [Abowd et al., 1999] three features that a context-aware application can support: **presentation**, **execution**, and **tagging**. From an IoT perspective, context can be used to filter the content to be presented to the user (**presentation**). For instance, a context-aware application could automatically generate a shopping list based on the items sensed as available or missing in the fridge and other appliances by deployed sensors [Institutes, 2011, Moses, 2012]. The presentation of this content and other applications should be automatically executed (**execution**)based on the context. Machine-to-machine communication is a significant part of the IoT. Sensor data from a single sensor does not provide enough information to fully derive the context. Therefore they often need to be combined together according to the context. Context has to be associated with the sensor data for processing and interpretation (**tagging**).

Context data can be classified according to several criteria. Abowd et al. distinguish between primary and secondary context types [Abowd et al., 1999]. Location, identity, time and activity are considered primary. Secondary context data is the data that can be derived from the primary context. For example, given primary context such as a person's identity, we can derive related information such as phone numbers, addresses, email addresses, etc. However, this definition does not include cases in which a data value is generated by combining sensor readings from two distinct sensors. Also, the same data value can be considered as primary context in one scenario and secondary context in another. For instance location can be derived from GPS sensors directly or from a user's profile indirectly. A type of context information can be classified as both primary and secondary type, too. For example, a location can be represented by raw GPS reading a or by the name of the location itself.

Operational categorisation schemes model quality and cost factors related to context, allowing to better identify issues and challenges in data acquisition techniques. Conceptual categorisation schemes model the conceptual relationships between context data. A precise context model should integrate these two different schemes. Several other schemes have also been introduced, focusing on different perspectives. Schilit et al. classified context into three categories using a conceptual categorisation technique based on three questions: Where, Who, What with regards to the nearby resources [Schilit et al., 1994]. Henricksen defined **four** categories based on the operational categorisation technique used: **sensed** (data directly collected from the sensors); **static**; **profiled** (information that changes over time with a low frequency); **derived** (from primary context)four. Van Bunningen et al. classified the context categorisation schemes into operational and conceptual [van Bunningen et al., 2005]. The operational categorisation is based on how the data was acquired, modelled, and treated. The conceptual categorisation is based on the meaning and conceptual relationships between the context. To build an ideal context-aware middleware solution for the IoT, different categorisation schemes need to be combined together in order to complement their strengths and mitigate their weaknesses.

Platforms . We distinguish between systems, middlewares and toolkits. Systems are designed to provide only a few tasks for the end user. Toolkits provide very specific functionalities for other systems, applications and middleware developers. Middleware is a software layer that lies between the hardware and application layers and provides reusable functionalities [Issarny et al., 2007]. They provide an abstraction to address common application development issues such as heterogeneity, interoperability, security, and dependability. Toolkits in general are suitable for limited scale application. Managing context in the IoT paradigm requires middleware solutions that can provide more functionality towards managing data. Applications should be able to be built on top of the middleware so they can request context from the middleware. Context Toolkit has introduced the notion of having common standard interfaces that are chosen according to the context [Dey et al., 2001]. The context widget component encapsulates the communication between context sources and the toolkit. Further, Intelligibility Toolkit provides explanations to the users to improve the trust between users and the context- aware applications which helps in faster adaptation of the users towards IoT [Lim and Dey, 2010].

In order to identify context, it is possible to combine data from different data sources. There is a significant gap between low-level sensor readings and high-level situation-awareness [Castelli et al., 2009]. Collecting low-level sensor data is becoming significantly easier and cheaper due to advances in sensing technology. As a result, enormous amounts (big data) of sensor data is available [Zaslavsky et al., 2012]. Since it is impossible to plan all the possible device interactions at the development stage, a programming model that allows dynamic composition in the IoT is desirable [Chen et al., 2008]. Software solutions should be able to dynamically orchestrate components such as reasoning models, data fusion operators, knowledge bases and context discovery modules, according to the requirements. Interactions among IoT devices are expected to be processed in real time unlike usual context-aware computing applications. Event detection, context reasoning and query processing are critical real time processing tasks. Solutions in this direction allow sensor stream data processing [Kwon et al., 2010]. In the following, we describe the major context-aware applications from the state of the art: 1. **Aura** is a task oriented system based on distributed architecture for different common devices [Garlan et al., 2002]. The objective is to run a set of applications called "personal aura" on all devices in order to manage user tasks in a context-aware fashion across all the devices smoothly. 2. **CARISMA** (Context-Aware Reflective middleware System for Mobile Applications) is a middleware that represents context by XML-based application profiles, allowing each application to maintain metadata [Capra et al., 2003]. Such metadata are classified as passive or active according to whether they define rule-triggered actions or not. It implements a conflict resolution mechanism based on macroeconomic techniques, where final decisions are made in order to maximise the social welfare among the agents. 3. **Gaia** is a distributed infrastructure that performs uncertainty based reasoning [Anand and Roy, 2003] to depict context represented using ontologies. 4. **e-SENSE** enables ambient intelligence using wireless multi-sensor networks for making context-rich information available to applications and services [Gluhak and Schott, 2007]. e-SENSE combines body sensor networks (BSN), object sensor networks (OSN), and environment sensor networks (ESN) to capture context in the IoT paradigm. 5. **HCoM** (Hybrid Context Management) is a hybrid approach which combines semantic ontology and relational schemas [Ejigu et al., 2007]. Standard database management systems are not considered able to manage context on one side. Ontologies may not perform well in terms of efficiency and query processing with large volumes of data. As a result of such considerations, HCoM presents an hybrid approach. 6. **EMoCASN** (Environment Monitoring Oriented Context Aware Sensor Networks) is a Context-Aware model for Sensor Networks (CASN) [Li et al., 2008]. It focuses on low-level context data. For example, the

remaining energy of a node is low-level context information that can be used to determine an energy efficient routing. 7. **Hydra3** is an IoT middleware that aims to integrate wireless devices and sensors into ambient intelligence systems [Badii et al., 2010]. It provides the capabilities of both high-level, powerful reasoning - based on the use of ontologies - and lower-level semantic processing - based on object-oriented/key-value approach -. 8. **MidSen** is context-aware middleware for WSN [Patel et al., 2009]. The system is based on Event-Condition-Action (ECA) rules. It highlights the importance of efficient event detection by processing two algorithms: event detection algorithm (EDA) and context-aware service discovery algorithm (CASDA). 9. **Octopus** is an open-source, dynamically extensible system that supports data management and fusion for IoT applications [Firner et al., 2011]. Octopus develops middleware abstractions and programming models for the IoT. It enables non-specialised developers to deploy sensors and applications without detailed knowledge of the underlying technologies and network. 10. **CoOL** allows to extend any general purpose service model with context management functionality, e.g., context modelling and reasoning [Strang et al., 2003].

Making correct design decisions is a critical task in IoT. For example, data modelling and communication can be done using different techniques as follows where each method has its own advantages and disadvantages [Chen et al., 2008]. Binary is smaller in size than the other three formats and also portable due to its small size. In contrast, it is difficult to extend or modify data in binary format. Objects methods allow complex data structures. Attribute-value pairs methods support more limited complexity than an object representation. However, simpler representations enable applications that are independent from language and platform. XML methods provide more opportunities for complex data structures. However, XML brings substantial overhead in term of network communications and processing.

Due to the unpredictability and broadness of the IoT, data models need to be extensible on demand. For example, IoT solutions may need to expand their knowledge-bases towards different domains. SOCAM [Gu et al., 2005] shows how knowledge can be separated into different levels of ontologies, i.e., upper ontology and domain-specific ontology. In SOCAM, the upper ontology models general purpose data while the domain specific ontologies model domain specific data, which is allowed to extend to both levels independently. As an IoT solution will be used in many different domains, the ability to add ontologies (i.e. knowledge) when necessary is critical for wider adaptation. SCK [de Freitas et al., 2005], Zhan et al. [Zhan et al., 2009] and BIONETS [Jacob et al., 2006] use different ontologies for each context category. There are many different types of context categories which

model context from different perspectives. Therefore, it is important to store different types of context for different situations. They also stress the requirement of having domain specific and domain independent ontologies. Gaia [Roman et al., 2002], Ko and Sim [Ko and Sim, 2008], CDMS [Xue et al., 2008] and HCoM [Ejigu et al., 2007] highlight the importance of employing multiple reasoning techniques such as Bayesian networks, probabilistic and fuzzy logic according to the different situations. Incorporation of multiple modelling and reasoning techniques can mitigate individual weaknesses using each other's strengths. COSAR combines statistical reasoning and ontological reasoning techniques to achieve more accurate results [Riboni and Bettini, 2009].

Different types of context providers, which are dedicated to communicating and retrieving data related to a specific domain, can be employed when necessary. In line with above solutions, COPAL [Li et al., 2010] demonstrates the essential features that an IoT middleware should have, such as loosely coupled plugin architecture and automated code generation via abstracts which stimulates extendibility and usability. This is a one of the most commons tasks need to be performed by IoT solutions.

2.6. Conclusion

The creation of the Internet has marked a foremost milestone towards achieving the Ubiquitous Computing's vision which enables individual devices to communicate with any other device in the world. The inter-networking reveals the potential of a seemingly endless amount of distributed computing resources and storage owned by various owners. Caceres and Friday [Caceres and Friday, 2012] discuss the progress, opportunities and challenges during the 20 years anniversary of Ubiquitous Computing. They discuss the building blocks of Ubiquitous Computing and the characteristics of the system to adapt to the changing world. More importantly, they identify two critical technologies for growing the Ubiquitous Computing infrastructure: Cloud Computing and the Internet of Things.

The IoT has gained significant attention over the last few years. With the advances in sensor hardware technology and cheap materials, sensors are expected to be attached to all the objects around us, so these can communicate with each other with minimum human intervention. Understanding sensor data is one of the main challenges that the IoT faces. IoT does not merely concern the connectivity of smart things, but more about the interaction or interoperation between things and between things and people.

This requires that all the smart things can speak the same language to communicate freely with each other. It has been considered as a good solution to extend existing web architecture to this new domain by incorporating smart things into the Web. This vision has been supported and heavily invested by governments, interest groups, companies, and research institutes. For example, context awareness has been identified as an important IoT research need by the Cluster of European Research Projects on the IoT (CERP-IoT) [Guillemin and Friess, 2009] funded by the European Union. The EU has allocated a time frame for research and development into context-aware computing focused on the IoT to be carried out during 2015-2020. In this chapter, we analysed and evaluated context-aware computing research efforts to understand how the challenges in the field of context-aware computing have been tackled in desktop, web, mobile, sensor networks, and pervasive computing paradigms. A large number of solutions exist in terms of systems, middleware, applications, techniques, and models proposed by researchers to solve different challenges in context-aware computing. We also discussed some of the trends in the field. The results clearly show the importance of context awareness in the IoT paradigm. Our ultimate goal is to build a foundation that helps us to understand what has happened in the past so we can plan for the future more efficiently and effectively.

Chapter 3.

Semantic Web of Things

The key to the power of the Internet of Things paradigm is the ability to provide real time data from many different distributed sources to other machines, smart entities and people for a variety of services. One major challenge is that the underlying data from different resources is extremely heterogeneous, can be very noisy and is usually very large scale and distributed. Furthermore, it is hard for other entities to use the data effectively, without a clear description of what is available for processing. Unlike the World Wide Web of documents, in which the objects themselves are described in terms of a natural lexicon, the IoT objects and data are heterogeneous, and may not be naturally available in a sufficiently descriptive way to be searchable, unless an effort is made to create standardised descriptions of these objects in terms of their properties.

In order to enable effective use of this very heterogeneous and distributed data, frameworks are required to describe the data in a sufficiently intuitive way, so that it becomes more easily usable. In other terms it is necessary to address the problem of *semantic interoperability*. This leads to unprecedented challenges both in terms of providing high quality, scalable and real time analytics, and also in terms of intuitively describing to users information about what kind of data and services are available in a variety of scenarios. Therefore, methods are required to clean, manage, query and analyse the data in the distributed way. The cleaning is usually performed at data collection time, and is often embedded in a middleware that interfaces with sensor devices. Therefore, the research on data cleaning is often studied in the context of the *things-oriented* vision mentioned in Chapter 2, Section 2.3. The issues of providing standardised descriptions and access to the data for smart services are generally studied in the context of standardised Web protocols and interfaces, and description/querying frameworks such as offered by semantic web technology. The idea is to reuse the existing

Web infrastructure in an intuitive way, so the heterogeneity and distributed nature of the different data sources can be seamlessly integrated with the different services.

So far, the approach to address interoperability was based on standardisation. This has led the creation of the Universal Plug and Play (UPnP) set of networking protocols, published as an international standard, ISO/IEC 29341, in 2008. UPnP-compatible devices can seamlessly discover each other, dynamically join a network and advertise their capabilities upon request. However, the UPnP architecture does not allow authentication, thus leading to major security flaws. A Semantic Web-based architecture would be more flexible, extensible and abel to rely on a broad range of already existing technology solutions.

These interoperability issues are usually studied in the context of the Web of Things and Sensor Web visions described in Chapter 2 and the Semantic Web vision which we detail in the following. Standardisation can remove some of the difficulties of device incompatibility, and there are a number of standards for sensor networks [Botts et al., 2006]. However, standardisation is typically more successful in removing interface heterogeneity than solving data and concept incompatibilities. The Open Geospatial Consortium's (OGC) Sensor Web Enablement (SWE) suite of standards [Botts et al., 2006], for example, standardise interfaces for services and description languages for sensors and their processes. Quite deliberately, the OGC's SWE working groups have not attempted to provide standards for interoperability beyond describing a standard set of functions or a standard syntax: domain semantics, for example, have been left to the relevant communities. The OGC's choice is prudent for, and a key feature of, a suite of domain independent standards. It does, however, mean that without external agreement, SWE cannot provide more than syntactic interoperability. Using vocabularies of concepts, relationships between those concepts and various reasoning techniques, semantics can, with largely domain independent techniques, provide more than syntactic interoperability.

Standardisation is important in the IoT paradigm, because it increases interoperability and extendibility. Standard interfaces and structures would guarantee a smooth interaction between new and old components. The semantic approach to information systems design uses declarative descriptions of information and processing units, allowing (semi-)automatic satisfaction of declaratively described requirements. Declarative descriptions enable both domain-independent and domain-specific reasoning of various forms (logic-based or otherwise) to be applied in processes such as entity identification, search, and query and workflow generation. Metadata is used to annotate a spectrum of data and service functions with the purpose of documenting or explicitly and implicitly

linking them. Frameworks like the *Resource Description Framework* (RDF) support such a standardised descriptive approach, which greatly eases various functions such as search and querying in the context of the underlying heterogeneity and lack of naturally available descriptions of the objects and the data. Semantic technologies are viewed as a key to resolving the problems of interoperability and integration within this heterogeneous world of ubiquitously interconnected objects and systems [Katasonov et al., 2008]. In the following sections we will describe the semantic technologies which constitute the foundation of modern Semantic Sensor Web solutions.

3.1. Semantic Web Vision

The Semantic Web vision [Lee et al., 2001] was to tense the World Wide Web more intelligent by layering the networked Web content with semantics. The idea was that a semantic layer would enable the realization of automated agents and applications that understand or comprehend Web content for specific tasks and applications. Similarly the Semantic Sensor Web puts the layer of intelligence and semantics on top of the deluge of data coming from sensors. In simple terms, it is the Semantic Sensor Web that allows automated applications to understand, interpret and reason with basic but critical semantic notions. For instance, the system could automatically recognise geo-spatial and spatio-temporal characteristics of sensor data and appropriately reason over them, e.g., *nearby/far*, *soon/immediately* . In summary, it enables true semantic interoperability and integration over sensor data.

3.2. Technologies

The World Wide Web [Berners-Lee et al., 1994] allowed people to publish information easily. It provided read-only access to an immense quantity of information, which grew exponentially, as the technology evolved. Internet connections have become faster, cheaper and more reliable while the connection coverage has extended worldwide. With the Semantic Web we move on from a Web of Documents understood by humans to a Web of machine understandable information [Berners-Lee et al., 2001]. But this was not something new: most of the ideas we now attribute to the Semantic Web — typed links and nodes — were in fact present in the initial proposal of the Mesh [Lemahieu, 1999], but they were omitted in favour of simplicity.

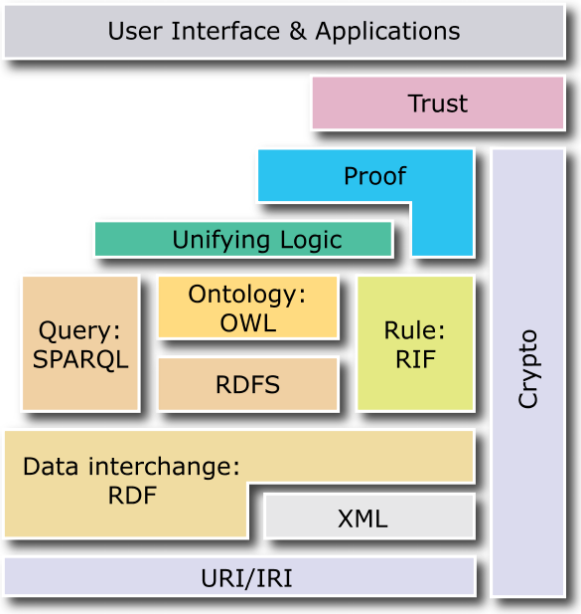


Figure 3.1.: The Semantic Web layer cake.

Thus, the goal of the Semantic Web is to add (machine understandable) meaning to the huge repository of connected information that is the Web. To accomplish this, the Semantic Web uses the same infrastructure and standards as the Web, along with additional technologies, which make up the "Semantic Web layer cake"[1] shown in Figure 3.1. In this section we will discuss some of the technologies that make up the layers, and which are relevant for the following chapters.

3.2.1. Resource Description Framework (RDF)

"The Resource Description Framework (RDF) is a framework for representing information in the Web." [Klyne and Carroll, 2004] It defines a standard model for representing and exchanging information on the Semantic Web. The RDF Specification has been a W3C Recommendation since 1999 [Lassila and Swick, 1999], with the latest version being a suite of six W3C Recommendations[2], published in 2004.

[1] http://www.w3.org/2007/03/layerCake.svg
[2] http://www.w3.org/standards/techs/rdf

```
@prefix ex: <http://www.example.org/RDFexample#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix bibo: <http://purl.org/ontology/bibo/> .
@prefix dcterms: <http://purl.org/dc/terms/> .

ex:doctorow a foaf:Person ;
   foaf:name "Cory Doctorow" ;
   foaf:firstName "Cory" ;
   foaf:surname "Doctorow" ;
   foaf:homepage <http://craphound.org> ;
   foaf:made ex:ftw .
ex:ftw a bibo:Book ;
   dcterms:title "For The Win" ;
   foaf:maker ex:doctorow ;
   bibo:issn13 "9780765322166" .
```

Listing 3.1: RDF example represented in Turtle.

RDF is designed to allow flexible representation of information. The underlying structure of any data represented with RDF is a graph, which is a collection of triples. Each triple represents an RDF statement, and is made up of a subject, a predicate (or property), and an object. An example is shown in Listing 3.1. A triple can be seen as a graph made of two nodes, the subject and the object, connected through an arc, represented by the predicate. Thus, a set of triples can be represented as a graph by a corresponding set of nodes and arcs (see Figure 3.2 for an example). Any information about a resource can be expressed through triples, including information about a triple, through a process called reification. A powerful feature, reification adds complexity, and is usually used sparsely. An extension to RDF allows statements to be grouped in *named graphs* which helps avoid the use of reification to store meta-information about triples, like provenance for example.

The RDF specification defines three types of elements:

- identified resources, which are represented by a URI. They can appear on any position in a triple.
- unidentified resources, called blank nodes, which cannot appear as predicates in triples. They are unnamed nodes within a graph, and using them can pose problems, thus it is discouraged [Bizer et al., 2007].

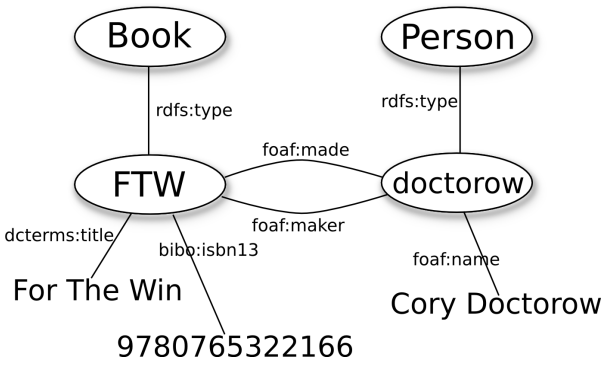


Figure 3.2.: RDF example represented as a graph.

- literals, which can only appear as objects in a triple. Literals represent values of properties. They can be typed, using XML Schema datatypes, or can have a language tag.

RDF is a data model which can be serialised in several ways. The most popular serialisation of RDF was RDF/XML [Beckett, 2004], due to the popularity and familiarity of XML. More recently however, other serialisations, more suitable for human consumption, have become preferred: N-Triples [Grant and Beckett, 2004], Turtle [Beckett, 2007], and N3 [Berners-Lee, 2006b]. RDFa [Adida et al., 2008] is another syntax for RDF, which allows embedding of RDF statements in HTML pages. In the following chapters of the thesis, we will use the Turtle notation in listings.

SPARQL [Prud'hommeaux and Seaborne, 2008] is the recommended query language for RDF. Other query languages for RDF exist, including:

- RDF Query Language (RQL) [Karvounarakis et al., 2002],
- Sesame RDF Query Language (SeRQL) [Broekstra and Kampman, 2003],
- RDF Data Query Language (RDQL) [Seaborne, 2004].

3.2.2. Ontologies

In this Section we give a concise introduction to semantics and vocabularies required for understanding the foundations of the approach we undertake for the core contribution of this thesis. An ontology is a formal specification of a shared conceptualization. Every knowledge base (or corresponding agent) is committed to some conceptualization. We can describe the ontology of a program by defining a set of representational

terms[Obitko, 2007]. In such an ontology, the names of entities in the universe of discourse (i.e., set of objects that can be represented, e.g., classes, relations, functions) are associated with both descriptions of what the names mean and formal axioms. In particular, such axioms constrain the interpretation and well-formed use of the corresponding terms. Agents commit to ontologies and ontologies are designed so that the knowledge can be shared among agents.

Ontologies are expressed using a formal representation in order to be machine processable. There exist several formal languages for this purpose, each characterized by different levels of expressivity. A specification is considered formal when at least one relation is defined between terms in a formal language, so that new conclusions can be inferred. Usually this relation is the "is-a" one. In fact, the backbone of an ontology is often a taxonomy, i.e., a hierarchical classification, e.g., of such living organisms, expressing subsumption. For instance, A subsumes B meaning that everything that is in A is also in B. More expressive formal languages provide a set of constructs to describe classes, instances, relations and constraints. The most formal and expressive ones are those that use full logics. During the ontology development is usually better to choose an expressive language. Afterwards, in case the performance is not acceptable, the ontology can be reduced to a subset for some levels of automatic processing.

Ontologies can be modelled by using several representations such as Concept Networks, Conceptual Graphs and either Common or Description Logics. We used Conceptual Graphs (which uses Concept Networks), detailed below. The rationale behind this choice relies in the expressivity that this representation allows us to keep, since these graphs are directly translatable to first-order logic [Scherp et al., 2011]. In a nutshell, the formalism we used, is as follows: 1. A Concept Network: Graph where vertices represent concepts and where edges represent relations between concepts, as depicted in Figure 3.2 2. A Conceptual Graph: Bipartite, directed graph based on a Concept Network but directly translatable into first-order predicate logic. For instance, we could consider a subgraph of the graph in Figure 3.2 as a conceptual graph representing the Display Form (DF) of the sentence "FTW is titled *For the Win*". This can be directly translated into the textual notation Linear Form (LF) as $[FTW] - (Titled) - [ForTheWin]$.

Knowledge Expression formalisms: The Semantic Web [Lee et al., 2001] is an effort supported and started by the W3C, to make all information available on the Web, "understandable" and processable by machines. Thus humans would be able to easily find required knowledge rather than just web documents in which the knowledge is hidden and

Figure 3.3.: Concept Network where nodes represent concepts and arches represent the relationship between them.

sparse. Like the Web is a distributed hypertext system, the Semantic Web is a distributed knowledge base system. Consequently ontologies are needed to unambiguously define meaning and relations of such distributed heterogeneous data items, and they must be represented in a proper machine-understandable formalism. The W3C recommends RDF, the Resource Description Framework, for this purpose.

We used the Web Ontology Language (OWL2)[3] because we want to support reasoning. In fact, it adds constructs to the RDFS ones that add in expressivity and it is syntactically embedded into RDF. It defines three sublanguages (i.e., syntactic restrictions). Each of the profiles trades off different aspects of OWL's expressive power in return for different computational and/or implementational benefits (e.g., reasoning complexity in range of LOGSPACE to PTIME):

1. **OWL2 EL** enables polynomial time algorithms for all the standard reasoning tasks
2. **OWL2 QL** enables conjunctive queries to be answered in LogSpace using standard relational database technology
3. **OWL2 RL** enables the implementation of polynomial time reasoning algorithms using rule-extended database technologies operating directly on RDF triples.

Reasoning and Querying. RDFS and OWL2 have semantics defined that can be used for reasoning by means of the RIF (W3C Rule Interchange Format, to express rules and have the computer executing them) and SWRL (Semantic Web Rule Language, with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language). In order to both query and properly access RDF data, the Simple Protocol and RDF Query Language (SPARQL) has been defined as both an SQL-like query language and a protocol for accessing RDF data. It is a W3C recommendation [W3C RDF Data Access Working Group members, 2010] . RDF triples and resources

[3] http://www.w3.org/TR/owl2-overview

are used for both matching part of the query and returning results. Since both RDFS and OWL2 are built on RDF, SPARQL can be used for querying and accessing ontologies and knowledge bases directly.

Operations on Ontologies: An application might use multiple ontologies, especially when a modular design of ontologies is adopted to integrate with systems that use other ontologies (which is the case, regularly). In this case, some operations on ontologies are needed in order to work with all of them. The terminology in this area is still not stable and different authors may use these terms slightly differently. However, each of these operations is important for ontology maintenance and integration. In particular, we applied Alignment and Inheritance to the SPITFIRE ontology, i.e., respectively, mapping of ontologies in both directions to modify original ontologies and to get a correct translation, and inheriting from an already existing sensor ontology, all concepts, relations and restrictions or axioms without introducing any inconsistency via additional knowledge. Such operations are not always suitable for any kind of ontology. In general, they are very difficult tasks not solvable automatically e.g., because of not decidability due to very expressive logical languages or because of insufficient specification of an ontology that is not enough to find similarities with another one. Because of these reasons these tasks are usually performed manually or semi-automatically, where a machine helps to find possible relations between elements from different ontologies, but a final confirmation of the relation is left to human experts. Experts decide accordingly to either the description of the ontology elements or just their names and common sense.

Modularisation: The purpose of authoring ontologies is also knowledge reuse. Once an ontology is created for a domain, it should be, at least to some degree, reusable for other applications in the same domain. To simplify both ontology development and reuse, modular design is critical. In a modular design, ontology inheritance is applied to achieve the scenario illustrated in Figure 3.4, in which upper ontologies describe general knowledge while application ontologies describe knowledge for a particular application.

An ontology can be classified, depending on its scope, as follows :

1. Upper, generic, top-level ontology: Describes general knowledge, e.g., time and space.

2. Domain ontology: Describes a domain, e.g., the medical domain or the electrical engineering domain, or narrower domains, such as the personal computer domain.

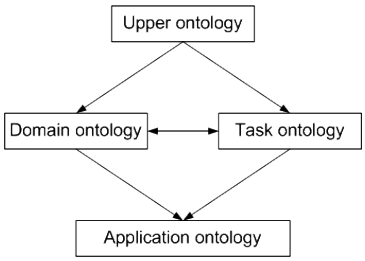


Figure 3.4.: Modular design of ontologies [Obitko, 2007].

3. Task ontology: Suitable for a specific task, e.g., assembling components.

4. Application ontology: Developed for a specific application, e.g., assembling personal computers.

At each of the above levels, modularization can be applied recursively as well. For instance, an upper ontology may consist of modules for real numbers, topology, time, and space. Such modules of an upper ontology are usually called generic ontologies. At lower level, ontologies import from upper level ones and add additional specific knowledge, creating a lattice of ontologies that is defined by partial ordering of inheritance. Task and domain ontologies may be independent and are merged for application ontologies, or it is also possible that a task ontology imports a domain ontology (see Figure 3.4).

When developing new ontologies, it is desirable to reuse existing ones as much as possible, and to import upper level ones when suitable. This will simplify the development since one can focus at the domain or application specific knowledge only (a certain degreed) and it will ensure interoperability with existing users of the re-used ontologies. It will also simplify integration between applications in the future since shareable sub-parts can be defined.

3.2.3. Linked Data

The term Linked Data refers to a set of best practices for publishing and interlinking structured data on the Web. These best practices were introduced by Tim Berners-Lee in his Web architecture note *Linked Data* and have become known as the Linked Data principles [Berners-Lee, 2006a]. These principles are the following:

- use URIs[4] as names for things;

[4] A Uniform Resource Identifier (URI) is a string of characters used to identify the name of a resource.

- use HTTP URIs, so that people can look up those names;
- when someone looks up a URI, provide useful information, using the standards (RDF, SPARQL);
- include links to other URIs, so that they can discover more things;
- The basic idea of Linked Data is to apply the general architecture of the World Wide Web to the task of sharing structured data on global scale.

A significant number of individuals and organizations have adopted Linked Data as a way to publish their data, resulting in the Web of Data [Bizer, 2009], i.e., a global data space [Bizer et al., 2010] consisting of billions of RDF statements from numerous sources covering all sorts of topics.

Ontologies have been extensively used in data integration as[Cruz and Xiao, 2003]:

- metadata representation;
- global conceptualization;
- support for high-level queries;
- declarative mediation;
- mapping support.

We used them as both a metadata representation and a global conceptualisation. Data publishers may facilitate the integration of resources across different data sources on the Web of Data by 1. reusing terms from widely used vocabularies 2. publishing mappings between terms from different vocabularies 3. creating RDF links between related resources.Therefore, data providers and data consumers share a common responsibility in the data integration effort. The downside of this open approach is that the quality of the links provided is uncertain [Bizer et al., 2009].

3.2.4. Semantic Web Data Access

Data access concerning Linked Data in itself can pose some challenges. Not only can the original datasource and its type vary (relational databases, sensors, etc.) but also questions such as response times and availability of data sources have to be addressed.

In the following we will discuss tools and frameworks typically used in the back-end and challenges that come along with their usage.

Relational databases - RDB2RDF

The majority of dynamic Web content is backed by Relational DataBases (RDB), and so are many enterprise systems. In order to make this huge amount of relational data available for the Web of Data, a connection must be established between RDBs and a format suitable for the Web of Data. RDB2RDF is a proposed standard language for mapping RDB schemas into RDF and OWL [Malhorta et al., 2012]. In the following, we will use RDB2RDF to denote any technique that takes as an input a RDB (schema and data) and produces one or more RDF graphs, as depicted in Fig. 3.5.

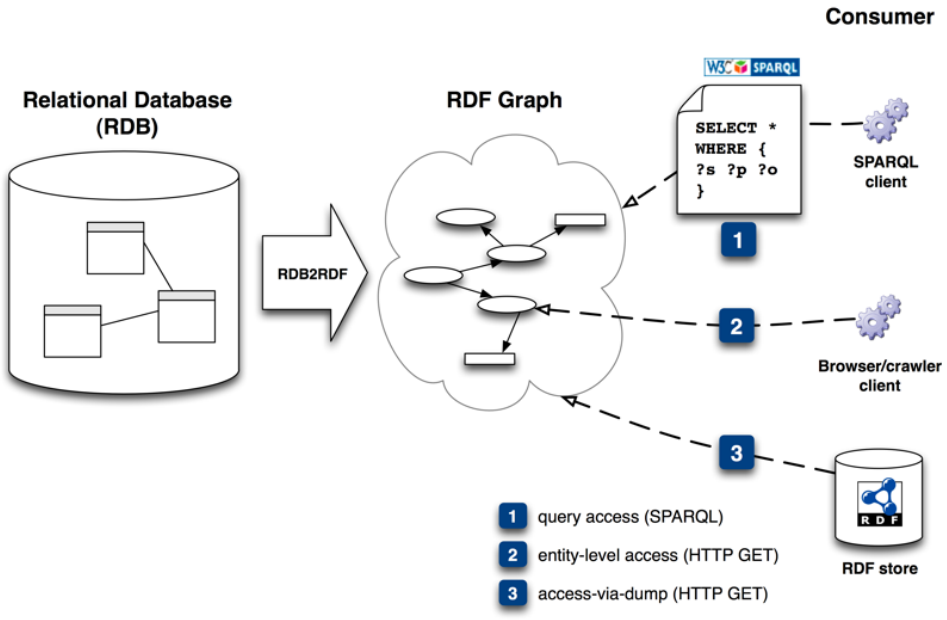


Figure 3.5.: RDB2RDF concept including data access by clients.

A client consuming the resulting (virtual or materialised) RDF graph essentially can access the data in the following ways:

1. Query access, which means the agent issues a SPARQL query against an endpoint exposed by the system and processes the results (typically the result is a SPARQL result set in XML or JSON);

2. Entity-level access, which means the agent performs an HTTP GET on a URI exposed by the system and processes the result (typically the result is an RDF graph);

3. Dump access, which means the agent performs an HTTP GET on dump of the entire RDF graph, for example in Extract, Transform, and Load (ETL) processes.

An RDB2RDF tool is D2R[5], which enables to publish the content of relational databases on the Web as Linked Data. D2R uses a customisable mapping to map database content into Linked Data. DERI is very active in the standardisation of RDB2RDF[6] within the W3C and currently implements the upcoming standards in D2R.

A Linked Data interface for SPARQL endpoints

Many triple stores and other SPARQL endpoints can be accessed only by SPARQL client applications that use the SPARQL protocol. Pubby[7], a tool co-developed by DERI, makes it easy to turn a SPARQL endpoint into a Linked Data server by offering a Linked Data and HTML interface. It is implemented as a Java web application. Pubby provides dereferenceable URIs by rewriting URIs found in the SPARQL-exposed dataset into the Pubby server's namespace. Furthermore it takes care of handling redirects (via HTTP 303 status code) as well as content negotiation for serving different serialisations (RDF/XML, Turtle, HTML) to the client.

Linked Data API

The Linked Data API[8] is a set of JSON-based APIs that are readily usable by developers who are not familiar with RDF or SPARQL. The API layer may be deployed directly by the publisher of the SPARQL endpoint or may be deployed by a third-party, for example as a local proxy to a remote endpoint.

[5] http://d2rq.org/
[6] http://www.w3.org/2001/sw/rdb2rdf/
[7] http://www4.wiwiss.fu-berlin.de/pubby/
[8] http://code.google.com/p/linked-data-api/

SPARQL endpoints considerations

While SPARQL endpoints are a useful interface to issue structured queries against RDF stores in general, the following issues should be taken into consideration when building applications on top of them:

- Performance and response time - performance usually means the time it takes for a system to execute a certain task. When we refer to response time in this context, we mean the execution duration of a task, measured in time per task, such as "queries per second". Response times of SPARQL endpoints (especially remote ones) can be crucial factors to be considered when building applications that access data from them. In DERI we have developed a profiler for Linked Data[9] that allows to measure individual and average response times of SPARQL endpoints.
- Availability - as SPARQL endpoints essentially constitute a distributed architecture, typically over the Web or in the Internet, their reliability is an important issue for the data access. In special the availability [10] of SPARQL endpoints, essentially their current status and overall up-time, is of interest.
- Bulk operations - while SPARQL endpoints provide an appropriate tool for structured queries they are not suitable for all kinds of operations, especially not for bulk loads (replication, mirroring, data warehousing etc.). For this, data dumps are better suited, in our experience.
- Look-ups - most of the available SPARQL endpoints provide look-ups and filtering backed up or complemented with full-text indices, such as provided by the Lucene-family. SIREn, the "Semantic Information Retrieval Engine"[11], is a plug-in for Apache Lucene to efficiently index and query RDF, as well as any textual document with an arbitrary amount of metadata fields.

3.3. Semantic Sensor Web

Standardisation is important in the IoT paradigm, because it increases interoperability and extendibility [Pan et al., 2013]. Semantic technologies are viewed as a key to resolving the problems of interoperability and integration within this heterogeneous world of

[9] http://github.com/mhausenblas/ld-profiler
[10] http://labs.mondeca.com/sparqlEndpointsStatus/
[11] http://siren.sindice.com/

ubiquitously interconnected objects and systems [Katasonov et al., 2008]. Thus, the IoT will become a **Semantic Web of Things**. Sensor metadata and sensor measurements represented as Linked Data (introduced in Section 3.2.3) are called *Linked Sensor Data*.

It is generally recognised that this interoperability cannot be achieved by making everyone comply to too many rigid standards in ubiquitous environments. Therefore, the interoperability can be achieved by designing middlewares [Katasonov et al., 2008] which act as a seamless interface for joining heterogeneous IoT applications together. Such a middleware can offer application programming interfaces, communications and other services to applications. Clearly, some data-centric standards are still necessary, in order to represent and describe the properties of the data in a homogeneous way across heterogeneous environments. The semantic approach to information systems design uses declarative descriptions of information and processing units, allowing (semi-)automatic satisfaction of declaratively described requirements. Declarative descriptions enable both domain-independent and domain-specific reasoning of various forms (logic-based or otherwise) to be applied in processes such as entity identification, search, and query and workflow generation.

The IoT requires a plethora of different middlewares, at different stages of the pipeline from data collection and cleaning to service enablement. The end goal is to have *Plug n'Play* smart objects which can be deployed in any environment with an interoperable backbone allowing them to blend with other smart objects around them. Standardisation of frequency bands and protocols plays a pivotal role in accomplishing this goal.

3.4. Challenges and Approaches

Based on "SPITFIRE: Towards a Semantic Web of Things" [Pfisterer et al., 2011b] published at the IEEE Communications Magazine (IEEE-CommMag 2011)

Sensors are ubiquitous in infrastructures, appliances, mobile phones, and wireless sensor networks. Their widespread deployment represents a significant financial investment and technical achievement and the data they deliver is capable of supporting an almost unlimited set of high value proposition applications. This is a powerful and profitable confluence of need, capability, and economic opportunity – yet the true potential of sensor technology is massively under-exploited.

A central problem hampering success is that sensors are typically locked into unimodal closed systems. For example, motion detection sensors in a building may be exclusively controlled by the intrusion detection system. Yet the information they provide could be used by many other applications, e.g., placing empty buildings into an energy-conserving sleep mode or locating empty meeting rooms. Unlocking valuable sensor data from closed systems has the potential to revolutionise how we live. To realise this potential, a service infrastructure is needed to connect sensors to the Internet and publish their output in well-understood, machine-processable formats on the Web thus making them accessible and usable at large scale under controlled access.

So far, the sensor world and the Web world have been largely disconnected, requiring the human in the loop to find, integrate and use information and services from both worlds in a meaningful way. Publishing sensor-related data on the Web would help to find relevant information by directly accessing sensor data, i.e., by directly observing the real world, integrated with related information from the Web. Already today, smart phone applications such as CenceMe [Miluzzo et al., 2008] exist that infer the activity of the person wearing the phone from sensor data and publish this in the Web. Another example are energy consumption sensors that end-users can install in their house to measure energy consumption of appliances, for example to compare their energy consumption with that of other, similar households to identify opportunities for saving energy. To do this easily, with open interfaces and data formats, and at large scale, technologies from the Web need to be customised for and integrated with their relevant counterparts on the Internet of Things (IoT). This means that application experts who are able to publish Web pages today should have the same easy-to-use technologies at hand to publish sensor descriptions, sensor data and make use of sensor outputs without requiring deep knowledge of embedded computing. In particular, we believe that users are primarily interested in real-world entities (things, places, and people) and their high-level states (empty, free, sitting, walking, ...) rather than in individual sensors and their raw output data. Therefore, the infrastructure must provide appropriate abstractions to map sensors and their raw output to real-world entities and their status representation.

Real-world entities are rarely useful when considered in isolation – the ability to put multiple entities into a common semantic context is needed. For example, we want to reason about rooms being in the same building, belonging to the same company, with nearby parking spots. This requires a machine-readable representation of world knowledge and appropriate reasoning capabilities. Further, this representation needs to be unified - while most sensor data published so far on the Web relies on heterogeneous

data models and serializations. In addition to discovery and query facilities on static properties of those machine-readable representations of sensors and real-world entities, specialized search approaches to support queries on the dynamically changing state of sensors or entities consisting of many sensors (possibly integrated with static data), will be required, e.g., which rooms in a building are currently occupied.

Imagine that sensors, which are connected to the Internet, measure the state of real-world entities such as meeting rooms and parking spots. Internet-connectivity not only requires network-level integration (IP), but also application-level integration to enable structured access to sensor data. To enable automatic reasoning about sensors (e.g., finding free parking spots close to meeting room), these sensors, their output, and their embedding into the real world must be *described in a machine-readable format* that is compatible with data formats used to describe existing world knowledge in the Web. Not only syntax and semantics of such a description must be defined, but efficient mechanisms to *annotate newly deployed sensors* with appropriate descriptions are required.

Users are primarily interested in *real-world entities* (e.g., meeting room) and their *high-level states* (e.g., room occupied) rather than sensors (e.g., sensor 536) and their raw output (e.g., motion detected at time T). Therefore, appropriate mechanisms to establish an explicit mapping of sets of sensors to real-world entities they are monitoring (e.g., all motion detection sensors in a certain room) must be provided. Further, the raw output of these sensors (e.g., motion detection events) must be mapped to a high-level state (e.g., room occupied). Often, this involves fusing the output of multiple sensors (e.g., multiple motion sensors are needed to cover a large room) or even scheduling sensors for energy efficiency (e.g., only one out of two available battery-powered motion sensors is required to cover a smaller room).

Finally, the user wants to *search for real-world entities by their current state* (e.g., empty meeting rooms). Often, such search requests refer not only to the output of sensors, but also to further machine-readable information that is available elsewhere in the Web (e.g., company maps, meeting schedules, calendars). The search engine needs to integrate these different static and dynamic data sources in a seamless way.

Realizing the above use case on an Internet scale requires

- that the sensors are connected to the Internet,
- that machines can discover and understand the semantics of the data returned by the sensors, and

- a technique to find the sensors that could provide the relevant data.

This section briefly discusses the state-of-the-art in relation to this with a focus on Internet-scale, Web-based technologies upon which we build our approach, employing the use case as an example.

There are efforts to realize a Semantic Sensor Web including the SENSEI [Villalonga et al., 2010a], SemSorGrid4Env [Partners, 2011], Exalted [Partners, 2013], and 52 North projects [Partners, 2014], as well as work by the Kno.e.sis Center [Patni et al., 2010], CSIRO [Compton et al., 2009], and the Spanish Meteorological Agency [AEMET, 2014]. Most notably, the Open Geospatial Consortium's (OGC) Sensor Web Enablement (SWE) [OGC - Open Geospatial Consortium, 2010] project builds a framework to publish and access sensor data using XML-based protocols and APIs. The choice of XML, however, ties SWE to system-specific schemas, providing neither semantic interoperability nor a basis for reasoning. This problem is in the focus of the Semantic Sensor Web which proposes annotating sensor data with semantic meta-data, whose meaning is machine-understandable through vocabulary definitions, i.e., an ontology [Sheth et al., 2008]. By annotating sensor-related features such as the network, deployment, data formats, etc., it becomes possible to automate further tasks, e.g., deployment, maintenance, and integration.

However, these efforts have limitations which we address in this thesis. There is no general-purpose approach compatible with the growing body of semantic world knowledge available as Linked Open Data (LOD) on the Web; existing efforts are either too sensor-centric or too knowledge-centric, i.e., they do not provide comprehensive, integrated abstractions for things, their high-level states, and how they are linked to sensors; and a number of important services are missing in existing efforts, notably support for semi-automatically creating Linked Data representations of sensors and things, as well as efficient search for things based on their current states.

SPITFIRE [Pfisterer et al., 2011b] addresses these limitations by providing 1. vocabularies to integrate descriptions of sensors and things with the LOD cloud 2. semantic entities as an abstraction for things with high-level states inferred from embedded sensors 3. semi-automatic generation of semantic sensor descriptions 4. efficient search for sensors and things based on their current states. In addition, SPITFIRE integrates these ingredients into a unified service infrastructure to ease adoption of the Semantic Web of Things for end-users and developers. On top of this infrastructure, applications are assembled by issuing search requests for matching (real or aggregated) sensor services

and by invoking found services directly. An overview of the SPITFIRE contributions to detailed challenges, is available in Appendix B.

The specific thesis contributions made by the author with respect to SPITFIRE are

- the merging and extension of existing vocabularies to integrate descriptions of sensors and things with the LOD cloud, into a new ontology;
- the automated creation of links with the LOD cloud;
- the automated semantic annotation of raw sensor data and metadata.

3.5. Conclusions

Existing Semantic *Sensor Web* technologies enable the integration of sensors into the Web, but the underlying model is focused on sensors rather than on things and their high-level states. The techniques for integration of sensors into the Web outlined so far are necessary but not sufficient to realize a Semantic Web of Things. In particular, semantic descriptions must be dynamically linked with external resources. They must also make use of widespread ontologies while at the same time not requiring a too steep learning curve which may otherwise prevent their adoption. These issues lead to our Research Question **Q 2. Communication**: *How can sensors communicate across different platforms without ad-hoc solutions?* We describe our answer in Chapter 4.

Also, semantic description should be:

1. integrated with the LOD cloud to support semantic reasoning;
2. semi-automatically created to let the technology scale;
3. represented as totally encapsulated in the abstraction concept they represent so as to facilitate the data handling and discovery.

We addressed these issues as a contribution to the SPITFIRE project (see Appendix A).

Furthermore, the search for things must be optimised in order to scale as the Semantic Web of Things vision becomes a reality. For this purpose, discriminating the relevancy sensors according to the task at hand is critical. This issue is represented by the Research Question **Q 3. Relevancy**: *How to identify which sensors are more relevant sources of*

information to define a specific small context scope - the Little Data - of interest?. We address it in Chapter 5.

Finally, an increasing gap is forming between the traditional Web and the Semantic Web of Things, where the broad range of average users are neither aware nor familiar with the advancements in sensing technologies. Research Question **Q 4. Quality**: *How can contextualised sensors improve the quality of traditional Web content?* We present our investigation on the subject in Chapter 6.

As it was difficult to foresee the wealth of current Web applications back when the Web was created, we have to wait and see how people will use the Semantic Web of Things. It is also hard to predict if a Semantic Web of Things will be as broadly adopted as the Web is today. One indicator is that LOD has already achieved significant uptake by governments (including UK, USA), the media sector (BBC), life sciences, geo information systems, and Web companies (Freebase). Making sensor data part of this data pool is clearly beneficial as then integration with knowledge from arbitrary sources is possible. For example, sensors and their data can be linked to geographic data (correlated natural phenomena), user-generated data (social feedback), government data (census information), life-science data (causes and effects of diseases), etc.

A strong indicator whether this line of development will be successful in the long run, is also provided by the exponentially growing amount of linked data and the support by major players. Since its beginnings in 2007, the LOD cloud has grown from 12 datasets to 203 data sets in 2010 with over 25 billion triples interlinked with 395 million links. Industry initiatives such as Google's Rich Snippet (2009), Facebook's Open Graph (2010), or very recently Schema.org (2011) all aim at adding semantic markups to web pages to improve search and discovery capabilities for the end user also confirm this uptake at Web-scale.

Part III.

Core

In the core part of the thesis we present our main contribution, consisting of research and applications towards better context-awareness in the Semantic Web of Things. First we apply semantic annotation and interlinking to sensor data. Our contributions here consist of both ontology modelling for context-aware linked sensor data and linked sensor data automated management and creation via *Linked Data for Sensors* (LD4S) to support the uptake of our approach. Second, we demonstrate the advantages of our approach based on linked sensor data. We investigate the advantage of predicting the relevancy of a sensor in the context of activity logging. We demonstrate the advantage for common Web users in terms of enriched information discovery. Our final contribution consists of demonstrating the improved adaptability of IoT systems, thanks to our approach. Each of the core chapters is based on published works.

We start from the premise of LD4S — a Web service to support storage, annotation and interlinking of sensor data according to the Semantic Web technologies, context-aware sensor vocabularies and principles. It provides the framework we need for building our applications that improve sensor relevancy prediction, Web data enrichment and IoT systems adaptability. The order of the chapters reflects the logical sequence of steps to deploy the IoT improvements we propose. Hence, we start from the basis in Chapter 4, by describing our system to create semantic annotations and linked data for sensors. In this context, we present the challenges we found and our solutions to the Research Questions **Q 1. Context** - *How can contextual information be used to enrich sensor data?* - and **Q 2. Communication** - *How can sensors communicate across different platforms without ad-hoc solutions?* -.

Once we explain the foundation of our approach in Chapter 4 - such as context-aware semantic modelling, annotating and interlinking for sensor data - we demonstrate how to exploit their potential. **First**, in Chapter 5 we describe a method for predicting the relevancy of a sensor with respect to the activity logging task at hand. We achieved an improvement over the state of the art with this proposed solution to the Research Question **Q 3. Relevancy** - *How to identify which sensors are more relevant sources of information to define a specific small context scope - the Little Data - of interest?* -.

Second, In Chapter 6 we exploit the demonstrate the linked sensor data discovery capability enhanced by our approach. We inject short lived sensor data into common Web search results. The purpose is to bridge the gap between traditional Web of Document and the Semantic Web of Things, depicting the advantages for both final users and website developers. We address the Research Question **Q 4. Quality** - *How can contextualised sensors improve the quality of traditional Web content?* - considering the lack of live

and short-lived information into traditional Web pages. The solution we present is not limited to the use case of Web search results, but it is valid also for any type of Web content.

Chapter 4.

LD4S: Linked Sensor Data Provisioning

Based in part on "inContext Sensing: LOD augmented sensor data"
[Leggieri et al., 2011a]
published at the 10th International Semantic Web Conference (ISWC2011)

In this chapter, we firstly address the Research Question **Q 1. Context** - *How can contextual information be used to enrich sensor data?* - by modelling an ontology for contextualised sensor metadata. We designed, modelled and implemented this ontology as detailed in Section 4.2.2 and we extended it to enable the collection of user feedback, as in Section 4.2.3. This work constituted the main contribution to the SPITFIRE Deliverable D2.1 [Leggieri et al., 2012a], as specified in Appendix A.

We designed and implemented through the whole life-cycle our main contribution, the LD4Sensors web service (LD4S), its architecture, REST API and GUI, as explained in Section 4.3. Section 4.4 describes the successful results of our system evaluation. LD4S aims at enabling cross-network communication among sensors as required by the Research Question **Q 2. Communication** - *How can sensors communicate across different platforms without ad-hoc solutions?* -. We investigate the related work in Section 4.5. This work constituted the main contribution to the SPITFIRE Deliverable D2.4 [Leggieri et al., 2013a], as specified in Appendix A.

Section 4.2.1 depicts a scenario to showcase the advantages of LD4S and its link customisation, as also demonstrated by each of the other core chapters of this thesis. In fact, innovative solutions for sensor selection based on relevancy prediction, sensor

discovery and IoT systems adaptiveness are presented respectively in Chapter 5 and Chapter 6.

4.1. Community Sensing

User feedback and sharing have already been successfully applied to improve data quality but never on sensor data in ubiquitous systems. For instance, crowdsourcing - multiple small contributions from various individuals towards a larger body of work - is a currently a broad phenomenon [Howe, 2006], with Wikipedia [Lih, 2009] being the most well-know example. Citizen reporting refers to social media updates from many contributors on a particular event of interest, and citizen sensing [Boulos et al., 2011] systems, e.g., Twitris [Jadhav et al., 2010], have relied on these citizen reports to some effect. Although Sensorpedia[1] and Cosm[2] are platforms to share sensor observations, the stored data is difficult to disambiguate and reuse because of the absence of semantic annotations. In addition, sensor information that is relevant for the reuse is often missing, e.g. unit of measurement.

4.1.1. Web 2.0 Lessons Learnt

The data source capabilities also influence the data quality requirement, together with the user preferences, since quality can often be subjective parameter. The measurement capabilities, however, might often be difficult or impossible to retrieve. In these cases, users might be supportive in determining such capabilities by providing their feedback. The feedback is then used to assess the data quality both in terms of user preference and of data source capabilities; thus allowing our data model to fully accomplish the first of the listed requirements. This crowdsourcing approach has already proven to be successful in several applications, most notably, in the Web 2.0 context. We can apply some of the notions of crowdsourcing or citizen sensing/reporting to sensor data generation and collection, by incorporating a human element in the distribution of sensor data, i.e., votes, comments, reviews provided by average users for the annotation, the links and the sensed activities themselves.

[1] http://www.sensorpedia.com/
[2] http://cosm.com/

While sensor data is normally perceived to be somewhat objective, there is a lack of context information about who created a particular sensor or how a person rated a sensor reading which could add some valuable subjective context, especially when observed in a network of people and sensors. We can apply some of the notions of crowdsourcing or citizen sensing/reporting to sensor data generation and collection by incorporating a human element in the distribution of sensor data.

Crowdsourcing and Citizen Sensing are examples of user input collection, in which humans acting as sensors improve the data delivery. However, apart from some notable examples of geo-location data gathered from GPS[3] or mobile/network access points by a multitude of people, the idea of human-driven reporting and sharing of arbitrary sensor data has not yet become widespread [Sheth et al., 2013].

The owner of a particular sensor could help a decision maker to act on a particular situation if the sensor is attached to a person with a valid reputation. Also, users may choose to comment or rate a particular activity associated with a sensor reading, and this in turn can serve as useful contextual information when one is trying to base a decision or carry out an action based on sensor data.

Privately owned sensors such as cameras, GPS devices, cell-phones, and home weather stations are bountiful in the world. In principle, the data from large populations of such sensors could be harnessed to provide valuable services. For example, GPS devices, which are becoming popular integral components of smartphones and automobiles, could provide real-time traffic monitoring services with extensive coverage. In practice, however, privately-held sensor data is rarely shared because of privacy concerns of the sensor owners or others whose privacy might be violated by such sensing. Beyond privacy considerations, applications depending on authorization via real-time requests for data could be disruptive and annoying to owners. Furthermore, owners may not wish to donate battery and networking resources required for sensing and transmitting data.

Sensor ontologies by themselves do not incorporate this notion of user rating or commenting on or in conjunction with some sensor data. However, through a combination of (a) the *Association ontology*, which allows two things to be connected through a rated association, (b) the *Review ontology*, describing various terms associated with online reviews, and (c) *SIOC* and SIOC Types schema, which models reviews and comments,

[3]The Global Positioning System (GPS) is a space-based satellite navigation system that provides location and time information.

we enable people to comment in parallel with a sensed value or subsequently through comments and feedback ratings.

4.1.2. Requirements

. For the seamless data merging that we enable with the Community Sensing, the main requirements are

- aggregated data quality. When aggregating data one must specify the data provenance since that determines the level of trustworthiness. Given the difference in necessities, requirements and preferences of users with respect to aggregated data, the user profile should also be taken into account. The selection of data to aggregate should be customised on the specific user needs. Finally, the data source capabilities themselves may affect the aggregated data quality in terms of how this data is exposed, annotated, linked and accessed;
- semantics of the context aggregation. Distinct contexts may be linked for several reasons from causation to correlation or spatial proximity. It is important to explicitly specify the nature of the relationship that links two contexts, since reasoning engines and incremental machine learning algorithms may rely on such past relationships to determine new ones.

4.2. The Design of LD4S

Based in part on "Ontologies for representing sensor information"
[Leggieri et al., 2012a]
published as the Spitfire Deliverable 2.1

So far we have given an overview of our approach, how it was inspired by Community Sensing and the requirements to address. Here we describe first a scenario in Section 4.2.1 that we aim at enabling. It motivated our design of LD4S which starts with the ontology modelling as described in Section 4.2.2.

4.2.1. Use Case: Dynamic Building View

There exist several possible scenarios in which a sensor ontology can be useful to easily merge different data sources and provide detailed descriptions that can be used to perform both supervised and unsupervised learning. In particular, energy conservation is the core driver of this use case. In a multi-tenant office building, devices and rooms are monitored in order to apply energy conservation plans. Then, different scenarios can occur at different levels of granularity, as follows:

- Room / office / building switched into sleep mode when either not reserved or unoccupied;
- Room / office energy usage analysis;
- Appliance level energy usage analysis: sensor-instrumented appliances (anything from air-conditioning units and photocopiers to kettles and toasters) can be monitored to determine their relative operating efficiencies.

However the majority of buildings do not host sensor deployments at all or only in some areas. In both cases, a dynamic interaction with mobile sensors as they enter the scope of the building can yield benefits. For instance, even if no motion sensor is installed in a room, detecting the continuous movement of a sensing device, could indicate, with high probability, a human presence. Similarly, even in the case where a printer's energy consumption is not being monitored, counting the amount of times in which a mobile device is detected entering the printing room provides an indication of printer usage, thus how much energy and ink have been consumed. The sharing of such sensor data and metadata annotations would facilitate the collection of useful information about the environment.

Consequently, metadata about each different component involved in the scenario can be easily merged in order to

1. group descriptions that refer to the same real world object (Semantic Entity creation [CTI et al., 2011]);
2. automate both cross-domain and cross-building energy consumption analysis;
3. automatically infer which devices can be disabled, by considering the semantic description of their role in the overall network structure.

These different data sources can be integrated by representing them as Linked Data. This linkage enables the following achievements:

- defining a threshold to identify employees who waste too much energy with respect to their job tasks;
- disabling sensor nodes that have recently been under-used and are not critical for the overall network functionality;
- coordinating time usage and switching among the different facilities according to the linked bill data plan.

We can imagine the occurrence of a sequence of interaction steps as follows:.

- A user asks the Intelligent Building Automation System (IBAS), which employee saves more energy in the office building.
- IBAS requests and receives a list of Appliances from the Device Manager.
- For each appliance, IBAS asks the Efficiency Service to get a *threshold value* about the maximum amount of energy that should be consumed. It then gets the energy consumption threshold and an associated time range for each device.
- IBAS requests and receives a list of employee URIs from the Company Manager software.
- For each employee, IBAS requests and receives the list of corresponding job tasks from the Company Manager software.
- For each device, IBAS requests and receives the list of devices which the specified employee has been submitting tasks from the Device Manager.
  - IBAS receives from the Link Finder either one or no link between the specified device and the job tasks (this should work as a *tolerance level*). Such a link would indicate that the specified task has been assigned to the specified device.
  - IBAS requests and receives from the Efficiency Service a calculation of the credits for the specific employee (e.g., for being green and avoiding pollution. This should be based on both the threshold and tolerance level.

The usage of ontologies and Linked Data makes it possible to cross-query all the different components / datasets listed above, and additionally to enable inference and automated tasks. In other words, this example scenario shows the need of a context-aware sen-

sor ontology to be created. We describe the modelling of such ontology in the next Section 4.2.2.

4.2.2. Context-Aware Sensor Ontology

The ontology we designed, supports the users in many scenario based on energy conservation, which is the core driver of our use cases. The issues we focused on were those based on both in-network and cross-building energy conservation. Usually sensors are intended to be physically small and inexpensive, thus making it possible to produce and deploy them in large numbers. In order to meet the objective of the sensors being small and low-cost, resources in terms of energy, memory, computational speed and bandwidth are severely constrained. Moreover sensors use each other to transport data to a monitoring entity. Then while each sensor has a limited energy supply, it is important that sensors conserve their energy for the network to last longer. An energy conservation practise consists of temporarily disabling those nodes that are not critical for the network activity and have been inactive for some time. Precisely identifying such nodes to be disabled is an issue especially in case of multiple network layers and nodes characterized by different access and description schema. Consequently we address this problem together with the detection of an optimal energy consumption threshold, as described below.

1. **Application of Energy Saving measures**: detecting sensor nodes that can be disabled to save energy. Currently our sensors detect some values that are useful for running energy saving algorithms. In particular one can infer whether a node is critical for a network (e.g. as either clusterhead or routing nodes) or not by considering both the total amount of bytes or packets received and the total amount of used RAM. In case a non-critical node has also been inactive for some time, then it is eligible to be disabled, to preserve energy efficiency. Still, when deciding whether to disable it, additional factors should be taken into consideration in order to avoid mistakes and failures. For instance a node might not be a clusterhead one but, being part of the core layer in a hierarchical network topology, it shouldn't be disabled. Even the particular topology in use can be meaningful by implying more or less redundancy of node interconnections.

2. **Performance calculation of Energy Saving measures**: detecting an optimal energy consumption threshold. When comparing local energy saving solutions against external ones, eventual inefficiency or leaks can be highlighted, that could

not be noticed otherwise (cross-building comparison as depicted in Section 4.2.1). For such a comparison to be meaningful, it is important to unambiguously define the purpose of both the whole building and the particular sensor network deployment located inside it. A defined purpose implies specific tasks that both humans using the energy consuming devices and sensors monitoring such consumption, are expected to perform. Such purposes can be unambiguous and machine-understandable, if represented by ontological concepts modelled as RDF, so that broad and automated links can be created among similar data to be compared.

3. **Surrounding Context**: the energy saving policies like any other kind of decision-making processes that are based on sensor observations, need to exploit the complex structure of the sensed events. For instance observing that one printer has consumed more ink during the last month, is of low interest if not combined with the observation of the exceptions triggered by the other printers in its close proximity and during the same time period, with a correlation kind of relation, i.e., these events occur at the same time but have no direct causality.

Such issues are addressed by the modules composing our ontology i.e., both the *SensorNetwork* and *NetworkComponent* modules target the first issue while both the *SensorProject* and the *Energy* modules target the second one. In fact the *SensorNetwork* and *NetworkComponent* modules enable nodes filtering based on the implemented topology, nodes' roles and the layer these nodes belong to, since they provide concepts to describe each of such characteristics. In this way less critical nodes to be disabled can be more precisely identified, thus addressing the first of the issues above. Similarly the *SensorProject* and the *Energy* modules - as in Figure 4.1 - enable the detection of an optimal energy consumption threshold by providing concepts to properly describe similar datasets to be compared and saved energy. Thus the optimum setting can be selected, while the second of the issues above is addressed. Each one of the new enabling concepts introduced, is detailed in the next following sections. All the modules of the SPITFIRE vocabulary are illustrated in Figure 4.1. The Energy module includes concepts related with the Electrical Energy consumption, providing support, in this way, to the SPITFIRE consolidated use cases, whose core driver is Energy consumption. These Energy consumption characterizes both Network components and interconnected sets of them (Sensor Networks). Network Components are linked with the Sensor Network they belong to, which is indeed, deployed for a higher purpose i.e. Sensor Project. Finally the *Context* module enables an enriched description of any sensed occurrence and its surrounding environment, as in Figure 4.1.

In order to add robustness, as already done by the SSN ontology [Compton et al., 2012a], we decided to align our vocabulary with the same upper level ontology as SSN had been aligned: Dolce+DnS Ultralite. An additional rationale is that Dolce+DnS Ultralite includes Ontology Design patterns that specifically target proper description of events and situations. Though to describe events in a comprehensive way, an ontology more specific than an upper level one is needed. In particular we want to structure the relations among an event and other events or entities involved in it. Then we aligned with an Event ontology, too: the Event Model-F ontology. Both the Dolce+DnS Ultralite and the Event Model-F ontologies are described in the next sections.

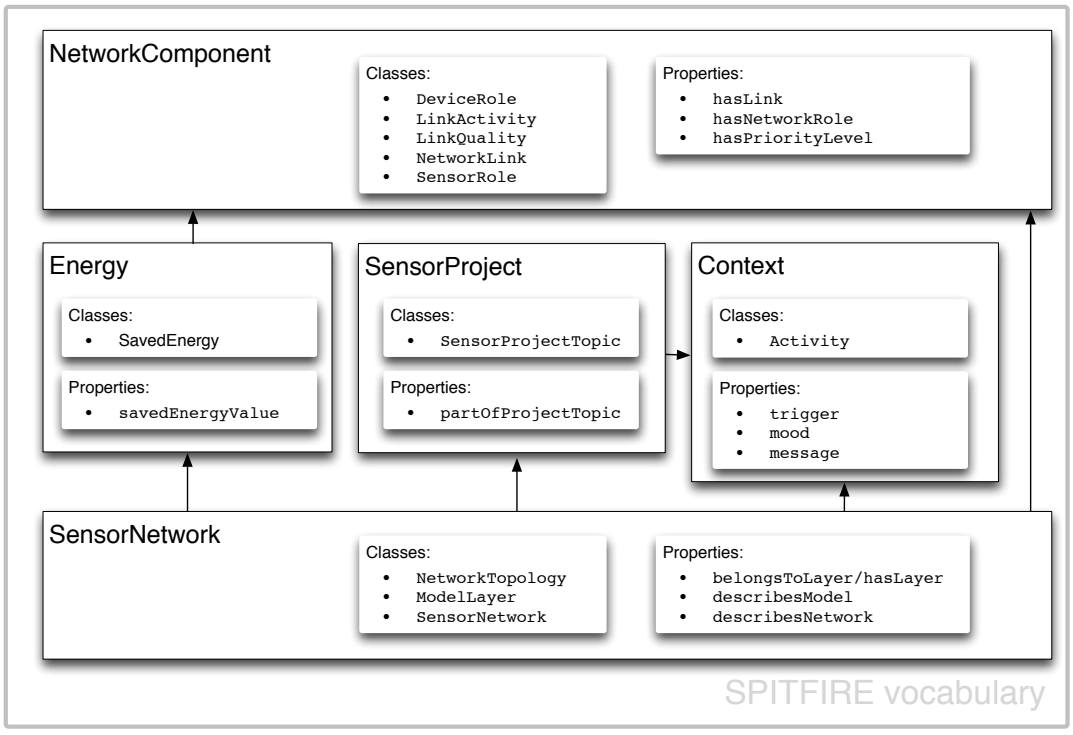


Figure 4.1.: Modules in which the SPITFIRE vocabulary is divided. For each module the main concepts, the main predicates and its relations with the other modules are depicted, too.

The Sensor Network Module

The *SensorNetwork* module includes a representation of a Sensor Network with its topology and eventual corresponding layers. Then the main concepts that belong to this module are *Sensornetwork*, *NetworkTopology*, *ModelLayer* and *NetworkLink*, as defined in the next paragraphs.

Definition: Sensor Network The SPITFIRE vocabulary allows to describe a Sensor Network. A **Sensor Network** consists of spatially distributed sensors, which cooperatively monitor physical or environmental conditions, and are interconnected by each other. The type of interconnection is a link that can be either wired as a link or wireless.

Since a network would not exist without its components, it can also be seen as a set of descriptions of certain nodes and links which communicate with each other.

In the SPITFIRE ontology a Sensor Network is modelled as an instance of the *spt: Sensor-Network* class. Such a *spt: Sensor-Network* instance is a single RDF resource that represents the entire Sensor Network, and thus allows us to easily make statements about the entire network and all its components.

The relationship between a *spt: Sensor-Network* instance and the components contained in the Sensor Network is established through the predicate *spt: belongs-To-Network*.

The example in Listing 4.1 declares the resource *:SPITFIRENetwork* as a *spt: Sensor-Network* together with its components.

```
1 :SPITFIRENetwork a spt:SensorNetwork  .
2 :platform12 a ssn:Platform ;
3  spt:belongsToNetwork :SPITFIRENetwork .
```

Listing 4.1: Example of using the SPITFIRE vocabulary to describe a Sensor Network and one of its components.

The resource *:SPITFIRENetwork* is intended as a proxy for any of the Sensor Network deployed by the SPITFIRE partners [partners all, 2011]. A good next step would be to make this unambiguously clear by adding general metadata like Web page links and basic Dublin Core metadata.

Definition: Network Topology The SPITFIRE vocabulary allows to describe a Network Topology. A **Network Topology** is the layout of the connections of a network. Topology control is fundamental to solve scalability and capacity problems in large-scale and ad-hoc sensor networks. The forthcoming multi-hop networks will allow network nodes to control the communication topology by choosing their transmitting range or scheduling nodes to sleep. In order for this to happen, the topology must be unambiguously described in a machine-understandable format using the concept *spt: Network-Topology* provided by the SPITFIRE ontology.

Network topologies may be 1. Physical 2. Logical.1. Physical topology means the physical design of a network including the devices, location and cable installation. 2. Logical topology refers to how data is actually transferred in a network as opposed to its physical design. We model only the Physical Network Topology because low level hardware and protocol details about the network communication which characterise Logical Topologies are out of the scope of this thesis. With respect to Physical Network Topologies, they can be modelled as follows:

- `Flat Topology` is the simplest one, consisting of a permanent link between two endpoints. It is represented by the concept *spt: Flat-Model* from the SPITFIRE vocabulary. *spt: Flat-Model* is a subclass of *spt: Network-Topology*.
- `Bus Topology` is typical of local area networks and consists of having each component connected to a single bus cable through some kind of connector. It is represented by the concept *spt: Bus-Model* from the SPITFIRE vocabulary. *spt: Bus-Model* is a subclass of *spt: Network-Topology*.
- `Star Topology` is typical of local area networks and consists of having each network host connected to a central hub with a point-to-point connection. All traffic on the network passes through the central hub. It is represented by the concept *spt: Star-Model* from the SPITFIRE vocabulary. *spt: Star-Model* is a subclass of *spt: Network-Topology*.
- `Ring Topology` is set up in a circular shape in which data travels around the ring in one direction and each device on the right acts as a repeater to keep the signal strong. It is represented by the concept *spt: Ring-Model* from the SPITFIRE vocabulary. *spt: Ring-Model* is a subclass of *spt: Network-Topology*.
- `Mesh Topology` consists of having each node not only capturing and disseminating its own data, but also serving as a relay for other nodes i.e., it must collaborate to propagate the data in the network. It is represented by the concept *spt: Mesh-Model* from the SPITFIRE vocabulary. *spt: Mesh-Model* is a subclass of *spt: Network-Topology*.
- `Hierarchical Topology` consists of having a central *root* node as the top level of the hierarchy, connected to one or more other nodes that are one level lower in the hierarchy, with a point-to-point link between them. Nodes in this level can be linked with other nodes in the next lower level respectively. It is represented by the concept

spt: Hierarchical-Model from the SPITFIRE vocabulary. *spt: Hierarchical-Model* is a subclass of *spt: Network-Topology*.

- The Hybrid topology consists of a combination of any two or more topologies, so that the resulting network does not exhibit one of the topologies described above. It is represented by the concept *spt: Hybrid-Model* from the SPITFIRE vocabulary. *spt: Hybrid-Model* is a subclass of *spt: Network-Topology*.

The relationship between a *spt: Network-Topology* instance and the Sensor Network it designs is established through the predicate *spt: describes-Network*.

The example in Listing 4.2 declares the resource *:SPITFIREHybridDesign* as a *spt: Hybrid-Model* together with the Sensor Network in which such model has been implemented.

```
1 :SPITFIREHybridDesign a spt:HybridModel  ;
2  spt:describesNetwork SPITFIRENetwork .
```

Listing 4.2: Example of using our vocabulary to describe a Network Topology and associate it to a Sensor Network.

The resource *:SPITFIREMeshDesign* is intended as a proxy for any of the network topology implemented in the sensor networks that we deployed for our demo. A best practice is to make the layers defined by this Mesh topology clear, by establishing a relationship with a *spt: Model-Layer* instance as described in the next paragraph.

Definition: Model Layer The SPITFIRE vocabulary allows to describe a Layer of a particular Network Topology. The network layout unless it consists of a point-to-point type of layout, delineates subsets in which the whole set of network components can be divided, according to either their physical or their logical inter-relationship. In the SPITFIRE ontology a Model Layer is modelled as an instance of the *spt: Model-Layer* class. As an example, we can consider the layers usually defined by the Hierarchical model i.e., Core, Access and Distribution layers, which are described as follows.

- Core Layer. The Core Layer is the backbone of a network, whose included network components must be highly reliable and switch traffic as fast as possible to provide fault isolation and backbone connectivity. It is represented by the concept *spt: Core-Layer* from the SPITFIRE vocabulary. *spt: Core-Layer* is a subclass of *spt: Model-Layer*.

- Distribution Layer. The Distribution Layer acts as an interface among the Core and the Access layers, providing routing, filtering and WAN access. Unlike the Core and Access layers, it is multi-purpose and usually consists of an aggregation point for the Access Layer switches. It is represented by the concept *spt: Distribution-Layer* from the SPITFIRE vocabulary. *spt: Distribution-Layer* is a subclass of *spt: Model-Layer*.

- Access Layer. The Access Layer is the edge of the entire network, in which a wide variety of consumer devices attach to the wired portion of the network. It is presented by the concept *spt: Access-Layer* from the SPITFIRE vocabulary. *spt: Access-Layer* is a subclass of *spt: Model-Layer*.

The relationship between a *spt: Model-Layer* instance and the Network Topology that defines it, is established through the predicate *spt: layerOf*.

The example in Listing 4.3 declares the resource *: SPITFIRE-Hybrid-Design-Layer* as a *spt: Core-Layer* of the Network Topology *: SPITFIRE-Hybrid-Design*.

```
1 :SPITFIREHybridDesignLayer a spt:CoreLayer  ;
2  spt:layerOf :SPITFIREHybridDesign .
```

Listing 4.3: Example of using the SPITFIRE vocabulary to describe a Layer of a Network Topology.

The resource *: SPITFIRE-Mesh-Design-Layer* is intended as a proxy for any kind of layer defined by the network topology implemented in the SPITFIRE partners' sensor networks. A good next step would be to establish a relationship among the specified layer and the network components included in its scope. This would be achieved by adding to the SSN ontology-based description of sensors and devices, the SPITFIRE predicate *spt: belongs-To-Layer* associating them with the network layer they belong to, as shown in Listing 4.4 .

```
1 :platform12 a ssn:Platform ;
2  spt:belongsToLayer :SPITFIREHybridDesignLayer .
```

Listing 4.4: Example of using the SPITFIRE vocabulary to describe a component of a Network Topology Layer.

The Network Component Module

The *Network-Component* module includes representations of all the components of a Sensor Network and their role in the network, according to the implemented Network

Topology. Then the main concepts that belong to this module are *Sensor-Role*, *Device-Role* and *Network-Link*, as defined in the next paragraphs; while sensors and device are already defined in the SSN ontology [Compton et al., 2012b].

Definition: Network Component Role The Network Topology by specifying the data transmission paths among network components, delineates specific roles for these components. For instance all the data that need to be routed will reach a node whose role is Routing (routing node); while data that must be provisioned to the OS or controlled, will reach a node whose role is the Head Node in a Cluster (clusterhead node). In fact, nodes have different roles and hierarchies to enable the programmed data flow within the network.

In our ontology, a network component role is modelled as either an instance of the *spt: SensorRole* class or an instance of the *spt: Device-Role* class, depending on the component being a sensor or a device, as defined by the SSN ontology.

The relationship between a *spt: Device-Role* or *spt: Sensor-Role* instance and the device or sensor on which the specified role applies, is established through the predicate *spt: network-Role*.

The example in Listing 4.5 declares the resource *:SPITFIRE-Clusterhead-Role* as a *spt: Device-Role* and assigns this role to the device *:SPITFIREDevice1*. Similarly the *ssn: Sensor* instance *: SPITFIRE-Node1* is assigned the role *: SPITFIRE-Routing-Role* as a *spt: Sensor-Role*

```
1 :SPITFIREClusterheadRole a spt:DeviceRole  .
2 :SPITFIREDevice1 a ssn:Device ;
3  spt:networkRole
4  :SPITFIREClusterheadRole .
5 :SPITFIRENode1 a ssn:Sensor ;
6  spt:networkRole
7  :SPITFIRERoutingRole .
```

Listing 4.5: Example of using the SPITFIRE vocabulary to describe roles of Sensor Network components.

The resources *: SPITFIRE-Clusterhead-Role* and *: SPITFIRE-Routing-Role* are intended as proxies for any kind of roles that can be defined by specific implemented logical network topologies in the SPITFIRE partners' sensor networks. A good next step would be to make the importance of the role clear, with respect to the overall network

functionalities, by using the SPITFIRE predicate *spt: priority-Level* and then match with a meaningful value, as shown in Listing 4.6.

```
:SPITFIREClusterheadRole a spt:DeviceRole  ;
 spt:priorityLevel "1" .
```

Listing 4.6: Example of using the SPITFIRE vocabulary to describe the importance of Sensor Network component role,with respect to the overall correct functioning of the Sensor Network.

The predicate *spt: priority-Level* indicates the importance of the role with respect to the proper functioning of the overall network. Its value is inversely proportional to the importance: the lower the value the higher the importance.

Definition: Network Link The SPITFIRE vocabulary allows to describe a point-to-point Link in a Sensor Network. A point-to-point link is a dedicated link that connects two communication facilities e.g., two sensor nodes. It can be also intended as an information transmission path. In the SPITFIRE ontology a Network Layer is modelled as an instance of the *spt: Network-Link* class.

The relationship between a *spt: Network-Link* instance and the two facilities that it connects, is established through the predicate *spt: link-Of*; while the relationship between each node and the link is established through the predicate *spt: link*.

The example in Listing 4.7 declares the resource *:SPITFIRE-Link12* as a *spt: Network-Link* that connects the sensor nodes *: SPITFIRE-Node1* and *: SPITFIRE-Node2*.

```
:SPITFIRELink12 a spt:NetworkLink ;
 spt:linkOf :SPITFIRENode1 ;
 spt:linkOf :SPITFIRENode2 .
:SPITFIRENode1  a ssn:Sensor  ;
 spt:link :SPITFIRELink12 .
:SPITFIRENode2 a ssn:Sensor ;
 spt:link :SPITFIRELink12 .
```

Listing 4.7: Example of using the SPITFIRE vocabulary to describe a Network Link and associates it to the two communication facilities that it is connecting.

The resource *:SPITFIRELink12* is intended as a proxy for any kind of link existing in the SPITFIRE partners' sensor networks. The best practise would be to establish a relationship among the specified `Link` and its `Quality`. In fact interference resources and noisy environments can be located by reasoning on the Quality of a Link. Similarly

faults can be detected and, as a consequence, workarounds and maintenance can be automatically applied.

In the SPITFIRE ontology the Quality of a Network Link is modelled as an instance of the *spt: LinkQuality* class, which can be related with the Link it refers to by using the SPITFIRE predicate *spt: linkQuality*, as shown in Listing 4.8.

The same example in Listing 4.8 declares the resource *: SPITFIRE-Link12* as characterized by the quality *: SPITFIRE-Link Quality12*. *:SPITFIRELinkQuality12* represents the Latency of the Link, as a specific Link Quality. Then the Latency is modelled as an instance of the *: SPITFIRE-Latency* subclass of the *spt: Link-Quality* class. A latency value has also been specified together with the unit of measurement in use. The inverse of the predicate *spt: link-Quality* is the predicate *spt: link-Quality-Of* that relates a `Link Quality` to the `Link` it refers to.

```
1 :SPITFIRELatency rdfs:subClassOf spt:LinkQuality .
2 :SPITFIRELinkQuality12 a :SPITFIRELatency ;
3  spt:linkQualityValue "0.8" ;
4  muo:measuredIn ucumunit:time/second ;
5  spt:linkQualityOf :SPITFIRELink12 .
6 :SPITFIRELink12 spt:linkQuality
7  :SPITFIRELinkQuality12 .
```

Listing 4.8: Example of using the SPITFIRE vocabulary to describe the Quality of a Network Link in terms of it Latency per second.

The resource *: SPITFIRE-Link-Quality12* is intended as a proxy for any kind of Link Quality that can be associated with a Link in the SPITFIRE partners' sensor networks. In fact it is up to the user to define ad-hoc link qualities according to his necessities, as sub-classes of *spt: Link-Quality*, as has been done above for the Quality *: SPITFIRE-Latency*. The best practise would be to establish a relationship among the specified Link and its Activity. A specification of the Link Activity can be used as a filter criteria while searching for inactive nodes to be disabled so that energy can be saved.

The Link Activity is intended as the total amount of packets sent along a Network Link during a specific time range. In the SPITFIRE ontology the Activity of a Network Link is modelled as an instance of the *spt: Link-Activity* class, which can be related with the Link it refers to by using the SPITFIRE predicate *spt: link-Activity*, as shown in Listing 4.9. The inverse of the predicate *spt: link-Activity* is the predicate *spt: linkActivityOf* that relates a Link Activity to the Link it refers to. The relation among a *spt: Link-Activity* instance and its value, is established by the predicate *spt: link-Activity-Value*. Also a

spt: Link-Activity instance can be linked with the time range during which it occurred, by the predicates *spt: start-Activity-Date-Time* and *spt: end-Activity-Date-Time*.

The example in Listing 4.9 declares the resource *: SPITFIRE-Link-Activity12* as a *spt: Link-Activity*. Specifically during a time ranging from 8:00 AM until 8:05 AM of the 26th of March 2012.*: SPITFIRE-Node2*, 35 packets have been sent through the Network Link *: SPITFIRE-Link12*.

```
:SPITFIRELink12 spt:linkActivity  :SPITFIRELinkActivity12 .
:SPITFIRELinkActivity12 a spt:LinkActivity ;
 spt:linkActivityOf :SPITFIRELink12 ;
 spt:linkActivityValue "35" ;
 spt:startActivityDateTime "12-03-26T8:00Z" ;
 spt:endActivityDateTime "12-03-26T8:05Z" .
```

Listing 4.9: Example of using the SPITFIRE vocabulary to describe the Activity level occurred in a Network Link during a specific range of time.

The resource *: SPITFIRE-Link-Activity12* is intended as a proxy for any kind of Link Activity that can occurr in a Network Link of the SPITFIRE partners' sensor networks.

Sensor Project Module

Part of the context information that characterises sensor metadata includes the metadata of the project or experiment within which the sensing has occurred. We provide the Sensor Project module to support the modelling of this information. Several scenarios like searching for sensor-based experiment results on a specific subject, rely on this capability.

Definition: Sensor Project Topic Behind the deployment of a Sensor Network, there exist different purposes at different levels of granularity, and the deployment itself is part of a broader project. Usually a main overall topic leads this project and by unambiguously specifying it, several further conclusions can be inferred. For instance the same Sensor Network that includes temperature sensors, can be used by three different projects focusing respectively on device maintenance, weather forecasts and building automation. In case the topic is weather forecast then we can infer that the context of the deployment is an outdoor one, while if it is building automation we can infer that actuators are involved and building facilities are the main feature of interests.

In SPITFIRE, the topic of a project, in which sensors are involved, is modelled as an instance of the *spt: Sensor-Project-Topic* class.

The relationship between any either network component or set of network components and a *spt: Sensor-Project-Topic* instance to which they take part, is established by the SPITFIRE predicate *spt: part-Of-Project-Topic*, whose inverse is the predicate *spt: project-Topic-Of*.

The example in Listing 4.10 declares the resource *: SPITFIRE-IoT* as a *spt: Sensor-Project-Topic* associated with the *spt: SensorNetwork* instance *: SPITFIRE-Network*.

```
:SPITFIREIoT a spt:SensorProjectTopic  ;
 spt:projectTopicOf :SPITFIRENetwork .
:SPITFIRENetwork a spt:SensorNetwork ;
 spt:partOfProjectTopic :SPITFIREIoT .
```

Listing 4.10: Example of using the SPITFIRE vocabulary to describe the topic of a project in which sensors are involved, and associate it to a Sensor Network.

The resource *: SPITFIRE-IoT* is intended as a proxy for any kind of high-level purpose that might motivate the SPITFIRE partners' sensor networks. The best practise would be to link by using the predicate *owl: same-As* the specified topic with a concept from a Foundation ontology since this would be connected with a broader network of other related concepts and then, when searching for similar sensor project topics, enable the retrieval of more detailed results.

The Energy Module

The *Energy* module includes representations of all the concepts related with **Energy**. This SPITFIRE Module directly match with the SSN Energy Module, as this was meant, indeed, to be a plug-in point to facilitate the extension of the SSN ontology. The main concept that belongs to this module is *Saved-Energy*, as defined in the next paragraph.

Definition: Saved Energy The SPITFIRE vocabulary allows to describe the amount of Energy that has been saved in a Network. By the term Energy we refer to the Energy provided by Electricity. In particular Saved Energy is an estimation of the quantity of Electrical Energy that has been saved by applying Energy Saving initiatives. A possible way to calculate it is to subtract the total amount of Energy consumed in one year, with the total amount of Energy that would be consumed if all the energy-consuming devices were never either switched off or disabled.

In order to relate the Saved Energy quantity with the time range against which it has been calculated, it is necessary to include a time unit in the unit of measurement used. The unit used by many electrical utility companies is the Watt per hour, and though the one defined by the International System of Units (SI), the Joule, does not include any time unit, it can still be translated into Watt: one Watt equals one Joule per second. Also ad-hoc instances of the *muo: Complex-Derived-Unit* class can be created, in which any unit can be specified so that time is considered.

In SPITFIRE, the Saved Energy quantity is modelled as an instance of the *spt: Saved-Energy* class. The relationship between a *spt: Saved-Energy* instance and the Sensor Network whose components'energy has been considered during its calculation, is established by the SPITFIRE predicate *spt: saved-Energy*; whose inverse is *saved-Energy-Of*.

The example in Listing 4.11 declares the resource *: SPITFIRE-Energy-Gain* as a *spt: Saved-Energy*, and its value is defined by using the predicate *spt: saved-Energy-Value*. This value is specified as the result of a calculation in which the energy values from the two different Sensor Networks *: SPITFIRE-Network1* and *: SPITFIRE-Network2* is considered. For such calculation an ad-hoc unit of measurement has been used and it is modelled as an instance of the *muo: Complex-Derived-Unit* class, as shown in Listing 4.11.

```
:SPITFIREEnergyGain a spt:SavedEnergy ;
 spt:savedEnergyOf :SPITFIRENetwork1 ;
 spt:savedEnergyOf :SPITFIRENetwork2 ;
 spt:savedEnergyValue "80.9" ;
 muo:measuredIn :WattOverMin.
:SPITFIRENetwork1 a spt:SensorNetwork ;
 spt:savedEnergy :SPITFIREEnergyGain.
:SPITFIRENetwork2 a spt:SensorNetwork ;
 spt:savedEnergy :SPITFIREEnergyGain.
:WattPerMin a muo:ComplexDerivedUnit ;
 muo:derivesFrom  ucum-unit:time/minute ;
 muo:derivesFrom  unit:W ;
 muo:prefSymbol "wom" ;
 muo:measuresQuality :SPITFIREEnergyGain .
```

Listing 4.11: Example of using the SPITFIRE vocabulary to describe the amount of Energy saved as a result of energy-saving policies applied in two different Sensor Networks. An ad-hoc unit to measure the Saved Energy is also defined.

The resource *: SPITFIRE-Energy-Gain* is intended as a proxy for any kind of Saved Energy estimations calculated on top of the SPITFIRE partners' sensor networks.

Context Module

We consider the definition of "context" given in the background Section 2.5, Chapter 2 as "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" [Abowd et al., 1999]. As a consequence, we model the situation and the activity occurring in the `Context Module` of our ontology (see Figure 4.1).

Definition: Activity Sensors react to stimulus coming from the occurrence of an event, i.e., something happening somewhere. Such occurrence can be of any kind and in any format, either in the digital or in the real world, e.g., vibration, sound, log-in, user-status. In the SSN ontology, such events are modelled as instances of the *ssn: Stimulus* class which triggered a record of it, that is the *ssn: Observation*. Although this rationale is valid and consistent, the same causality, participation, mereology features that characterize an event (i.e., an instance of *ssn: Stimulus*) and enrich the description of its surrounding environment, could usefully structure the record of this event (i.e., an instance of *ssn: Observation*). The structure of the real world event that has been sensed is not reflected in the structure of the records of such event.

In our ontology, an event is modelled as an instance of the *spt: Activity* class. Specific kinds of *ssn: Activity* have been defined from a map with the XEP-0108's User Activity categories[4], e.g., *spt: Coding*, *spt: Status*. Since a mood and a message are often associated with an activity, a *spt: Mood* class is also defined, to model moods, from a map with the XEP-0107's User Mood[5]. All the *spt: Mood* instances and the *spt: Activity* types, are defined in a sub-section of the main SPITFIRE ontology, whose namespace suggested abbreviation is *spt-c*.

It is suggested to multi-type a resource as an instance of both the *ssn: Activity* class and the *ssn: Observation* class. In this way all the Ontology Design Patterns defined in the Event Model-F ontology to structure events, can be applied to structure the records of these events, too.

[4] http://xmpp.org/extensions/xep-0108.html
[5] http://xmpp.org/extensions/xep-0107.html

The relationship between any agent and the *spt: Activity* instance that it triggers, is established by the SPITFIRE predicate *spt: trigger-Of*, whose inverse is the predicate *spt: trigger*.

The example in Listing 4.12 declares the resource *: SPITFIRE-Code-Phase1* as a *spt-c:Coding* associated with the *spt: Mood* instance *spt-c: Stressed* and triggered by the *dul: Person* instance *: Philip-Zimmermann* . This activity has been observed by the *spt-c: Logger* instance *: Logger1*,

```
:SPITFIRECodePhase1 a spt-c:Coding, ssn:Observation ;
 spt:trigger :PhilipZimmermann ;
 spt-c:mood spt-c:Stressed ;
 ssn:observedBy :Logger1 .
:PhilipZimmermann a dul:Person ;
 spt-c:triggerOf :SPITFIRECodePhase1 .
:Logger1 a spt-c:Logger .
spt-c:Logger rdfs:subClassOf ssn:Sensor .
```

Listing 4.12: Example of using the SPITFIRE vocabulary to describe activities that are sensed by sensors.

The resource *:SPITFIRECodePhase1* is intended as a proxy for any kind of either higher or lower level purpose that might have been sensed by a sensor and recorded. The best practise would be to associate the specified activity with other activities that enrich its surrounding environment, by using the Mereology, Participation, Correlation and Causality Design Patterns from the Event Model-F ontology. For instance a *spt: Sensor-Project* instance can be the cause of the Philip Zimmermann-s stress (causality pattern); or another activity triggered by a *spt: Sensor-Network* instance can be correlated (i.e., activities that occur at the same time but have no direct causality) with this coding activity.

Achievement by Examples

The following examples will demonstrate how the two issues listed at the beginning of this section i.e. application of Energy Saving measures and performance calculation of Energy Saving measures, have been solved by using SPARQL queries to access the datasets. The examples are based on the SPITFIRE consolidated use cases, whose core driver is Energy Consumption in a Building Automation scenario.

The first example demonstrates a solution for the first of the above listed issues, showing how to retrieve all the sensor nodes that can be disabled according to specific criteria, in order to save energy. This query involves all the main concepts from both the SPITFIRE ontology modules *Sensor-Network* and *Network-Component*. The second example indeed, demonstrates a solution for the second of the above listed issues, showing how to list the amount of energy saved in external projects focused on Building Automation and involving sensors. Afterwards an average of the resulting values can be calculated so that it can be used as a point of reference while estimating the performance of the Energy Saving policies applied. This query involves the usage of all the main concepts from both the SPITFIRE ontology modules *Sensor-Project* and *Energy*.

Example 1: Energy Saving measures A network administrator needs, in order to save energy, to disable all the sensor nodes that

- have been receiving less than 10 packets in the last five minutes. Disabling a sensor node allows to efficiently save energy only if this node is usually characterized by low activity;
- are characterized by a latency higher than 2 seconds. The nodes to be disabled are also filtered according to the quality of their link connections. Those who have a poorer link quality, are disabled first;
- are located in the Access Layer of the network or else in the Distribution Layer but while having network roles of either medium or low importance. In order to be disabled, sensor nodes must not be critical to the overall correct working of the network. When the Network Topology is Hierarchical as in our example, this happens mainly when the nodes belong to the Access Layer, but also in the Distribution Layer there might be some less critical nodes. This query selects in fact, any node belonging to the Access Layer and only the less critical ones belonging to the Distribution Layer.

This translates in the SPARQL query illustrated in Listing 4.13 that has to be run on a SPARQL engine.

```
1 SELECT ?node
2 WHERE{
3  ?node a ssn:Sensor ;
4  spt:link ?link ;
5  spt:belongsToLayer ?layer .
6  {?layer a spt:AccessLayer}
```

```
7  UNION
8  {?layer a spt:DistributionLayer .
9  ?node spt:networkRole ?role .
10 ?role spt:priorityLevel ?roleImportance .
11 FILTER(?roleImportance > 5)}
12 ?link spt:linkActivity ?activity ;
13 spt:linkQuality ?quality .
14 ?activity spt:linkActivityValue ?activityValue .
15 ?quality a :SPITFIRELatency ;
16 spt:linkQualityValue ?qualityValue ;
17 muo:measuredIn unit:s .
18 FILTER(?activityValue < 20) .
19 FILTER(?qualityValue > 2) .
20 }
21 ORDER BY ?activity ASC(?activityValue)
```

Listing 4.13: SPARQL query that selects all the sensor nodes that can be disabled to save energy. They are filtered according to the Network Topology Layer they belong to and their importance (role priority level), the quality of their links with respect to latency and their level of inactivity. The filters are such that only those nodes that have both the worst link quality, the lowest activity level and the less importance (either because belonging to the Access Layer or because belonging to the Distribution Layer but with minor roles) are selected.

This SPARQL query will return a list of the searched sensor node URIs, in either JSON, XML, RDF or HTML formats. An excerpt of the possible results in JSON format is shown in Listing 4.14.

```
1  {"head":{"vars":["node"]},
2  "results":{
3    "bindings": [
4    {"node":{"type":"uri", "value":"http://example.org/node/node6"}
5    },{"node":{"type":"uri", "value":"http://example.org/node/node7"}
6    },{"node":{"type":"uri","value":"http://example.org/node/node5"}
7    },{"node":{"type":"uri", "value":"http://example.org/node/node4"}
8    },{"node":{"type":"uri", "value":"http://example.org/node/node2"}
9    },{"node":{"type":"uri", "value":"http://example.org/node/node3"}
10   },{"node":{"type":"uri", "value":"http://example.org/node/node1"}
11   }]}}
```

Listing 4.14: JSON results to the SPARQL query in Listing 4.13. It consists of a list of the URIs of those sensor nodes that can be disabled, in order to save energy.

Example 2: Performance of Energy Saving measures A network administrator needs to assess the quality of the Energy Saving measures that have been applied during the last year in an automated building. He wishes to compare his own results with the state of the art i.e., with the amount of energy saved by others who

- have deployed sensors in an automated building,
- have applied energy saving policies in their sensor networks,
- have calculated the amount of energy saved,

so that an average of the amount of energy saved by others could constitute a reference while judging the value obtained by himself. Rather than loosing time searching for and reading literacy about building automation experiment results, he just runs the SPARQL query in Listing 4.15 on a SPARQL engine. This query selects the amounts of energy saved in different Sensor Networks across different deployments sharing the higher purpose of realizing a Building Automation environment. Such amounts will be listed in descending order since the highest ones are the most relevant for our example. In fact those who differ most from the network administrator's values would show a low performance of the current policies, for which the administrator is expected to find a workaround, as soon as possible. The SPARQL query in Listing 4.15 also selects the publisher of the data for the Sensor Network in which the energy has been saved, for future reference.

```
1 SELECT ?value ?owner
2  WHERE{
3    ?savedEnergy a spt:SavedEnergy ;
4  spt:savedEnergyValue ?value ;
5  spt:savedEnergyOf ?network .
6  ?network a spt:SensorNetwork ;
7  spt:partOfProjectTopic :BuildingAutomation ;
8  dcterms:publisher ?owner.
9 }
10 ORDER BY ?savedEnergy DESC(?value)
```

Listing 4.15: SPARQL query that selects the amounts of energy saved in different Sensor Networks (whose publisher is also displayed for future reference) deployed for Building Automation purposes. These values are rearranged in descending order.

This SPARQL query will return a list of values representing the amount of energy saved in other external automated buildings. An example of this results in an ASCII

table format, is illustrated in Listing 4.16). Alternative formats are either JSON, XML, CSV (Comma Separated Version) or TSV (Tab Separated Version).

```
+----------------------------------+
|      value   |  owner    |
|==================================|
|        12.5      |  john@email.com |
|        24.63     |  john@email.com |
|        .5        |  john@email.com |
|        12.5      |  john@email.com |
|        12.5      |  john@email.com |
+----------------------------------+
```

Listing 4.16: Results to the SPARQL query in Listing 4.15, serialized to an ASCII Table format. It consists of a list of values of energy saved by external sensor network deployments. They can be used to compare the amount of energy saved locally with others, in order to judge the performances of the energy-saving policies applied locally, against the state of the art.

Our ontology as defined at this point, enables descriptions of sensors, sensor data and sensor networks with a focus on energy saving and the environment surrounding their stimuli. Thus it supports future development of complex-event processing systems that optimize the energy consumption, as required by the SPITFIRE consolidated use case. This has been achieved by levering well-established, robus, already existing ontologies, i.e. the W3C SSN, Dolce+DnS Ultralite and the Event Model-F ontology. The SSN, Dolce+DnS Ultralite and the Event Model-F ontologies are described in the Appendix D, Appendix E and Appendix C, respectively.

4.2.3. Contextualised Sensor Ontology

We create Linked Data that depict the context surrounding the raw sensor observations. Each link that is created and each activity that is sensed can be commented on or rated. We enabled all this by aligning a selected set of existing ontologies with our own ontology for contextualised sensors, called the *SPITFIRE ontology* (source code available in Appendix F). Among the ontologies useful for our purpose, we have chosen ontologies that are stable and broadly used (see Figure 4.2). The set of existing ontologies include the Semantically Interlinked Online Communities (SIOC) vocabulary, the Association ontology (AO), the Friend-of-a-Friend vocabulary (FOAF), the Provenance ontology (PROV) and the Event Model-F ontology [Scherp et al., 2009a]. SIOC and AO support

the reviewing and rating activities description, together with the description of the network of relations that derive from such activities, e.g., groups of like-minded people. FOAF and PROV support the description of the objects and people involved into a context depiction, while the Event Model-f one enables the description of relations between different contexts.

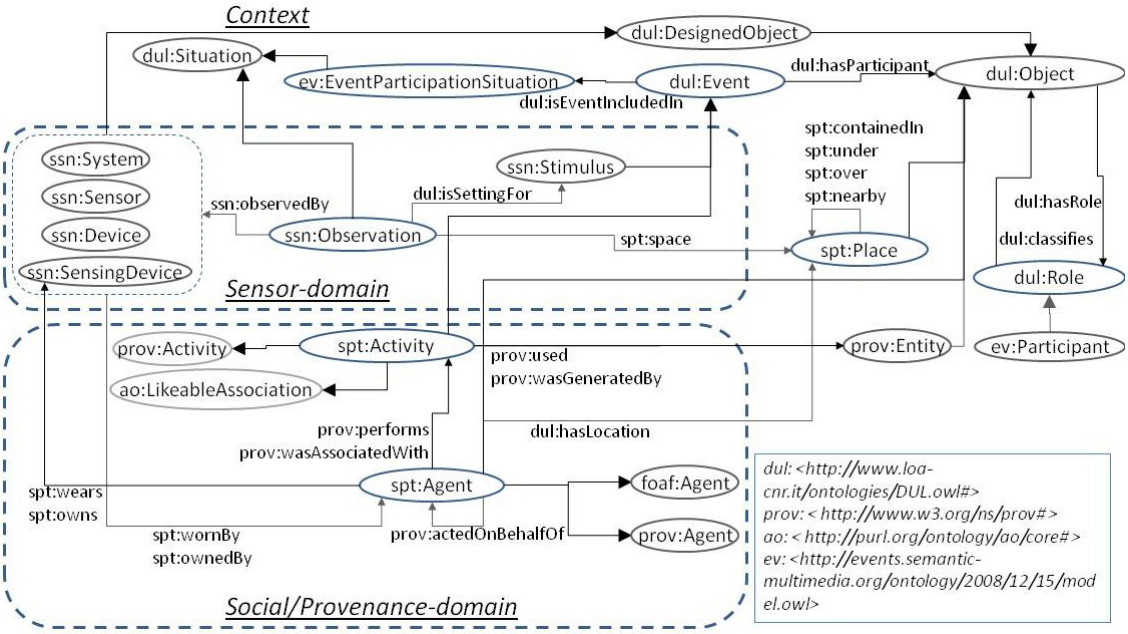


Figure 4.2.: Proposed SSN ontology extension for provenance, social and contextual information.

The rationale behind the aggregation of different contexts highly depends on the contextual information related to them. For instance, the overall situation depicting an increase in the distraction rate of an employee can be aggregated (linked) with another one, i.e., depicting two colleagues increasing the noise level in his proximity. The semantics of such aggregation is of causality: the noise level around the employee *causes* him to more easily distract. Such semantic, if not expressed, would hinder the reuse of the stored knowledge, since third-party applications would not be able to disambiguate the existence of such a link and would probably need to perform an analysis again to mine the same causality relation.

In our model, we enable the expression of complex relations between different contexts, i.e., causality, correlation and participation patterns, by aligning with the Event Model-F ontology [Scherp et al., 2009b] as depicted in Figure 4.2. We consider any *spt:Activity* as an association of different entities to an event, described by following the participation pattern. This activity can involve any object, agent or person (*dul:hasParticipant*,

prov:wasAssociatedWith); or might have been generated by them (*prov:wasGeneratedby*). Each of these entities participate in the context depiction with different roles (*dul:hasRole*) and location (*dul:hasLocation*). The sensed activity is linked with the sensing device both by its association with the *ssn:Stimulus* (i.e., the occurred event) that triggered the observation (Listing 4.17) and by the association between the sensing device and the involved entities that usually act as a platform which the devices are attached to (Listing 4.18).

```
ex:obs1 a ssn:Observation ;
 dul:isSettingFor ex:assoc1 .
```

Listing 4.17: Example of linking the sensed activity with the sensing device by its association with the stimulus.

```
ex:Myr a spt:Agent ;
 prov:actedOnBehalfOf ex:MyrFamily ;
 spt:owns ex:dev1 ;
 spt:owns ex:dev2 ;
 spt:wears ex:dev3 .
```

Listing 4.18: Example of linking the sensed activity with the sensing device by its association with the involved entities.

As an example of a context annotation process and user feedback description, a check-in at a particular location or event (e.g., a party) could be accompanied by all the available sensor data (represented using the Semantic Sensor Network ontology), a title or text description and additional information on the event type and mood of the reporter. Listing 4.19 shows the semantic annotation of different kinds of user feedbacks about a status update (a specific type of *spt:Activity*). The feedback information about the activity, the total number of votes, the number of positives votes, the comments and the people who, by submitting positive votes, have shown an agreement (thus, being identifiable as like-minded, e.g. *ex:friend1* as shown in Listing 4.20), are all tracked and included in the activity description.

```
ex:assoc1 a spt:Activity ;
 a spt:Status ;
 sim:weight 8.5 ;
 rev:reviewer ex:Myr ;
 rev:text "Unusual for this time of year..." ;
 rev:title "Great party weather!" ;
 rev:hasComment ex:comment1 ;
 rev:hasFeedback ex:feedback1 ;
 rev:hasFeedback ex:feedback2 ;
```

```
10  rev:totalVotes "12"^^xsd:integer ;
11  rev:positiveVotes "8"^^xsd:integer ;
12  prov:wasGeneratedBy ex:application1 ;
13  ao:mood spt-c:Happy ;
14  ao:occasion ex:party1 ;
15  rev:type ex:Text ;
16  ao:likeminded ex:friend1 .
```

Listing 4.19: Example of semantically annotated user feedbacks about a specific type of activity like a status update.

For each feedback, it is possible to specify votes, comments and the author (Listing 4.20).

```
1 ex:comment1 a rev:Comment ;
2     a sioc:Comment ;
3  rev:commenter ex:friend1 .
4 ex:feedback1 a rev:Feedback ;
5  rev:commenter ex:friend2 ;
6  rev:rating "-1" .
7 ex:feedback2 a rev:Feedback ;
8  rev:commenter ex:friend1 ;
9  rev:rating "1" .
```

Listing 4.20: Example of semantically annotated user feedbacks.

The author of the status update sets a textual title and message, together with a specific mood and occasion. The occasion might be even further described, as shown in Listing 4.21.

```
1 ex:party1 a dul:Event ;
2  ex:organizer ex:friend1 .
```

Listing 4.21: Example of semantically annotated occasion (a specific event) description.

Reviews for more than one activity can be grouped by levering on the *sioc:ReviewArea* concept provided by the SIOC ontology, as shown in Listing 4.22.

```
1 ex:area1 a sioct:ReviewArea ;
2  sioc:container_of ex:assoc1 ;
3  sioc:container_of ex:assoc2 .
4 ex:assoc2 a spt:Activity .
```

Listing 4.22: Example of grouping different semantically annotated reviews together.

The interpretation of sensor data is highly dependent on time. A sensor observation about the flooding of a water pipe for instance, is relevant only in the next couple of hours after it has been sensed. If considered after that it would be too late and the flood could not be stopped anymore. It is usually also necessary to keep an archive of past sensor observations. Therefore, in our ontology we support the grouping of sensor-related data to time under the concepts of *spt:SensorTemporalProperty*. An instance of *spt:SensorTemporalProperty* can be associated with a specific time range and any other property that is expected to change through time, e.g., location (in case of mobile sensors), feature of interest (if the sensor is moved from one object to another), owner, etc.

4.3. Implementation of LD4S

LD4Sensors (LD4S) is a RESTful Web server implemented using Java with the Jena library and the Jena Triple DB, levering on the SPITFIRE GUI. The API allows to access, store, update and delete specific resources that are typically involved in sensor measurements and sensor networks, after having semantically annotated them. Each functionality is better detailed in Section 4.3.1 and in Figure 4.4. Data can be accessed by querying a SPARQL endpoint, in addition to the REST API. The semantic annotation is inspired by best practices that follow the Linked Data principles and benefits from our ontology (called the *SPITFIRE ontology*). Also, a UML sequence diagram in Figure 4.3 shows the dynamic of some of the possible actions.

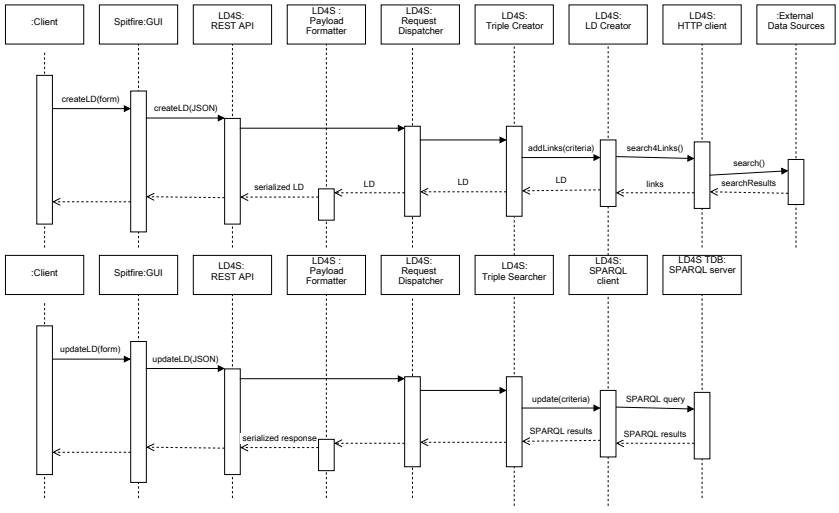


Figure 4.3.: LD4S Sequence diagram.

4.3.1. Use Cases

Use Case: Automate archiviation of auto-annotated sensor data. **Actor:** Muriel, Sensor Network Administrator. **Preconditions:** LD4S Web Service is up, running and remotely accessible. LD4S GUI is up, running and remotely accessible. **Postconditions:** Every metadata related to sensors of the networks is annotated as RDF, linked with external similar resources and stored in the LD4S triple store. **Basic Flow:**

- Muriel fills a form on the LD4S GUI with all the required information about her sennsors, via her browser. She then clicks the button "Save".

Extensions:

- Search for all sensors that are observing properties of the same concept "Entertaining" (e.g., sensors attached to television or radio, etc.). Muriel fills the Search form of the LD4S GUI from her browser, to create a SPARQL query that searches for this concept. This search is made possible by the Linked Data representation, despite the data stored via Muriel's script never specified the association with the "Entertaining" concept. The association was automatically created by LD4S.
- Feedback on the concept association. Muriel believes that one of the associations between sensors and the concept "Entertaining" is wrong. Muriel fills the Comment and Rate form of the LD4S GUI from her browser, to rate it negatively and explain why in the comment section.
- Add or Update one single sensor annotation. Muriel uses the LD4S GUI to load the metadata of the sensor odf interest in a form within her browser. Muriel changes the data. Muriel presses the button to Save the modifications.

Use Case: Automate archiviation of auto-annotated sensor data. **Actor:** Client Application or Semantic Web Expert. **Preconditions:** LD4S Web Service is up, running and remotely accessible. **Postconditions:** Every metadata related to sensors of the networks is annotated as RDF, linked with external similar resources and stored in the LD4S triple store. **Basic Flow:**

- The client collects all the sensor readings and sensor metadata; send such data to the LD4S API via HTTP requests with JSON payload compliant to the LD4S API specification.

Extensions:

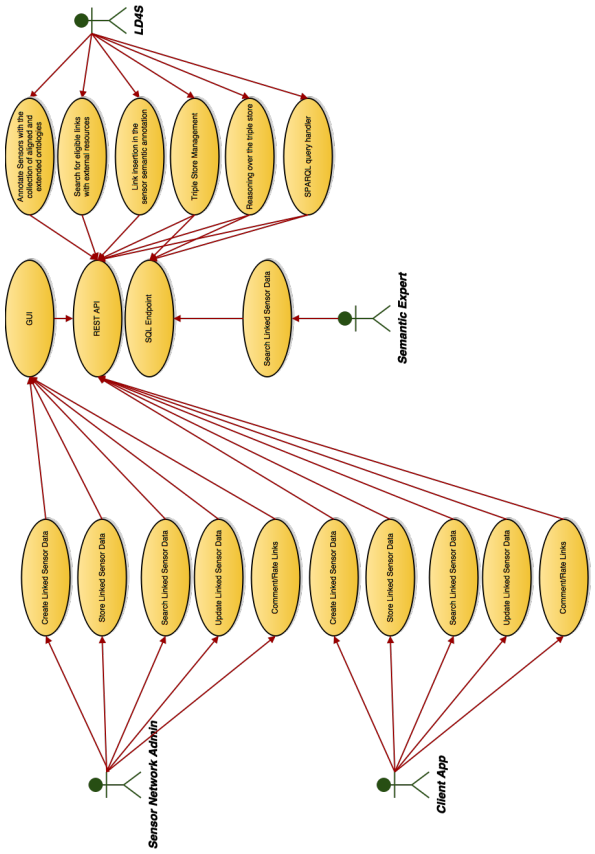


Figure 4.4.: LD4S Use Cases.

- Search for all sensors that are observing properties of the same concept "Entertaining" (e.g., sensors attached to television or radio, etc.). The client sends a SPARQL request as a payload to a HTTP request addressing the LD4S SPARQL endpoint. This search is made possible by the Linked Data representation, despite the data stored via Muriel's script never specified the association with the "Entertaining" concept. The association was automatically created by LD4S.
- Feedback on the concept association. The client sends comment and rate for the association of interest to the LD4S API as JSON payload to HTTP requests compliant with the LD4S API specification.

4.3.2. Architecture

The overall architecture, depicted in Figure 4.5, enables the storage, update, search, access and deletion of semantically annotated data about sensors, their measurements and their surrounding environment.

Clients can interact with the REST API either directly or through a user interface managed by the Web-based SPITFIRE GUI (Figure 4.5). The interface makes use of the Bridge Pattern. Clients can also access the data stored in the Triple Database by using the provided SPARQL endpoint. Content negotiation is performed in order to return RDF triples to the client, serialized in the preferred RDF serialization language. All the existing RDF serialisation languages are supported, i.e., turtle, n-triple, n3, rdf/xml. Figure 4.5 depicts the LD4Sensors components and their interactions. The components associated with different shades of the same color represent clients and server that are able to communicate with each other because of shared protocols. Each component is described in the following.

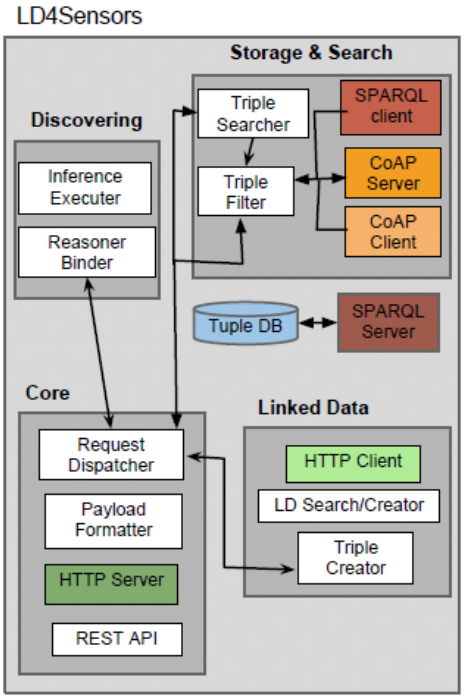


Figure 4.5.: Overall architecture of the LD4Sensors Web Service.

4.3.3. Core

The Core component, as in Figure 4.5 provides the base functionalities to communicate with Internet clients, applying a proper serialization to the response and distributing the incoming requests to the responsible component. The Core component includes:

- a RESTful HTTP server that constitutes the interface to the Internet to accept common HTTP requests according to RESTful principles;

- a Payload Formatter that formats the outgoing HTTP responses according to the Accept Header of the related HTTP request;
- a Request Dispatcher that forwards incoming HTTP requests to the responsible component.

HTTP PUT, POST, DELETE and GET requests are forwarded to respectively store, update, delete and access data. The data is submitted to the service as either HTML, JSON or Java serialized object payload and will be semantically annotated as Linked Data before being stored in a Triple Database (TDB). The TDB can be either the local one or a remote one, accessible through a SPARQL endpoint.

4.3.4. Linked Data

The Linked Data component in Figure 4.5 provides all the functionalities to create semantic (sensor and sensor-related) annotations using RDF, the SPITFIRE ontology, ontological inference and links with external resources. The Linked Data component includes:

- a Triple Creator that creates semantic annotations using RDF and the SPITFIRE ontology;
- a Linked Data (LD) Search / Creator that adds triples that link to external resources and can be rated and commented on. It also dispatches proper searches for external resources to link with the local ones;
- an HTTP Client that constitutes the network interface for outgoing traffic to LOD datasets and semantic search engines.

The semantic annotation is performed by the Triple Creator component and enriched by the Linked Data enrichment component by searching for external links. This search is run on the LOD resources indexed by Sindice [Oren et al., 2008]. Sindice collects Web Data in many ways, following existing web standards. It offers search and querying across this data which keeps updating live every few minutes. LD4S supports the creation of queries according to criteria specified by the user either in the payload in case of PUT or POST requests, or in the URI itself in case of GET request. These criteria consists of domain and/or context (time, space, thing) specifications which the retrieved resources to be linked are required to match. According to the specific criteria matched by each of the first retrieved resources, a different type of link (e.g., *spt:sameDomain*,

spt:sameThing, *spt:sameTime*, *spt:sameSpace*) is created with the resource of interest. In fact, we believe that for the links with external data to be useful, they must change dynamically according to the specific use case of interest. Consequently, we allow users to define custom criteria (Figure 4.6) to search for these links in the Web of Data, i.e.,

- domains of the datasets to be included in the search for links;[6]
- at which level the external resource should share the same context, (i.e., same thing of interest, time or space) with the local ones for a link to be created.

This linking system is an improvement of previous work and available online. The previous work relied on Silk as the automated framework to create links among related resources. However Silk had a very high response time thus was not scaling and required manual configuration to operate for each different scenario. For this reason, we then evolved LD4S in order to create links by directly querying Sindice with default settings only eventually customisable.

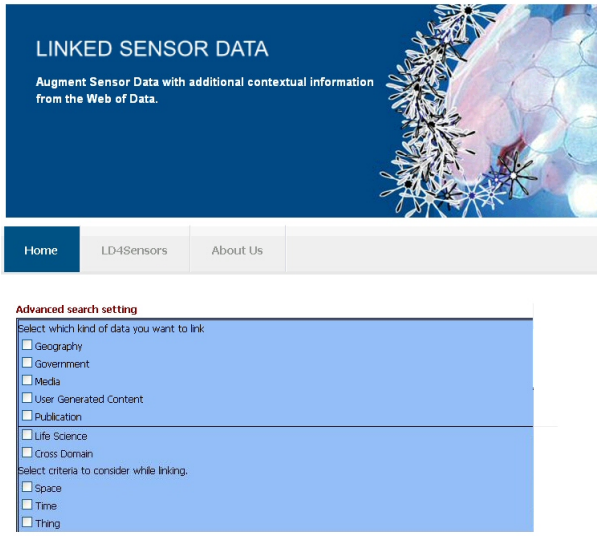


Figure 4.6.: Screenshot of the developed web service for semantically annotating and dynamically linking data from ubiquitous devices.

The *LD Search/Creator* component in Figure 4.5 is called by the *Triple Creator* one whenever the link criteria requested by the user are not yet stored in the TDB. The resources provided by LD4Sensors are abstracted from following the Abstract Factory Design Pattern.

[6] http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/DataSets/CKANmetainformation

The *Triple Creator* and *LD Search/Creator* component implementations follow the Command Pattern and communicate with the Domain Model that includes, in particular, *Link* and *LinkReview*.

4.3.5. Discovering

The *Discovering* component in Figure 4.5 provides all the functionalities to discover new facts, binding an inference model to a reasoner. In particular, it includes:

- a Reasoner Binder that, given an ontology and specific features of its predicates (e.g., transitivity), bounds it to a reasoner, i.e., an engine able to infer logical consequences from ontology-defined inference rules;
- an Inference Executer that uses the reasoner to create an inference model; it executes proper queries on it to discover new facts.

The *Storage & Search* component, depicted in Figure 4.5, provides all the functionalities to dispatch the storage of triples between triple store(s) exposing a SPARQL endpoint and sensor node(s) directly, according to the best practices. It includes:

- a Triple Filter that filters the RDF triples composing a semantic annotation so that they are dispatched to the correct external storage systems according to the resource availability of these systems and the best practices. Its implementation follows the Interceptor Design Pattern;
- a SPARQL client that constitutes a network interface for outgoing traffic to forward SPARQL queries to external SPARQL endpoints;
- a CoAP client that constitutes a network interface for outgoing traffic to forward CRUD[7] requests to tuple stores on sensor nodes.

4.3.6. GUI

LD4S also provides a GUI to facilitate the creation (Figure 4.7), editing (Figure 4.8), search (Figure 4.9) and ratings (Figure 4.10) of linked sensor data even further.

[7]Basic operations of a relational database: Create, Read, Update, Delete (CRUD).

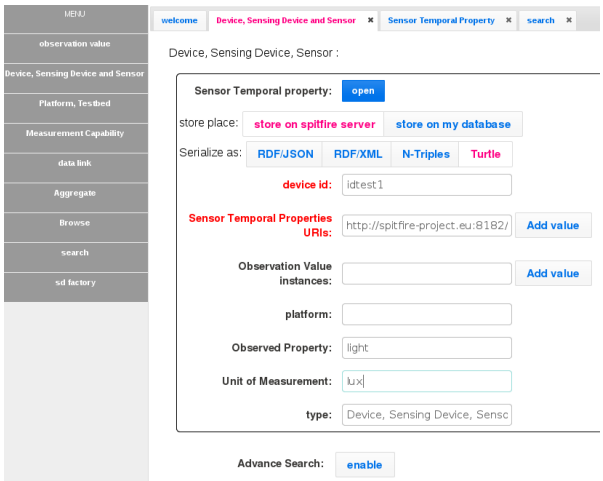


Figure 4.7.: GUI to create one of the resources exposed by the API: the Sensor Device's temporal properties.

The GUI is a web application (tested so far on Mozilla Firefox and Chrome) that requires authentication. Once logged in, the user is presented with a menu bar on which each item corresponds to one of the resources exposed by the LD4S RESTful API.

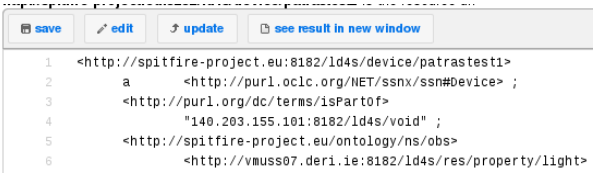


Figure 4.8.: GUI to view, edit and save a linked sensor data annotation.

These items are followed by a menu item for Search. The Search panel reduces the creation of a Sparql query to a form filling activity as shown in Figure 4.9.

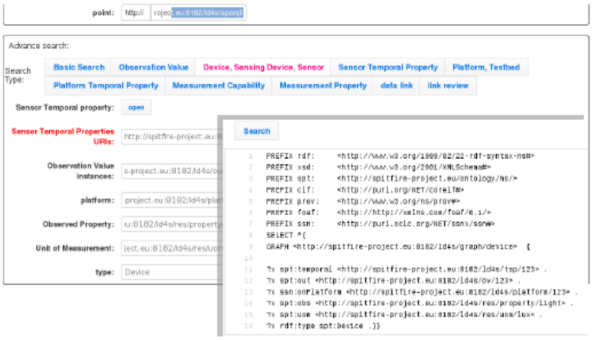


Figure 4.9.: GUI to create SPARQL query for searching among the stored linked sensor data annotations.

When clicking on any of these items, the interface provided is always split in two sections. The top section, as in Figure 4.7, contains a form to fill with required information to either trigger the semantic annotation or search for a specific resource. The bottom section, as in Figure 4.8, displays always a serialised version of the resource that has just been annotated, created, updated or somehow loaded. Within the Search menu item, the bottom section displays the search results while the top section contains a form to fill in order to trigger the automated creation of a SPARQL query.

Figure 4.10.: GUI to rate and comment a specific link between local and external resources.

Every time the bottom section displays a serialised linked sensor data, the serialisation is accompanied by buttons to share it on the most common social networks and add a comment or a rating to each link, as in Figure 4.10.

The GUI is open source and currently available online[8].

4.4. Evaluation

We evaluated LD4Sensors from the perspectives of usability, utility, uptake and performances.Our overall goal is to (a) quantify the actual gain of using LD4S in a building automation scenario while (b) also evaluating the quality of the implementation.

[8] https://code.google.com/p/ld4s-gui/

4.4.1. Setup

We deployed the use case described in Section 4.2.1 as follows. We run the LD4S sensors on commodity hardware equipped with an Intel Core 2 Duo processor and 305 GB of disk space. We opened the port on which D4S was listening so that it could be accessed from outside networks. LD4S logging capability was set to enable the verbose mode.

We deployed three iSense[9] sensor platforms that included temperature, light and pressure sensors. We placed each of them into the three meeting rooms available in the Insight building (i.e., our reserach institute), at the first floor. We used the iSense IDE to implement and install a C program, directly on the platforms, which forwarded any collected sensor observation to the LD4S server. iSense nodes are equipped with 6LoWPAN and CoAP, thanks to which they were able to send packets over IPv4 and HTTP to reach LD4S.

We implemented an Android Java application which forwarded sensor observations only from the GPS sensor of Android smartphones, to the LD4S server.

We selected 38 users among the staff members of Insight, see Figure 4.11. Researchers at Insight have a background knowledge that varies widely from Natural Language Processing, to Biology to Sensor Networks and to Semantic Web. Not all of the researchers have a Computer Engineering background. The users were selected randomly but while paying attention that

- they all owned an Android smartphone, so that we would be able to monitor their entrance to the first floor printer room via our Android application;
- their desk was located on the first floor, so that they would normally be using the first floor printer;
- 1% of them were actually working on sensor observations at that time, thus had sensor network experience to some degree. This is meant to support the evaluation of our usability heuristics, as will be explained in Section 4.4.2. It is also leading us to a more realistic measure of the uptake, as explained in Section 4.4.2.

We asked all the users to

- install our application on their smartphones;
- use LD4S at least once over a period of 30 days;

[9] http://www.coalesenses.com/index.php/products/

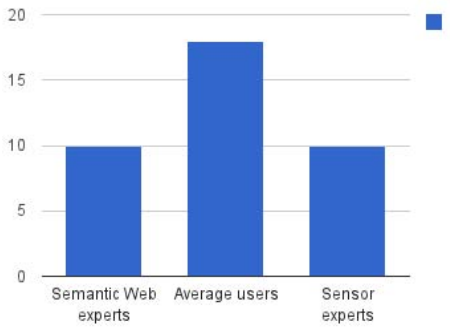


Figure 4.11.: Characterization of the users that were involved in our evaluation.

- fill a usability survey at the end of the 30 days.

The users received instructions at the beginning of the 30 days, on the minimum set of actions to perform when using LD4S. After having performed this initial set of pre-defined actions, they were left free to use LD4S to eventually perform any other action in any other way (GUI or API) and as often as they wished. The instructions included a Glossary of the terms used and their meaning, links to the API specification and GUI, and the following pre-defined set of actions to perform:

- use the GUI to semantically annotate a sensor (among the ones deployed in the meeting rooms);
- use the GUI to semantically annotate a platform (among the ones deployed in the meeting rooms);
- search for the platform that has just been created;
- comment and rate the links that LD4S included in the loaded semantic annotation.

4.4.2. Evaluation Steps and Results

While the users installed the Android application and started following the instructions given, we logged the IP address, URL and payload of each request received by the LD4S server, along with the timestamp (to calculate how long the same user kept accessing the system). This was meant to support the Utility and Uptake evaluation, as described in Section 4.4.2 and Section 4.4.2.

Energy Consumption

At the end of the 30 days, we searched the LD4S triple store for GPS sensor observations originated by Android smartphones and whose latitude and longitude values were ranging between the coordinates of the first floor printer room. We considered every set of observations from the same smartphones within those coordinates and a time range of 10 minutes, as one single access by one single person to the printer. For each single access we associated 5 paper sheets printed as black and white, as we considered this to be the average consumption rate in normal conditions. In order to make up for the imprecision of the GPS coordinates, we only considered GPS observations that ranged within the scope of the printer room ones, for a period of at least 1 minute.

Within the Insight building, there are multiple printers at the second floor, one at the ground floor and one at the first floor. Employees tend to use mostly the printers located on the same floor as where they are seated. All the users selected for this experiment were seated on the first floor, therefore were mostly using the same single printer on the first floor.

Then, we compared the amount of sheets and energy consumption that we caluclated to the amounts that had been logged by the printer server, over the 30 days period. Our results matched the activities logged by the printer's server, with a high accuracy (80%), thus revealing our approach to be successful. The following factors affected this result:

- some users do not carry their smartphones in their pockets all the time. They may enter the printer room while not carrying any smartphone;
- the amount of sheets printed may vary;
- users may enter the printer room for reasons other than printing;
- users not involved in the experiment, either seated on the first floor or on different floors, may have used the first floor printer.

Therefore, the result could be improved by broadening the sensors involved, thus further exploiting theadvantages of the Linked Data principles. For instance, data from the PIR[10] sensors placed on the ceiling along the corridor to reach the printing room could be collected and employed to improve the estimation of the printer usage.

[10] A Passive InfraRed (PIR) sensor is an electronic sensor that measures infrared light radiating from objects in its field of view. They are most often used in PIR-based motion detectors.

Usability

A good usability for average users is a major challenge when developing applications to handle sensor data or semantic annotations. Since LD4Sensors aims at handling both, usability surely is a big challenge to address. We evaluated the usability according to the following heuristics:

- Match between system and the real world. The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
- Consistency and standards. Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
- Recognition rather than recall. Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another.
- Help and documentation. Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

In order to ensure that the terms used in the LD4S API and GUI were of immediate understanding (i.e., matching the real world and easy to recognise rather than recall), we relied on a demographics that included a majority of non-sensor-experts. The feedback from non-experts in the field would show with higher certainty whether the terms used are of immediate understanding and reflecting common knowledge, since this is all this type of users could rely on, given their lack of any further background knowledge. The demographics also included a minority of sensor-experts in order to ensure the adherence to standards, thus satisfying one of the above heuristics. Finally, all the users were asked to evaluate the API specification and their experience using the GUI, thus allowing us to evaluate the last of the above heuristics.

We aim at making the HTTP request payload creation as simple and intuitive as possible by fully adhering to REST principles and by providing a clear user interface and comprehensive documentations. On the semantic technology side, we provide a graphical

support to help the user formulating SPARQL queries, annotating sensor metadata and linking the metadata with external resources.

The results, as depicted in Figure 4.12, show that the majority of user did not use the API. Those who did found it difficult to use the API properly, and they suggested the documentation to be improved.

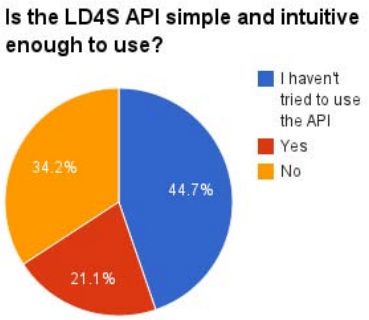


Figure 4.12.: Pie chart showing the proportion between the answers submitted to the survey question about the usability of the LD4S API.

The API could be used via curl or the creation of ad-hoc scripts. Sample curl requests are depicted in Listing 4.23.

```
1 curl --request post --data '{"observed_property":["light"]}, "uri":[
    "http-or-coap://something/where/to/get/this/description/from"]}'
    --header "Accept: application/x-turtle" --header "Content-type:
    application/json" http://spitfire-project.eu:8182/ld4s/device/
2
3
4 curl --request post --data '{"uom":["lux"], "sensor_id":["http-or-
    coap://something/where/to/get/the/node/description/from"], "uri"
    :["http-or-coap://something/where/to/get/this/description/from"],
     "foi":["tunnel"], "location-coords":["38.24444_21.73444"], "
    location-name":["Patras"], "author":[{"surname":["Boldt"],"
    firstname":["Dennis"], "email":["boldt@itm.uni-luebeck.de"]}], "
    start_range":["13-09-17T19:03Z"], "end_range":["14-09-17T20:03Z"
    ]}' --header "Accept: application/x-turtle" --header "Content-
    type: application/json" http://spitfire-project.eu:8182/ld4s/tps/
5
6
7 curl --request post --data '{"uri":["coap://something/where/to/get/
    this/description/from"], "sensor_id":["http-or-coap://something/
```

```
      where/to/get/the/node/description/from"], "values":[["12.4","21.9
      ","88.7","24.5"]],"start_range":["13-09-17T19:03Z"], "end_range"
      :["13-09-17T20:03Z"]}' --header "Accept: application/x-turtle" --
      header "Content-type: application/json" http://spitfire-project.
      eu:8182/ld4s/ov/
```

Listing 4.23: Example of using the LD4S API to create static and time-varying (spt:SensorTemporalProperty) features of a sensor node, followed by the creation of a sensor reading.

On the other hand, all the users used the LD4S GUI and found it of immediate undersatnding and easy to recognise its functionalities rather than recalling them. The semantic web experts and the sensor experts confirmed the adherence to standard terms from the sensor and the semantic web worlds. However, as Figure 4.13 depicts, non-semantic-web-experts did not understand the overall motivation behind semantically annotating sensor and sensor-related data, thus why using the system in the first place. This suggests us to improve the way we advertise the advantages of our system, perhaps including videos of relevant user stories.

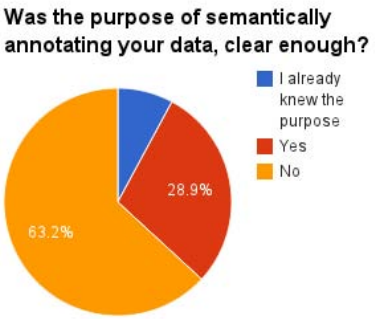


Figure 4.13.: Pie chart showing the proportion between the answers submitted to the survey question about the clarity of the motivation behind semantic annotations for sensors.

We can then summarise the results as follows:

- Match between system and the real world. Passed.
- Consistency and standards. Passed.
- Recognition rather than recall. Passed.
- Help and documentation. Failed.

Utility

Our interest focused on analysing which types of API resources were annotated the most. The amount, type and uniqueness of the Internet Connected Objects (ICOs) that are semantically annotated are the parameters defined for measuring the utility of our LD4Sensors web service.

The results show that the most annotated resources were the sensor data and sensor metadata, while the service got almost no request for semantically annotating sensors' contextual data (see Figure 4.14). The result could be explained by the lack of understanding of the purpose of semantic annotation, as indicated by Figure 4.13, due to the quality of the help and documentation provided. This may have led to a lack of interest for annotating data from domains apparently unrelated to the sensor domain, as the externally linked ones are.

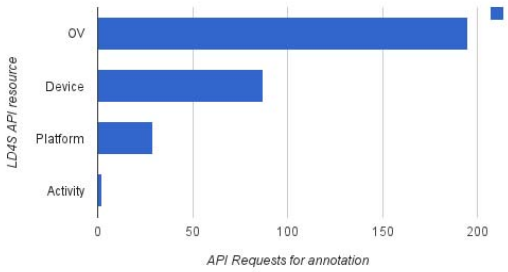


Figure 4.14.: Bar chart showing the type of LD4Sensors API resources whose semantic annotation was requested the most, by the users. OV stands for Observation Value, i.e., sensor reading.

Finally, we investigated whether the users considered the links with external resources that LD4Sensors created during the Linked Data generation, as useful. The results were encouraging with regard to the relatedness of the subject between the resource of interest and the external ones, as shown in Figure 4.15.

However, it seems that the external links, though related with the subject resource, are useless for the users' purposes most of the time. Figure 4.16 depicts the high percentage of people who did not consider the externally linked data useful for their own purposes.

External links tend to be referencing concept definitions from Wikipedia and further information about geographical locations, as in Figure 4.17. However, users were probably not motivated enough to further investigate on the content of these external links.

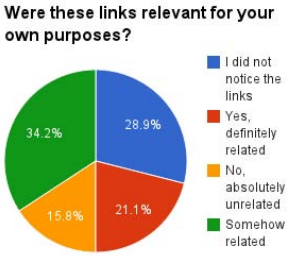


Figure 4.15.: Pie chart showing the percentage of users who found the resource of interest sharing the same subject of the ones it had been linked to.

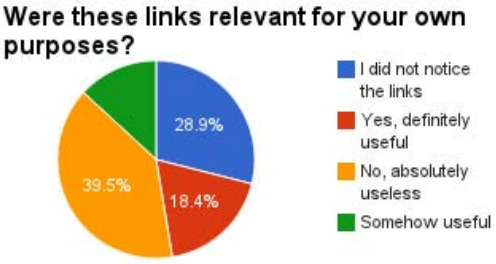


Figure 4.16.: Pie chart showing the percentage of users who found useful the links with external resources, with respect to their personal purposes.

```
from : http://spitfie-project.eu:8182/ld4s/device/idtest1
title : http://geo.linkeddata.es/resource/wgs84/40.867331836153966_-2.1129880070419715
http://geo.linkeddata.es/resource/wgs84/40.86736577151648_-6.557479747029053
to : http://geo.linkeddata.es/resource/wgs84/40.86736577151648_-6.557479747029053
http://geo.linkeddata.es/resource/wgs84/40.867331836153966_-2.1129880070419715
temporal : 2011/03/03 2010/11/12
```

Figure 4.17.: Semantic annotation extract, highlighting the creation of a link between a local resource and an external resource. The external resource here, includes further information on a geographical location.

Uptake

The uptake directly depends on both usability and utility. It is measured in this experiment, in terms of frequency of accesses to our service by different users. During the 30 days in which the users have been using LD4Sensors, we monitored each access.

Given that 1% of the users at the time of the evaluation were working with semantically annotated sensors for their reserch tasks, we checked whether they ended up making use of LD4S for their every day activities. For this purpose, we compared how many unique accesses with the total amount of accesses received over the 30 days period. If

the two values did not match, it would mean that the same person accessed the service multiple times, rather than just the minimum one-time access enforced by our experiment instructions requirements. This would of course mean that the uptake was positive and users actually grow interested and used the system for their own purposes, successfully.

Unfortunately, the amount of unique accesses did not differ much from the total amount of accesses received during the evaluation period. Figure 4.18 shows the total amount of visits per day, to the LD4S server, over the 30 days time period between the 27th of January 2013 and the 26th of February 2013. These visits sum up to a total of 43 visits, which does not differ much from the total amount of 38 users involved in the experiment, i.e., does not differ much from the total amount of unique visitors. Therefore, the uptake was not significant. This result may also be related to the weakness of the help and documentation highlighted by the usability evaluation.

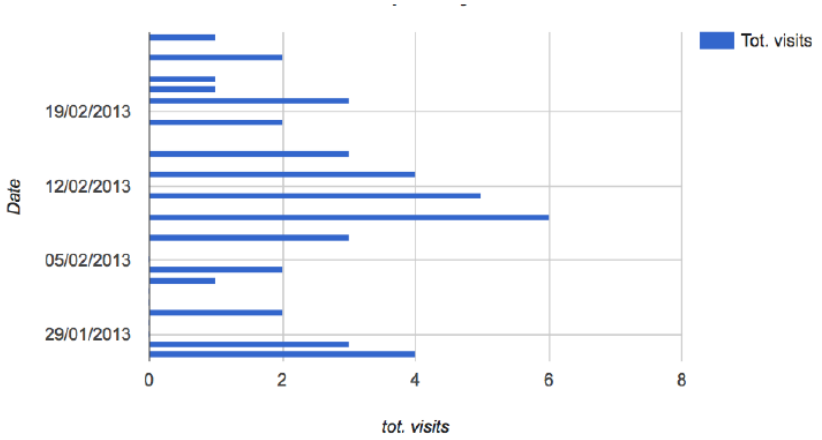


Figure 4.18.: Bar chart visualising the amount of visits per day to the LD4S server over the 30 days time period.

We counted how many times it was required to access the local triple store. The creation, updating, loading and searching for semantically annotated resources, required at least one access to the triple store. We investigated the correlation between access frequency, unique accesses and the volume of accessed data. The result, as displayed in Figure 4.19, shows the frequency of accesses and the amount of unique accesses equal most of the time, for the same reason as explained for Figure 4.18. Meanwhile, the volume of accessed data doubles, on average, the frequency of use. The reason why this proportion is quite constant, is that users sticked to the instructions given and only performed the minimum set of actions required. This again, demonstrates how the uptake was low and the participants in the evaluation did not use LD4S for their own purposes any further.

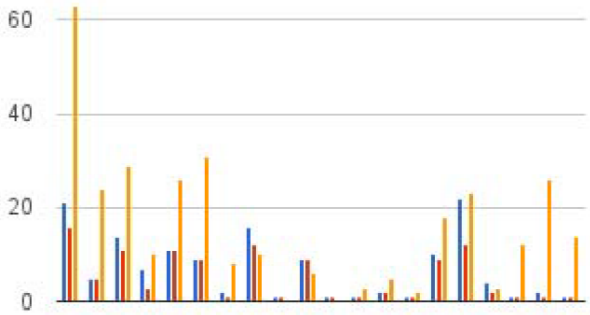


Figure 4.19.: Column chart showing the access frequency to the service (blue column), the amount of unique users performing the access (red column) and the amount of data accessed in the local triple store (orange column).

However, since this was a pilot evaluation, the uptake was satisfying. It will be necessary to repeat the evaluation over a longer time period and after having improved the usability in terms of help and documentation.

Performance

Finally, we investigated the performance of the LD4S web service. This is a primitive benchmark since it does not take into account any business logic information. The performance evaluation makes use of the LD4S verbose logs that we configured in the setup (see Section 4.4.1). In particular, here we make use of the logged requests timestamps and both input and output payload size.

We measured the throughput as ratio between the amount of requests and the seconds required to produce an answer. In Figure 4.20 we compare the throughput with the size of the payload. The size of the payload considered and displayed, is the average of both the payload received as input by LD4S and the one returned as output. Figure 4.20 depicts a decrease of the throughput as both input and output payload sizes increase. This decrease in throughput is, however, reasonable and not exponential. In the future, a cache implementation could improve such performance.

Conclusions

We demonstrated how LD4S can bring advantages into a building automation scenario for monitoring energy consumption by simply installing an Android application (rather than requiring complex sensor deployments). Also we mentioned how the nature of the linked sensor data produced can facilitate the improvement of the performance in

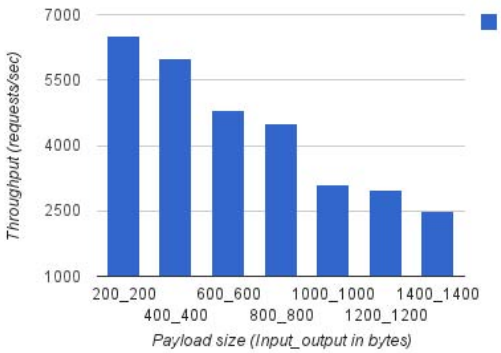


Figure 4.20.: LD4Sensors web service performances.

this scenario. The usability heuristics were all satisfied with the exception of the help and documentation quality, which indeed, needs to be improved. The utility evaluation highlighted a necessary further explanation on the added value of the external links created. The performance evaluation was promising while could be improved by the implementation of a caching system. The uptake was not particularly satisfied as all the participants only used the system as much as they had been required to. In particular those users who could have made use of LD4S for their every day tasks since these related to sensors, chose not to do so. However, the uptake was enough for a pilot evaluation and is expected to increase as we remediate the usability issues.

4.5. Related Work

To the best of our knowledge, no existing work has

- applied user feedback to sensor data to improve their quality and reliability,
- semantically enhanced the typically entity-centric context model while enabling seamless context merging, or
- used ontologies for network components to support the network dynamism and energy efficiency.

This is missing since the lack of data quality hinders data reusability and trust, and there is a need for energy efficiency and a simplified merging of different contexts. The context identification is performed by activity recognition algorithms which have been investigated thoroughly [Chellappa et al., 2008].

The importance of single local contexts to depict the overall situation has been demonstrated [Gordon et al., 2011] and research tends to focus on Adaptive On-Line Learning to recognize contexts in real-world domains as a function that needs to remain flexible since a new context may be continuously added to the system and old contexts may no longer be perceived [Korel and Koo, 2010]. This motivates us to focus on local context aggregations and on the support of dynamism.

In the context aggregation process, heterogeneity is the main issue. It is caused by the XML-based data representation which requires ad-hoc and expensive mappings to enable any merging. For instance, the OGC, after recognizing this limitation of their XML-based standards [OGC - Open Geospatial Consortium, 2010], is currently working on an integration [Br¨oring et al., 2011] with the W3C SSN ontology [Compton et al., 2012a]. A further step in using simple semantic annotations is to follow the Linked Data principles [Berners-Lee, 2006a]. To the best of our knowledge, this has not yet been done for dynamic sensor datasets, despite the usefulness of the external links found for sensor data [Leggieri et al., 2011a].

Context modeling is relevant and also enables the coordination of more reliable transmissions to be achieved in minimal time [Chong et al., 2009]. Although it has been argued [Ramparany et al., 2011] against the common habit of defining context as attributes of entities in favour of a more comprehensive representation where semantic relations and structural information are included, the usual context model for sensor data is the entity-centric one. It is defined [Villalonga et al., 2010b] as *any information that can be used to characterize the situation of an entity*; several ontologies (not yet published) have been applied to it [F.Paganelli and Giuli, 2011, Dey and Abowd, 2000, Ramparany et al., 2011] and no alternative have been proposed yet.

User feedback and sharing have already been successfully applied to improve data quality, but never to sensor data in ubiquitous systems. For instance, crowdsourcing - multiple small contributions from various individuals towards a larger body of work - is a currently broad phenomenon [Howe, 2006] , with Wikipedia [Lih, 2009] the most well-known example. Citizen reporting refers to social media updates from many contributors on a particular event of interest, and citizen sensing [Boulos et al., 2011] systems, e.g., Twitris [Jadhav et al., 2010], have taken advantage of these citizen reports to some effect. Although Sensorpedia[11] and Cosm[12] are platforms to share sensor observations, the stored

[11] http://www.sensorpedia.com/
[12] http://cosm.com/

data are hard to disambiguate and to reuse because of their lack of semantic annotation, and are often missing information relevant for their reuse, e.g. unit of measurement.

4.6. Conclusion

We have presented a set of innovative research areas including:

- modelling of a context-aware ontology that for the first time correlates network topology concepts with contextual information, energy saving and user feedback (on sensor data and links);
- meaningful link typing that explicitly justifies the creation of the link itself; differently from the generic meaning of the mostly used link type in the LOD cloud, i.e., *rdfs:seeAlso*.

The achievements above address the Research Question **Q 1. Context**: *How can contextual information be used to enrich sensor data?* We then delivered further contributions aimed at

- facilitating the adoption of the data model and best practises we defined, along with the Linked Data principles;
- link customisation
- supporting the discovery, querying, visualisation and sharing of the Linked Sensor Data

.

The above achievements address the Research Question **Q 2. Communication**: *How can sensors communicate across different platforms without ad-hoc solutions?*

We demonstrated the validity of our approaches using a real-world application scenario derived from real users' needs. This resulted in increased efficiency thanks to our community sensing approach, broader areas can be monitored by the same amount of sensors. The web service performances, uptake and utility are promising.

The user feedback still indicates that some improvements are necessary in terms of usability and explanation of our intentions. The results in Figure 4.12 about the LD4S API usage lead us to plan the inclusion of additional examples for each possible API

request in the online guide, along with its outcome. Also, Figure 4.16 motivates us to plan an advanced analysis of the data content to filter out those links that are most likely not to be considered useful. In the future, we plan to make our service portable on any kind of (especially mobile) device, so that it will be easier to automatize a more pervasive data collection system and its direct semantic annotation.

Chapter 5.

Predicting Sensor Relevancy for ADLs Logging

Based on "Distributional Semantics and Unsupervised Clustering for Sensor Relevancy Prediction" [Leggieri et al., 2015a]

In this chapter we address the Research Question **Q 3. Relevancy**: *How to identify which sensors are more relevant sources of information to define a specific small context scope - the Little Data - of interest?* We use the foundations built in the previous Chapter 4 - such as the ontology and LD4S - in an innovative sensor relevancy prediction based on clustering and distributional semantics. Our system predicts how relevant is a given sensor with respect to the task at hand; where task is an activity that is currently being sensed.

The algorithm is independent from the amount of devices that are queried or deployed. As a consequence, it is a perfect candidate to support the scalability of the IoT vision where the amount of ICOs is predicted to grow thus increasing the challenges for data querying. A pre-selection of the most relevant sensors using our method, would dramatically speed up the query process despite the broad base of devices potential target of the query engine.

A typical application of our approach is the logging of Activities of Daily Living (ADLs) which is becoming increasingly popular thanks to cheap wearable devices. Currently, most sensors used for ADLs logging are queried and filtered mainly by location and time. However, in an Internet of Things and Pervasive Computing future, a query will return a large amount of sensor data. Therefore, existing approaches will not be feasible because of resource constraints and performance issues. More fine-grained queries will become

necessary. We propose to filter on the likelihood that a sensor is relevant for the currently sensed activity. Our aim is to improve system efficiency by reducing the amount of data to query, store and process (while at the same time saving energy) by identifying which sensors are relevant for different activities during the ADLs logging by relying on Distributional Semantics over public text corpora and unsupervised hierarchical clustering. We have evaluated our system over a public dataset for activity recognition and compared our clusters of sensors with the sensors involved in the logging of manually-annotated activities. Our results show an average precision of 89% and an overall accuracy of 69%, thus outperforming the state of the art by 5% and 32% respectively. To support the uptake of our approach and to allow replication of our experiments, a Web service has been developed and open sourced.

The logging of Activities of Daily Living (ADLs) is the process of tracking personal data generated by our own behavioural activities. New wearable devices - such as Fitbit[1], GoPro[2] and Google Glass[3] - and the ubiquity of sensors - such as in our smartphones and vehicles - are making the ADLs logging a reality. Since most of these devices are Internet-enabled, the ADLs logging process falls into the larger Internet of Things phenomenon, where the amount of Internet-enabled devices is growing quickly and is purported to reach 50 to 100 billion devices by 2020. This rises to 100,000 billion if we consider not only machine-to-machine (M2M) communications but communications among all kinds of objects [Sundmaeker et al., 2010b].

ADL logging is all about identifying and recording activities. However, the identification and characterisation of activities constitutes one of the main challenges for ADLs logging. Applications vary from critical situations - such as patient monitoring in healthcare - to recreational - such as live blogging. The amount of activities recorded is usually high; they are performed in idiosyncratic ways, may differ a lot from each other and involve sensing in real-world environments. This makes it difficult to predefine which variables to record during each different activity. In fact, given the unpredictability of daily life activities, it is not possible to select which ICOs (Internet-Connected-Objects) will become of interest through time. For instance, while being out for a walk the user may change plans and drive the car to a remote place. In this case, the system should recognise the necessity to change from sensing breath, air pollution, noise, temperature to sensing traffic, fog/humidity etc.

[1] http://www.fitbit.com/
[2] http://gopro.com/
[3] https://www.google.com/glass/start/

Currently, the sensors that are available for ADLs logging are mainly queried and filtered for a specific time range and location. However, given the predicted rise in such Internet-enabled embedded devices, such a query will eventually become unmanageable. The query could not be executed because of performance issues. A more fine-grained solution is required to determine which sensors can provide relevant information for logging the activity at hand, i.e., what variables to record during each different activity. In the example above, such variables would be `breath`, `noise`, `air-pollution`.

5.1. Previous Solutions

Previous research has focused on identifying activities over a dataset of collected sensor readings. In fact, given the variations, learning from sensor readings is the only way to obtain activity models. Supervised classification based on such models or unsupervised clustering is applied over the dataset, classifying the sensor readings with activity labels. Particular focus is given to learning techniques to reduce the amount of activity-labelling required. Unsupervised clustering also requires a subsequent labelling of the clusters both to classify the activity and to ground the sensor system (e.g., to identify particular objects under various environmental conditions). Labelled activities constitute a problem because of the manual effort required by end users to assign labels and because of the high number of Activities of Daily Living (ADLs).

Philipose et al. [Philipose et al., 2004] had previously demonstrated that it is possible to discriminate between many activities by taking as features the objects used, placing sensors on the objects themselves, thus modelling activities as sequences of object use. [Wyatt et al., 2005] also observed that the structure of these models strongly corresponds to natural language instructions (e.g., recipes) available for many activities. Wyatt et al. [Wyatt et al., 2005] levered the fact that, although daily activities are varied and idiosyncratic, they have common features that most people recognize, i.e., they have a generic *common sense* aspect that often suffices to recognize them. Wyatt et al. built on such observations to consider activities as sequences of object use and to model them by analysing the co-occurrence of object terms in websites (returned by web searches) in order to assemble a Hidden Markov Model [Stratonovich, 1960]. The goal is activity inference. It required no human input other than the natural language names of activities and of object tags.

5.2. Proposed Strategy

Our approach relies on the same parallelism between activities, common sense and textual content available online as in [Wyatt et al., 2005]. However, it differs because: **1.** we apply hierarchical clustering to achieve a different goal of predicting relevant sensors for the current activity; **2.** we reuse a well-known distributional semantics algorithm to analyse our corpora. This makes the text analysis more robust (rather than implementing the steps for the co-occurrence analysis from scratch). It also allows our system to be suitable for resource-constrained devices (since the analysis can be run offline) and for potentially being performed at runtime; **3.** we do not require natural-language names of activities as input; **4.** we can potentially infer the activity labels as centroids of the final clusters or place them among parents in upper-level ontologies; **5.** we achieve higher precision (89.5% versus 70% in previous works) and accuracy (52% versus 69%).

We predict which sensors are more likely to provide relevant information for the activity that is being currently sensed, among those placed in the same location and time. Such prediction is based on the degree of semantic relatedness between the objects that the sensors are monitoring. This translates into relatedness between the sensors themselves. Clusters are then created based upon the measured relatedness between the sensors and are interpreted as distinct activities.

Our prediction requires the availability of: **1.** one or more sensors that have recently sensed a change in status (e.g., light switched on after it had been switched off) in a specific location; **2.** sensor metadata which must include the sensor's *observed property* and *feature of interest*. The sensor's *observed property* is the property that it is designed to sense; while its *feature of interest* is the object which the observed property belongs to. For example, if a sensor measures the temperature of a microwave, the temperature is the observed property and the microwave is the feature of interest.

To the best of our knowledge, the approach proposed in this paper is the first of its kind. Previous attempts involved either only unsupervised clustering [Kwon et al., 2014] or only distributional semantics [Wyatt et al., 2005] but for different purposes, with different requirements and achieving lower results. When compared to the best of these attempts, we achieve an increase of 32% in accuracy and of 5% in precision.

5.3. Chapter Structure

In this chapter, after reviewing the state of the art on activity recognition in Section 5.4, we introduce our own approach in Section 5.6. To assist with the reader's understanding, we provide background information on the distributional semantics service, sensor representation and automated annotation process that we used. We propose our evaluation and discuss its results in Section 5.7, detailing the public dataset and clustering algorithms that we used and compared with.

5.4. Related Work

Activity recognition based on sensor data processing is performed using either specification-, ontology- or learning-based approaches. Specification-based approaches represent expert knowledge in logic rules. There has been a transition from first-order logic [Gu et al., 2004, Ranganathan et al., 2004a] to more formal logic models [Loke, 2009] which can achieve efficient reasoning. Ontology-based approaches are complimentary to formal logic ones in that an ontology can provide a standard vocabulary of concepts to represent domain knowledge and specifications [Chen et al., 2004, Ranganathan et al., 2004b]. The spread of resource-constrained devices has undermined the performance of specification-based approaches. It is less feasible to only use expert knowledge to define proper specifications of activities from a large amount of noisy sensor data. Envisioning the need for reasoning over our own system output in future developments, we reuse [Compton et al., 2012a] and other well-known ontologies for our data representation, following best practices.

Learning-based approaches can be further classified as either data or knowledge driven. Data-driven methods usually have classification models based on probabilistic reasoning [E.M. et al., 2004, Wilson, 2005] but the training examples required are expensive to collect (e.g., in the smart environment domain) and the assumption of independent observations is not suitable for dealing with concurrent or interwoven activities. Our system relies on activity descriptions from external corpora and on unsupervised clustering, with no distribution assumption, thus avoiding the above issues.

Knowledge-driven (or unsupervised) approaches incorporate the knowledge, thus requiring no training and allowing the knowledge to be reused across different systems [Huynh, 2008, Li and Dustdar, 2011, Gayathri et al., 2014]. To the best of our knowledge, the unsupervised learning method applied to activity recognition that has

achieved the highest precision so far is Kwon et al. [Kwon et al., 2014]. They investigate unsupervised learning methods for human activity recognition using smartphone sensors and when the number of activities k is known their hierarchical agglomerative clustering algorithm achieves 79% of precision. For this reason, we also used hierarchical clustering.

To reduce the amount of labelling required, several *semi-supervised* learning techniques have been proposed [Guyon and Elisseeff, 2003, Blum and Mitchell, 1998, Lewis and Gale, 1994, Thrun and Mitchell, 1995, Fernyhough et al., 2000]. These techniques all obtain sparse labels from end users. Our approach potentially enables completely unsupervised learning of labels from digitised common sense, as similarly explored by [Wyatt et al., 2005].

Several efforts have relied on machine-usable common sense to enable intelligent perception. The systems in [Craven et al., 1998, Etzioni et al., 2004] use statistical data mining techniques to extract information from the Web and accumulate common sense repositories. Any of these results can be fed to our system as a text corpus.

Leggieri et al. [Leggieri et al., 2010b] envisioned the usage of digitised common sense to improve reasoning over sensor data, levering the Linked Data principles as subsequently realised by [Bimschas et al., 2011, Leggieri et al., 2012b]. The web services [Page et al., 2009, Broering et al., 2011b] attempt to facilitate the creation of Linked Data for sensors but, unlike LD4S [Leggieri et al., 2011b], without allowing the client to customise the link creation.

Distributional Semantic Models (DSM) are based on the observation that semantically-similar words occur in similar contexts [Landauer and Dumais, 1997, Lund and Burgess, 1996]. They have been criticised for their lack of consideration of logical structures [Baroni and Zamparelli, 2010], while other systems [Landauer and Dumais, 1997, Grefenstette and Sadrzadeh, 2011, Baroni and Zamparel compute vector representations for larger phrases as composed by their parts. However, we are not interested in a thorough analysis of the logical structure of the textual content and therefore we use DSMs in our approach. Logical structure is intended here as, e.g., a relation of causality between two phrases. We are not interested in this because we apply the distributional semantic algorithm not on elaborate texts but on short labels used to identify RDF nodes. In fact, once the sensor data is sent to LD4S, it is represented as Linked Sensor Data in RDF.

5.5. Hierarchical Clustering on Sensor Lexicalisation

In this section, we propose an innovative methodology to predict the likelihood of a sensor producing relevant readings for an ongoing activity (see Algorithm 1). First, we query all the readings and metadata of sensors located in a specific location during a specific time range ($A = \{search\_results\}$). The query is constructed using the SPARQL query language which is designed to span across distributed datasets. Our system runs it against a public list of all open sensor datasets, available on the DataHub framework. DataHub is a data management platform from the Open Knowledge Foundation that exposes a JSON API to access metadata from the registered datasets. In doing so, we assume the data is compliant with our Linked Data representation (Chapter 4). This provides support for the inclusion of additional datasets as a consequence of the Internet of Things expansion.

Algorithm 1: Algorithm used in our methodology to predict sensors relevant for an activity.

```
   Data: Location, TimeRange
   Result: Clusters of objects likely to be relevant (i.e., used) during the same
           activity
 1 searchResults = queryDatahub(Location, TimeRange);
 2 activatedSensors = getSensorsfromReadings(searchResults);
 3 for sensorX in activatedSensors do
 4 |   for sensorY in searchResults do
 5 |   |   if sensorY not in activatedSensors then
 6 |   |   |   similarity = getESASimilarity(sensorX, sensorY);
 7 |   |   |   addToDistanceMatrix(similarity, matrix)
 8 |   |   end
 9 |   end
10 |   clustering(matrix)
11 end
```

We consider $B \subseteq A$ is the set of sensors whose readings represent a change in status, e.g., a change in temperature. The set of sensor readings C is such that $\forall x \in B : reading(x) = y \in C$ with reading() $injective \wedge surjective$. Our system then predicts which other sensor $z \in (A \setminus B)$ is likely to produce readings that will be relevant for the current ongoing activity. It obtains the semantic relatedness of each pair (x, z) where $z \in (A \setminus B) \wedge x \in B$ via a web service (Section 5.6.1) that had previously applied

ESA [Gabrilovich and Markovitch, 2007] on the English Wikipedia archive dump dated 2013[4].

Such relatedness is *semantic* or meaningful because it is calculated by considering the pair of sensors as not just mere electronic devices but rather in terms of the (semantic) function that they have from a natural language (human) perspective. For example, Figure 5.1 shows that a switch sensor attached to a fridge is uniquely identified in terms of its semantics as ¡switch, fridge¿ because once it is deployed, the human end user will not be interested in it as an electronic component, but rather as a provider of switch information about the fridge. As a use case, Figure 5.1 displays the example of users who use a door switch sensor to monitor information about how many times they go to a fridge during a typical week, for dieting purposes.

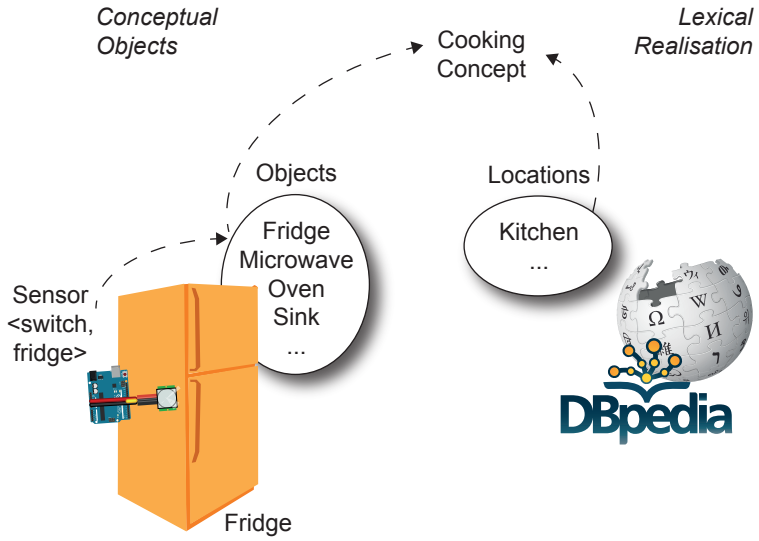


Figure 5.1.: Correspondence between the lexical realisation of a *switch* sensor and the concept object *fridge*. Fridge is then related to other objects, locations and activities according to what is extracted from Wikipedia.

In our terminology, a sensor is identified by $< op, foi >$, e.g., $< switch, fridge >$, where *op* is called the *observed property*, *foi* is called the *feature of interest*, and they are essentially what is referred to as the *sensor metadata*. In other words, our methodology relies on some correspondence between the lexical realisation of sensors and the conceptual objects that they are attached to, as in Figure 5.1. Such correspondence is often encountered when dealing with common sense knowledge, and its representation is supported by the ontology lexicalisation [Mccrae et al., 2012].

[4] http://treo.deri.ie/esa_downloads/data_wikipedia_en_2013.zip

Finally, our system collects all sensor similarities in a sparse matrix and runs three different hierarchical clustering algorithms on this matrix. Each resulting cluster corresponds to an activity, and its members are those sensors that will likely sense a change of status relevant for that activity. For example, the fridge switch sensor will likely be relevant whenever the microwave switch has previously sensed a change of status, i.e., fridge switch and microwave switch sensors will be part of the same cluster.

We do not perform cluster labelling yet but this could easily be realised. In fact,thanks to our data representation that makes use of ontologies, one could simply find the closest common ancestor between concepts (e.g., in an upper-level ontology such as UMBEL[5] or Cyc[6]) corresponding to the terms from the same cluster.

In this way, we predict that, given information from the sensors attached to a microwave (relevant for the *cooking* activity), the next most relevant information will come from sensors attached to the fridge. Consequently one could select which are the next sensors to query. Also, whenever one of the currently queried sensors fail, the system would know which other sensors can be substituted for it without lowering the information gain. Finally, if sensors that had not been predicted as relevant do indeed sense a change of status as well, it is easy to recognise their readings as part of a parallel activity which frequently occurs in multitasking human behaviour. This facilitates the definition of activity boundaries.

5.6. Methodology

In the following sections we will motivate and describe the services used and the underlying theory that they rely on. In Section 5.6.1 we present the Distributional Semantics theory that we use to pre-process Linked Sensor Data. Afterwards, we feed the resulting similarity measures among couples of ICOs/devices to hierarchical clustering algorithms presented in Section 5.6.2.

5.6.1. Distributional Semantics

Distributional View on Meaning . Distributional semantics is built on the *distributional hypothesis* stating that words that occur in similar contexts tend to have similar

[5] http://www.umbel.org/
[6] http://sw.opencyc.org/

meaning [Turney and Pantel, 2010]. The distributional view on meaning is inherently differential, i.e., the differences of meaning are mediated by differences of distribution. Consequently, Distributional Semantic Models (DSMs) quantify the amount of difference in meaning between linguistic entities. Such differential analysis can be used to determine the semantic relatedness between words [Freitas et al., 2011]. As previously explained, we consider the correspondence between lexical realisation of sensors and the concept objects they are attached to. The differential nature of DSMs is suitable then for our problem space. Considered also the availability of high volume and comprehensive Web corpora, we decided to use DSMs to determine the relatedness between sensors within the context of an activity. The computation of semantic relatedness and similarity measures between pairs of words is one instance in which the strength of distributional models and methods are empirically supported [Gabrilovich and Markovitch, 2007].

Statistical Analysis of co-occurrences Distributional semantic models (DSMs) are models based on the statistical analysis of co-occurrences of words in large corpora. They automatically harvest meaning from unstructured heterogeneous data and build comprehensive semantic models.

Explicit Semantic Analysis (ESA) [Gabrilovich and Markovitch, 2007] represents text by relying on the co-occurrence of words in a large corpus of articles, e.g. Wikipedia. A document containing a string of words is considered as the centroid of the vectors representing its words. Words are represented by vectors of their associations to each concept. Each association is determined using TF-IDF scoring, while cosine similarity measures the semantic relatedness between pairs of words.

Given a set of concepts $C_1, ..., C_n$ and a set of associated documents $d_1, ..., d_n$, ESA builds a sparse table T where each of the n columns corresponds to a concept, and each of the rows corresponds to a word that occurs in $\bigcup_{i=1...n} d_i$. An entry $T[i, j]$ in the table corresponds to the TF-IDF value of term t_i in document d_j.

$$T[i, j] = tf(t_i, d_j) * log\frac{n}{df_i} \tag{5.1}$$

where $tf(t_i, d_j)$ is the *term frequency* of the term t_i in the document d_j defined as

$$tf(t_i, d_j) = \begin{cases} 1 + log(count(t_i, d_j)) & \text{if } count(t_i, d_j) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (5.2)$$

while $df_i = |d_k : t_i \in d_k|$ is the *document frequency*, i.e., the total amount of documents that contain t_i.

EasyESA . The size of the textual corpus on which semantic models rely upon is critical to the quality of the results. This leads to high hardware and software requirements on the implementation side (e.g., the English version of Wikipedia 2013 contains 43 GB of article data). For simplicity, we use *EasyESA* [Carvalho et al., 2014], a JSON webservice which implements ESA based on Wikiprep-ESA[7]. It can be queried for either the semantic relatedness measure, concept vectors or the context windows. In particular, we query the online available instance[8] which run ESA on the English version of Wikipedia 2013. The query asks for semantic relatedness of pairs of sensors represented as tuples of terms like ¡ switch, fridge ¿.

5.6.2. Unsupervised Hierarchical Clustering

Unsupervised methods . We chose unsupervised methods because we believe that given the amount of different activities and sensors involved, supervised methods are not likely to scale with the expansion of the Internet of Things phenomenon. In particular, we chose hierarchical clustering because it is the approach that has so far achieved the better precision [Kwon et al., 2014].

We applied **three** different hierarchical clustering algorithms in our experiments: 1. Weighted Pair Group Method with Arithmetic mean (WPGMA) 2. Unweighed Pair Group Method with Arithmetic mean (UPGMA) 3. Farthest Point Algorithm, also called Voorhees (VH). We applied UPGMA mainly because it reflects observable similarities between activities by the distance of their semantic distribution. Thus, it perfectly fit our goal. WPGMA was chosen to explore the possibility that the structural subdivision of the objects (i.e., cluster items) had an influence in the belonging of the object to the

[7] https://github.com/faraday/wikiprep-esa
[8] http://vmdeb20.deri.ie:8890/esaservice

activity (i.e., cluster). The application of VH was investigated to explore the possibility that one of such objects may be central and more critical in the creation of the clusters.

Hierarchical Clustering . Given a set of N items to be clustered, and an $N * N$ distance (or similarity) matrix, the basic process of hierarchical clustering is the following.

- Each item is assigned to a cluster. Therefore, given N items, there will be N clusters, each containing just one item. The distances between clusters are given by the distances between the items they contain. Here *distance* corresponds to similarity (from the similarity matrix).
- The closest (most similar) pair of clusters are identified and merged into a single cluster.
- Compute distances (i.e., similarities) between the new cluster and each of the old clusters (also called *extant clusters*).
- Repeat the previous two steps until all items are clustered into a single cluster of size N.

What differentiate distinct hierarchical clustering algorithms is how to compute the distances between the new cluster and each of the old ones (as in the step (3) above). The different possible ways to compute this distance are classified as single-linkage, complete-linkage and average-linkage clustering. In single-linkage clustering, the distance between one cluster and another cluster is equal to the shortest distance from any member of one cluster to any member of the other cluster. In complete-linkage clustering, such distance is equal to the greatest distance from any member of one cluster to any member of the other cluster. In average-linkage clustering, the distance between one cluster and another cluster is equal to the average distance from any member of one cluster to any member of the other cluster.

Algorithms . UPGMA and WPGMA perform *Average Linkage Clustering*. They both compute the average similarity of a candidate cluster to an extant cluster (average arithmetic). As a result, they construct *ultrametric trees*, where the ultrametricity is satisfied when

$$\forall A, B, C \in \{taxa\} : d_{AC} \leqslant max(d_{AB}, d_{BC}) \tag{5.3}$$

where d_{AC} is the distance between A and C. In other words, in the resulting tree for every three distances under consideration, two are equal or larger than the third one. The tree is rooted (also called *dendrogram*) with all the end nodes equidistant from the root.

The UPGMA and WPGMA methods use a sequential clustering algorithm, in which sequences or groups of sequences called *Operational Taxonomic Units (OTUs)* are clustered in a new single OTU, if most similar to each other. The similarity is calculated among double numbers, output of the distributional semantic algorithm as a result of the semantic similarity between a couple of terms.

UPGMA is a bottom-up method that maps structures hidden in the pairwise similarity matrix (resulted by applying Easy-ESA, as in Section 5.6.1) into the dendrogram. At each step, the distance between any two OTUs A and B is calculated as the average of all distances between pairs of objects $x \in A$ and $y \in B$, i.e., the mean distance between elements of each cluster (see Equation 5.4).

$$d(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} dist(x_{ri}, x_{sj}) \tag{5.4}$$

The averages are then weighted by the number N of OTUs in each cluster $(A,B),C$, at each step, as in Equation 5.5. As a result, each distance contributes equally to the final result, though increasing the time complexity.

$$d(r,s) = \frac{N_{AB} d_{(A,B)k} + N_C d_{Ck}}{N_{AB} + N_C} \tag{5.5}$$

While UPGMA weights each item in the candidate cluster equally, regardless of its structural subdivision, WPGMA weights the member most recently admitted to a cluster equal with all previous members. It bases the averaging of the distances on the total amount of clusters rather than of OTUs per cluster, as shown in Equation 5.6.

$$d(r,s) = \frac{d_{(A,B)k} + d_{Ck}}{2} \tag{5.6}$$

At each step, assuming that B, C are clustered to form a new OTU D. Then, the distance $d(A, D)$ between cluster A and D is calculated as in Equation 5.7:

$$d(A, D) = \frac{1}{2}(d_{A,B} + d_{A,C}) \tag{5.7}$$

When there are unequal numbers of items in the clusters, the distances in the original matrix do not contribute equally to the intermediate calculations. Both UPGMA and WPGMA All run with time complexity $O(n^2)$.

The Farthest Point clustering algorithm or Voorhees(VH) is also a bottom-up hierarchical clustering method. However, unlike UPGMA and WPGMA, relies on centroids rather than averaging. If we denote $z_i \in X_i, i = 1, .., n$ as the centres of the OTUs X_i, then to generate $k = \frac{n}{n_m in}$ clusters we have to solve the *k-centre problem.*

K-Centre Problem 1 *Given a set of n points, find a partition into clusters $t_i, .., t_k$ with centres $c_i, .., c_k$ so as to minimize the functional cost:*

$$\max_{i=1,..,k} \max_{j \in t_i} \|z_j - c_{i_2}\| \tag{5.8}$$

This problem is NP-hard and the Farthest Point Clustering (FPC) algorithm or Voorhees (VH) computes an approximation for it. VH is widely used for large k cases and it consists in the following steps.

- Select any point from the set and consider it as one center;
- Search for the farthest point from the remaining set and consider it as another center;
- Repeat the two steos above until k centers are found.

In other words, at each step $i = 1, ..k - 1$, the algorithm chooses a random centre c_i and finds c_{i+1} such that

$$d_i(c_{i=1}) = \max_{j=1,..n} d_i(z_j) \tag{5.9}$$

where $d_i(z) = \min_{j \leqslant i} \|z - c_j\|$ is the distance between z and the computed centres $c_j, j \leqslant i$. At a generic iteration i VH knows the set of centres c_{i-1} (i.e., computed at the previous iteration) and a mapping μ associating each point $z_i, i = 1, .., n$ to its nearest centre $c_i, .., c_k$. After k iterations, the set $\{c_i, ..c_k\}$ and the mapping μ define the clustering. It has been demonstrated that the random choice of c_i in the initialisation phase does not affect either the efficiency or effectiveness of VH.

VH runs in linear time. As approximation with factor less than 2 is proved to be NP hard and VH having factor = 2, VH was regarded as the best approximation possible. The time complexity can be improved to $O(nlogk)$ with box decomposition technique.

Implementation . In our system we used the implementation of the WPGMA, UPGMA and VH algorithms provided by the Python library HCluster[9]. It is part of Scipy[10] a Python-based ecosystem of open-source software for mathematics, science, and engineering. We used HCluster to generate hierarchical clusters from distance matrices, compute distance matrices from observation vectors, compute statistics on clusters, cutting linkages to generate flat clusters and visualising clusters with dendrograms.

5.7. Evaluation

Our goal is to predict which sensors provide relevant information during an activity logging. We compare the list of "relevant sensors per activity" returned by our system with the sensors manually annotated as part of such activity logging. These annotations and readings are taken from the public[11] dataset MITes [Tapia et al., 2004] and were collected during live experiment settings. We pre-processed such dataset (i.e., CSV files of sensor readings and metadata about both sensors and activities) to form HTTP PUT requests to the LD4S API for annotating and storing the data, as in Listing 5.1. Based on such comparison, the overall accuracy and precision of our system are calculated when applying either of the clustering algorithms UPGMA, WPGMA or VH.

```
1 PUT ld4s:device/2_99
2
3 payload: {'observed_property': 'switch',
4 'location-name': ['Kitchen'],
```

[9] https://pypi.python.org/pypi/hcluster
[10] http://www.scipy.org/
[11] http://courses.media.mit.edu/2004fall/mas622j/04.projects/home/thesis_data_txt.zip

```
5 'foi': ['Fridge']}
6
7 headers: {'Content-type': 'application/json',
8 'Accept': 'application/x-turtle'}
```

Listing 5.1: HTTP PUT request forwarded to the LD4S RESTful API.

DataHub (see Section 5.5) was then queried for all the sensor datasets available[12] thus returning a JSON list of details of these datasets such as their ID, title, tags, license and endpoint URIs. The system filters only those datasets that either have no license or grant an open-access 1. expose a SPARQL endpoint and forward the query in Listing 6.1 towards each of them.Since LD4S triple store is published on DataHub, its endpoint is also mentioned in such JSON list. Consequently, our query will be forwarded to the LD4S endpoint as well, so that we will actually get all the data that we had annotated and stored in the pre-processing step but while also assuring that any other potential dataset is considered.

The results obtained from each endpoint are XML files - as by W3C standard recommendation - that the system merged and parsed to distinguish between sensors that sensed a change in status and the others who just happened to share the same location. In this experiment we evaluated the worse case: only one sensor has recently sensed a change in status. The semantic relatedness must be calculated between the higher amount of possible pairs that share the same location at the same time. This is used to fill a distance matrix on which the hierarchical clustering algorithms were applied. In addition to precision and overall accuracy, we also evaluated the performances in terms of execution time for the different HTTP requests, the SPARQL queries, the whole pre-processing step and the overall system.

5.7.1. MITes Dataset

Tapia et al. [Tapia et al., 2004] published the MITes dataset from an experiment where human activity was collected for two weeks. They installed 200 switch sensors deployed on 27 different *features of interest* (FoIs) in two single-person apartments. The sensors were installed in everyday objects such as drawers, refrigerators, containers, etc. to record opening-closing events (activation deactivation events) as 2 subjects carried out everyday activities. The subjects used a software application while they were performing

[12] http://ckan.net/api/3/action/package_search?q=sensor

an activity, to manually annotate it. This resulted in the annotated activities associated with readings as in Table 5.1. In our experiment we used the data from both subjects combined together, since evaluating the system differently according to the subject at end was out of the scope of this paper.

Table 5.1.: Activities labelled in the MITes dataset.

Number of Examples per Class		
Activity	Subject 1	Subject 2
Preparing dinner	8	14
Preparing lunch	17	20
Listening to music	-	18
Taking medication	-	14
Toileting	85	40
Preparing breakfast	14	18
Washing dishes	7	21
Preparing a snack	14	16
Watching TV	-	15
Bathing	18	-
Going out to work	12	-
Dressing	24	-
Grooming	37	-
Preparing a beverage	15	-
Doing laundry	19	-
Cleaning	8	-

5.7.2. Similarity Results

We considered the worst case in which only one of the sensors sharing the same location at the same time range has recently sensed a change in status for the current ongoing activity, while all the other nearby ones, which will likely do so in the near future, have to be predicted. In this case, given n sensors, the amount of pairs to check for semantic relatedness is the binomial coefficient as in Equation 5.10. In our case since there are 27 different features of interest, there are 27 different types of sensors and 351 distinct pairs.

$$\binom{n}{2} = \frac{n!}{2!(n-2!)} \tag{5.10}$$

Even though the binomial coefficient grows quickly, it only depends on the amount of features of interest rather than on the amount of actually deployed sensors. At the same time, the amount of ICOs is expected to grow but the amount of "types" of sensors is not, since there is only so much in the real world that can be monitored by sensors. Moreover, this amount can be reduced by exploiting the linked data representation. Since we consider each feature of interest as an ontological concept, we could calculate the similarity only between pairs of concepts that are not distant within an upper level ontology, more than a certain threshold. We could then ignore the more distant once. This would even further reduce the amount of comparisons required. Therefore, our method then is not expected to hinder the system from scaling during the Internet of Things expansion. The growth of time cost is analysed more thoroughfully in Section 5.7.4.

The lowest semantic similarity value calculated was -1.0 for the pair $switch, tv$ and $switch, hamper$, followed by 0.00036 for the pair $switch, jewelry\_box$ and $switch, microwave$. The highest similarity value was 0.75839 for the pair $switch, cabinet$ and $switch, medicine$, followed by 0.11285 for the pair $switch, refrigerator$ and $switch, freezer$.

5.7.3. Algorithms Comparison

The hypothesis we wanted to verify by applying the chosen algorithms were 1. UPGMA: is the distance of the semantic distribution of similarities relevant in predicting the sensor-activity association? 2. WPGMA: does considering the structural subdivision of the sensor objects positively influence such prediction? 3. VH: can we rely on the assumption that each activity is associated with a more central (i.e., critical) sensor object?

The evaluation results particularly confirm the second and third of the hypothesis above, since VH achieved the highest precision followed by WPGMA. Figure 5.2 shows the results we obtained by running UPGMA over the MITes dataset. The final clustering actual reflects the common knowledge, e.g., by grouping freezer and cold sink faucet together. However, too many sensors are too distant from any specific cluster.

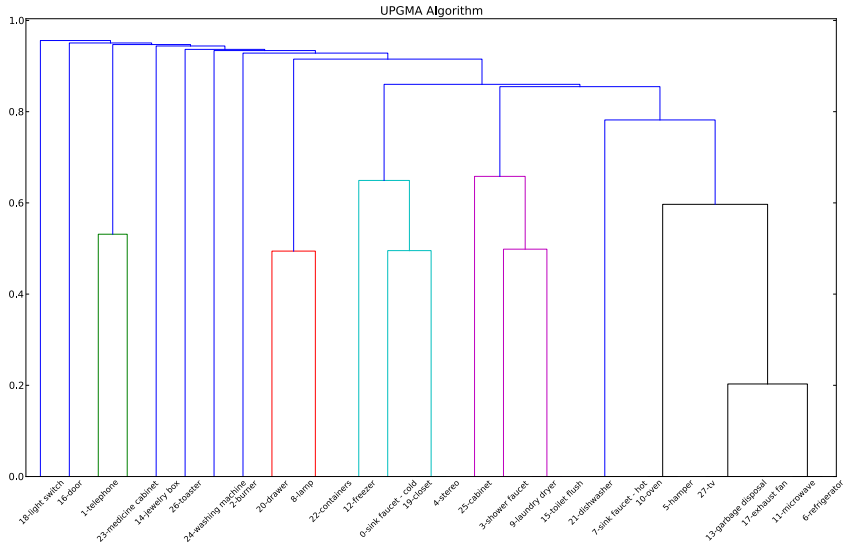


Figure 5.2.: Clustering performed by the UPGMA algorithm.

By applying WPGMA we got a better distribution of clusters, as shown in Figure 5.3. This result confirms that the sensors have structural relationships between each other that can be relevantly considered during the clustering.

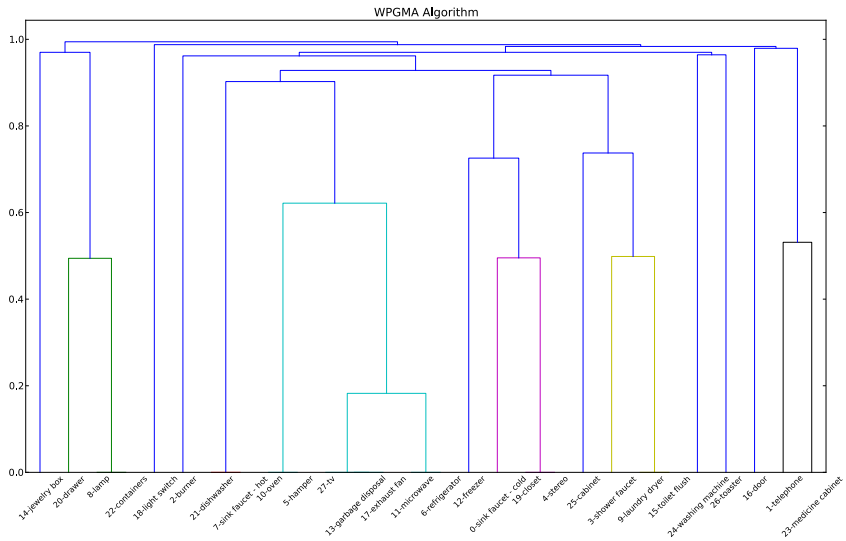


Figure 5.3.: Clustering performed by the WPGMA algorithm.

The results of applying Voor Hees (VH) are shown in Figure 5.4. The VH algorithm resulted in no sensor being distant from any specific cluster. Unsurprisingly then, this approach achieved the highest precision.

When comparing our results with the annotated dataset, since we do not perform cluster labelling, it was not possible to directly map our clusters to the labels in Table 5.1.

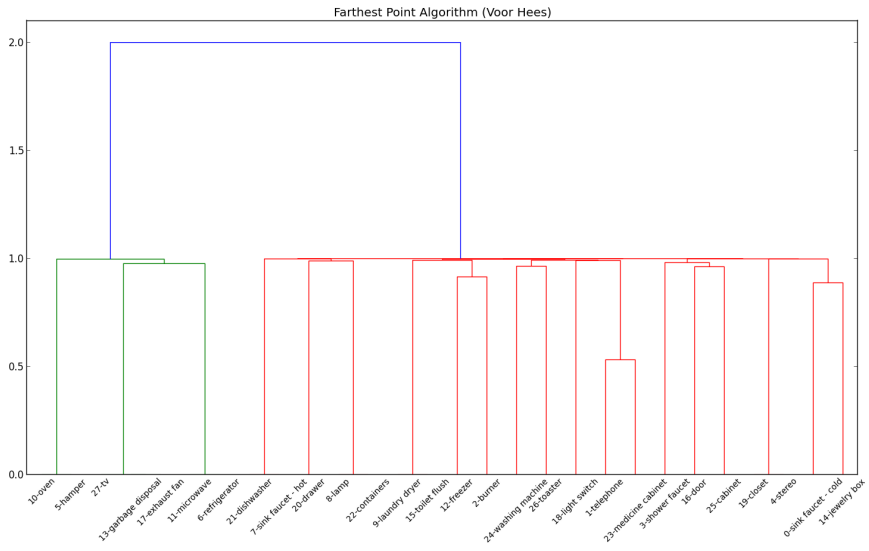


Figure 5.4.: Clustering performed by the Voor Hees algorithm.

However, we considered the match verified whenever the sensors belonging to the same cluster according to our system (i.e., *predicted* class) were the ones that sensed the same activity in the MITes annotations (i.e., *actual* class). Consequently, we considered a 2-class classification problem, i.e., whether the sensors actually part of the same activity had been clustered in the same cluster. As a result a separate confusion matrix (Table 5.2) was created for each of the annotated activity.

Table 5.2.: Confusion matrix displaying number of true positives, true negatives, false positives and false negatives for a 2-class classification problem.

Predicted vs Actual		Actual class	
		1	2
Predicted class	1	TP_{11}	FP_{12}
	2	FN_{21}	TN_{22}

With such settings, we calculated precision and overall accuracy.

$$Precision = \frac{TP_{11}}{TP_{11} + FP_{12}} \tag{5.11}$$

$$Accuracy = \frac{TP_{11} + TN_{21}}{TP_{11} + TN_{22} + FP_{12} + FN_{12}} \tag{5.12}$$

Figure 5.5 shows the precision percentage achieved by our system on the given dataset, by using each of the hierarchical clustering algorithms. VH achieves an average precision of 89.5% followed by WPGMA which achieves 85.6% and UPGMA with 75.2%.

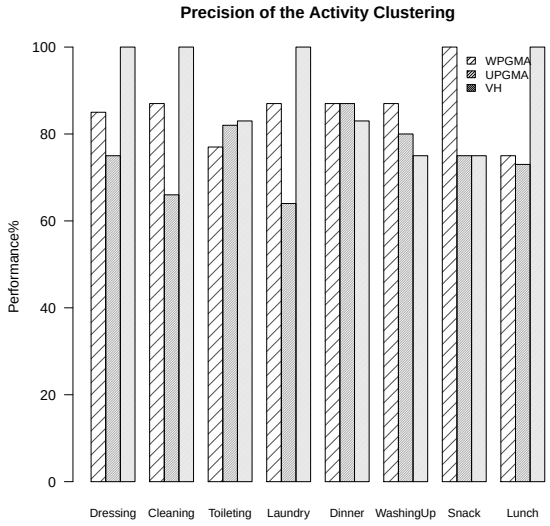


Figure 5.5.: Comparison between precision percentages achieved by the clustering algorithms for some of the activities.

Precision and overall accuracy were calculated and our system managed to predict which sensors were going to provide information relevant for each of the 27 annotated activities with an average accuracy of 69%. Details of the accuracy achieved by each algorithm for some of the activities are in Figure 5.6.

We believe our results to be relevant especially when compared with **a.** Wyatt et al. [Wyatt et al., 2005] which we consider being the most similar previous research effort, since it used text-analysis of websites **b.** Kwon et al. [Kwon et al., 2014] which achieved the state of the art in terms of precision with unsupervised hierarchical agglomerative clustering for sensor-based activity recognition. The experiments that we run is compared in Table 5.3 with those run by Wyatt et al. and Kwon et al. Although our goals differ between Activity Recognition (AR), Activity Inference (AI) and Relevant Sensor Prediction (RSP), if each cluster is considered an activity we can then compare our results.

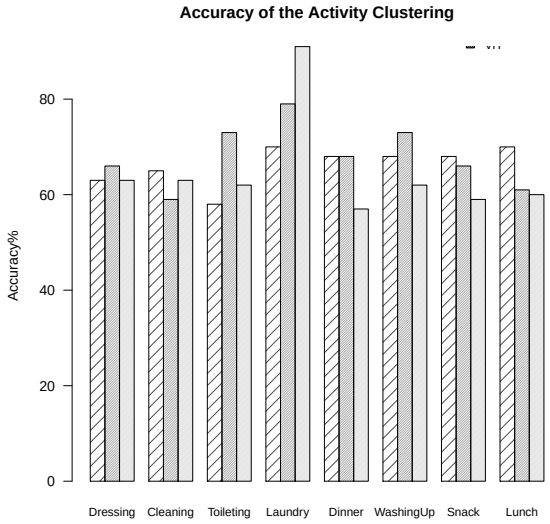


Figure 5.6.: Comparison between accuracy percentages achieved by the clustering algorithms for some of the activities.

Table 5.3.: Comparison between the experiment setup and results for our own approach and the previous closest research efforts.

	Kwon et al.	Wyatt et al.	Ours
# Sensors	3	100	200
# Activities	5	26	16
Collection Time	50 mins	360 mins	2 weeks
Goal	AR	AI	RSP
Algorithms	HIER	HMM	UH
Precision	79%	70%	89%
Accuracy	-	52%	69%

As precision and accuracy we considered the best values among the distinct attempts made using algorithms such as Unsupervised Hierarchical Agglomerative Clustering (HIER), Hidden Markov Models (HMM) and Unsupervised Hierarchical (UH) Algorithms. We can see how the datasets also differ among the listed approaches. Our data were related to an average amount of activities (i.e., fewer clusters to identify), sensed by an higher amount of sensors and over a longer time period. However, we considered the worst case scenario where only one sensor had been activated and labelled. This is supposed to be mediating on the advantages we were given by th higher quality of our initial dataset. Our system improved the accuracy by 32% and the precision by 5% with

respect to such previous efforts from the state of the art, although this result should be carefully considered as it is based on significant differences in the experiment settings.

5.7.4. Performance

The evaluated system run on a laptop equipped with Intel Core™2 Duo and 305GB of disk space. We used the LD4S and EasyEsa service instances running on external servers in order to support and test a modular and distributed architecture. These were Jetty servers running on a virtual machine with Debian Operating System. They used MongoDB [Inc., 2011] to store the *English Wikipedia 2013* dump archive.

During the pre-processing step, the HTTP PUT requests forwarded to LD4S to both create the annotation as in Chapter 4 and store it in the LD4S triple store had an average execution time of 3.226 milliseconds. The overall system execution (excluding the pre-processing step) time was of 18.464 milliseconds. Forwarding a query to DataHub (Section 5.5) to retrieve all the available sensor datasets had an execution time of 3.177 milliseconds; returning 20 datasets out of which 3 were both featured with an open license and exposing a SPARQL endpoint. Among them, only LD4S was actually accessible so that the average response for the SPARQL queries we run is actually referring only to the queries (Listing 6.1) run on LD4S and it is equal to 246 milliseconds.

Our system took 14.119 milliseconds to calculate the semantic relatedness of 351 pairs of sensors, during which the HTTP requests to the Easy-Esa API achieved an average response time of 9 milliseconds. In Figure 5.7 we analysed the growth of time cost for the similarity calculation with respect to the amount of sensor types. The highest time cost is 1 minute and 26 seconds for comparing 216 distinct sensor types, thus confirming our scaling expectation. As the amount of sensors that have already sensed a change in status for the current activity grows, the amount of sensors to predict as involved in the activity or not decreases.

Finally, during the clustering step in which the 3 hierarchical clustering algorithms run - both UPGMA and VH performed the clustering in 002 milliseconds while WPGMA took 012 milliseconds. The performance values achieved confirm the possibility of updating the clustering with new sensors similarities at run-time. Considered that some of the most constrained resource devices are characterised by limited RAM up to around 4 kB and limited ROM up to 128 kB, it may not be possible for them to run the clustering by themselves.

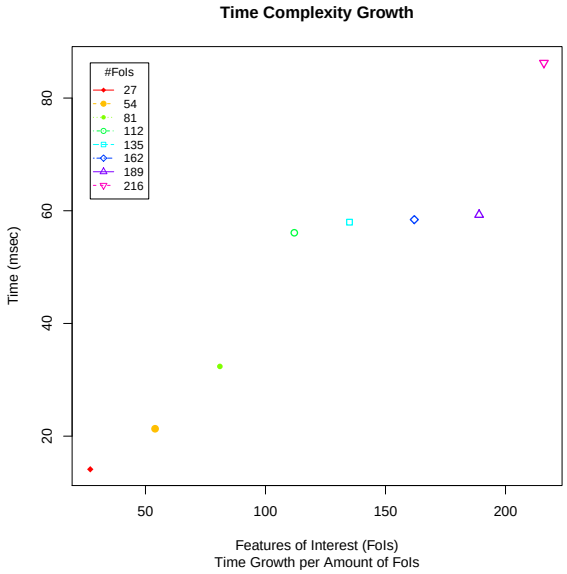


Figure 5.7.: Time complexity growth for the semantic relatedness calculation as the amount of FoIs increases.

However, a clustering for most of the possible features of interest could be pre-computed offline. The system may output a lookup table listing clusters of ICOs. A constrained resource device that wants to know which devices are currently relevant, only has to browse the lookup table. Storing our sensor representation is possible thanks to the availability of a RDF triple store for embedded devices [Hasemann et al., 2012].

5.8. Conclusion

Our approach managed to predict the sensors relevant for an ongoing activity with high precision and accuracy. We supported the uptake and reproducibility of our methodology by using online available services and datasets. We demonstrated the actual advantages, such as **1.** the independence from any a-priori knowledge and from any manual settings **2.** the scalability and support for distributed datasets **3.** the performance that makes it suitable for run-time execution **4.** the suitability to run on constrained resource devices (running clustering and text analysis steps offline) . In the future we will explore the several potential advantages, such as **1.** refining the results by running ESA on a domain-specific corpus for a specific scenario **2.** considering any sensor activated during an ongoing activity but not predicted by our system as a sign of unusual co-occurring events (i.e., activity boundaries recognition) **3.** further reasoning on the cluster centroids

in order to label each cluster by relying on the use of ontology in our system **4.** use our prediction to determine the sensors with which to substitute faulty ones during the currently ongoing activity logging without lowering the information gain..

Chapter 6.

G-Sensing: Bridging the Gap between Real Places and their Web-Based Representations

Based on "Using Sensors to Bridge the Gap between Real Places and their Web-Based Representations" [Leggieri et al., 2015b] published at the 10th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2015)

Finding relevant and reliable information on the Web is a non-trivial task, yet the tools to help us do this have reached a high level of maturity. For example, Google not only indexes all Web information, but also provides good results from the information that is contained within online fora, social networks, etc. We argue, however, that there are search requests that cannot be satisfied by traditional browsing or by existing technologies. The focus of this chapter is on users searching for information about real-world locations such as hotels, restaurants, bars, shops, as well as offices, governmental or private institutions. A user's location-related information needs often refer to data which is only valid for very specific and short time frames. As a result, they are typically not maintained on a Web page, let alone indexed by a search engine. Consider a user who is searching for an acupuncture clinic so as to book a visit. Using Google and Google Maps for example, the user can find suitable candidates in a given area as well as links to their official websites. Such websites usually provide information about the services that they offer, contact details, and maybe some pictures. The user can also read reviews about the clinic on recommender sites such as WhatClinic.com. However, the user might also be interested in live data which could help them with their choice of a clinic,

e.g., data about the number of free parking spaces close to the clinic or the number of customers currently in the waiting room.

As thoroughly described in Chapter 2, in recent years, there has been a tremendous uptake in the deployment of sensors within the concept of *Internet of Things*, including live camera feeds, weather sensors, etc. Data streams from such sensors are often publicly accessible, but are usually decoupled from other related Web resources (in our case, websites referring to real-world places near a specified geolocation). From a website provider's perspective, integrating live data is costly. Firstly, the data is distributed across different sources, i.e., live data providers. This includes that the number of sources may change over time. Secondly, the conventional static websites have to be modified to accommodate the nature of data streams.

In this chapter, we present G-Sensing, our approach towards the seamless integration of live sensor data into a user's normal browsing experience. To accomplish this, as a frontend application, G-Sensing features a browser add-on. The add-on injects live sensor information into Google search result pages – that is, if a search result refers to a real-world location, the add-on requests relevant sensor data from the backend and displays it adjacent to the corresponding result. Regarding our backend as a source of live data, we aim for an open and flexible infrastructure that will allow us to easily change between different publicly available data sources. As a set of minimum requirements, a G-Sensing-enabled data source must (a) expose a SPARQL endpoint to query the data, (b) provide semantically annotated sensor data, and (c) register on DataHub (Chapter 5, Section 5.5) with its sensor tag metadata. For our current system we have implemented LD4S (Linked Data for Sensors), which fulfils all three requirements. LD4S can be queried using SPARQL or via a RESTful API using the JSON data format. For our evaluation, we randomly generated sensor readings and metadata for 30 sensors. They were divided into groups of 10 sensors, each deployed within 1 km from three of the Galway acupuncture clinics listed in the first Google search result page. We achieved optimal performances at a minimum cost of bandwidth increase (only 30 KB). This solution improves the data quality for traditional Web content (ref. Contribution **C 4** and Reaearch Question **Q 4 Quality**: *How can contextualised sensors improve the quality of traditional Web content?*).

The chapter outline is: Section 6.1 describes G-Sensing, highlighting the core aspects of the browser add-on frontend application based on the LD4S backend infrastructure (described in Chapter 4). Section 6.2 presents the results of our evaluation to illustrate the practicality of our system. Section 6.3 provides a discussion and outlines a roadmap

for our ongoing and future work. Section 6.4 reviews related work to put our approach into context. Section 6.5 concludes this chapter.

6.1. G-Sensing

The most common way to discover sensor data sources (as well as the live data itself) is through the means of dedicated platforms. For example, users can browse DataHub's list of publicly accessible sensors. However, this decouples live data from the more traditional Web resources like websites, and therefore puts them somewhat out of the reach for normal users who are browsing and searching the Web. We argue, however, that users' search requests often refer to information about a physical location. In these cases, users could potentially benefit from the information stemming from live data that is typically not shown on websites relating to a particular location. Regarding our example, the websites for acupuncture clinics are unlikely to provide information about the number of currently available parking spots nearby. G-Sensing supports a seamless integration of live data into the user's normal browsing experience. G-Sensing features an add-on that injects relevant live sensor information into Google search result pages. The sources of sensor data are registered datasets on DataHub. This approach allows the use of different and/or multiple datasets as data sources for the live data to be requested by our add-on. With LD4S, we have implemented our dataset of semantically-annotated sensor data, and registered it on DataHub (as part of the dataset mentioned in Chapter 1, Section 1.3). LD4S is the backend of the G-Sensing application while we describe its frontend - the Mozilla Firefox plugin - in Section 6.1.1.

6.1.1. Frontend Application – Browser Add-On

Once a user has installed our add-on, it performs the following two event-driven tasks: 1. discovery of (new) data sources after starting the browser 2. requesting / injecting live data after loading a new Web page.

(1) Discovery of data sources. We aim for an open and flexible infrastructure that allows us to easily change between different publicly available data sources. We only assume that such data sources expose a SPARQL endpoint to query the data, provide semantically-annotated sensor data, and are registered on DataHub with sensor-tag metadata. DataHub is a data management platform from the Open Knowledge Foun-

dation that exposes a JSON API to access metadata from the registered datasets (as in Chapter 5, Section 5.5). Anyone can register a dataset for free and specify its copyright, its access endpoints, and the type of data contained therein. As a result, the set of available data sources can change over time. To reflect this, we request information about available data sources registered on DataHub each time a user opens the browser. For each dataset retrieved, we verify whether it is publicly open and whether it exposes one or more SPARQL endpoints. If so, we forward a SPARQL query to get the sensor details for the coordinates of interest from each SPARQL endpoint.

(2) Content request and injection. The browser add-on listens to each page load event. If a new page has been loaded into the browser, Algorithm 2 is executed. We first test if the new page is a Google results page (Line 2). For the time being, we limit ourselves to Google result pages, but intend to extend our idea to any Web pages that refer to geographic locations (see Section 6.3 for details). We then extract all individual search results from the page (Line 3). For each result, we first extract the URL of the link (Line 5) and request any place information that is associated with that URL by sending a search request to our backend (Line 6). If the URL is linked to a physical location, we again send a request to the backend to fetch all live data from each of the discovered data sources for that location (Line 8). Finally, we update the result dictionary (Line 9) and return the dictionary (Line 13).

Algorithm 2: *handlePageLoad(url)*

```
1  liveDataMap ← {};
2  if isGoogleSearchResult(url) = True then
3  |   searchResults ← extractSearchResults(url);
4  |   foreach result ∈ searchResult do
5  |   |   resultUrl ← result.url;
6  |   |   place ← requestPlaceData(resultUrl);
7  |   |   if place != ⊥ then
8  |   |   |   liveData ← requestLiveData(place.coords);
9  |   |   |   liveDataMap[resultUrl] ← liveData;
10 |   |   end
11 |   end
12 end
13 return liveDataMap;
```

Having received all of the live data, the add-on displays the data next to the corresponding search result by injecting the content into the HTML page. Figure 6.1 is an

example screenshot showing the additional live data provided for acupuncture clinics in Galway, Ireland.

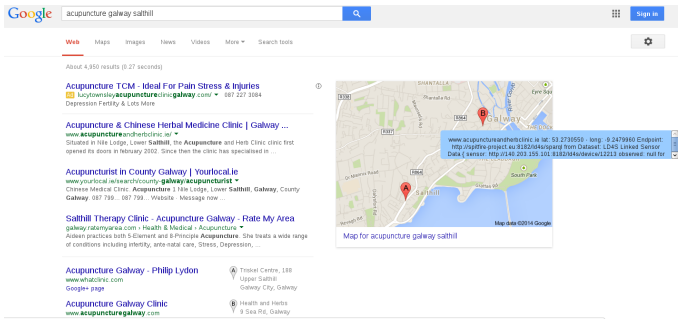


Figure 6.1.: G-Sensing overlays sensor data and metadata referring to the virtual representations of real places among Google search results in the browser.

6.2. Evaluation

We evaluated the added value of our basic idea, that is integrating live data into web-representations of physical locations. From now on, we refer to web-representations of physical locations as *physical locations' websites*. For this purpose, we calculated the coverage, that is the distribution of physical locations' websites. This measure, will indicate to us how feasible our approach is. If the coverage was low, there would be no reason for our G-Sensing application, since there would be no website to which live data could be injected.

On the other hand, we tested our system by simulating the deployment of sensors in locations related to a specific Google query. We considered three acupuncture clinics based in Galway, Ireland. We randomly chose them from the first page of GOOGLE's search results for the query "*acupuncture galway salthill*",[1] *Acupuncture & Chinese Herbal Medicine Clinic*,[2] *Acupuncture Galway Clinic*[3] and *Evidence-Based Therapy Centre*.[4] We randomly generated data and metadata for sensors deployed at different latitude and longitude coordinates within 1 km (approximately 0.009 degrees) of the coordinates of our three acupuncture clinics.

[1] https://www.google.ie/#q=acupuncture+galway+salthill
[2] http://www.acupunctureandherbclinic.ie
[3] http://www.acupuncturegalway.com
[4] http://www.ebtc.ie/acupuncture

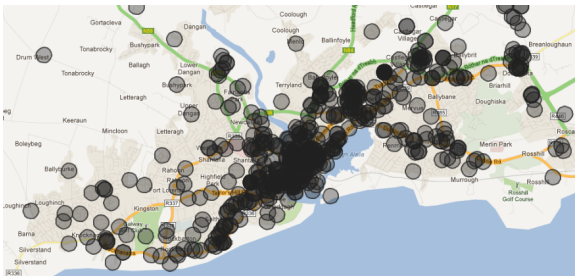
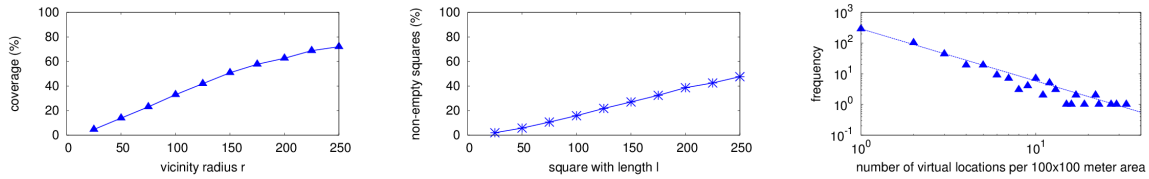


Figure 6.2.: Qualitative illustration of the coverage and distribution of places associated with a website across the city of Galway.

We simulated the deployment of groups of 10 sensors near each of the three acupuncture clinics of interest. A script implemented in JavaScript generated readings and metadata for a total of 30 sensors. It then forwarded PUT requests to LD4S for semantically linking, annotating and storing of such data (as described in Chapter 4).

We crawled Google Places to collect all places within the city of Galway. Our current dataset contains 3,692 locations, 1,455 (39.4%) of which are associated with a website, i.e., they have a URL.

6.2.1. Analysis of Data Repository



(a) coverage for different vicinity radiuses r

(b) ratio of non-empty squares

(c) distribution of non-empty squares

Figure 6.3.: Coverage and distribution analysis regarding virtual locations across the city of Galway, Ireland.

We first looked at the coverage, i.e., how much of the area defined by the virtual locations, i.e., places associated with a website, together within radius of Galway city. Figure 6.2 illustrates the coverage with a vicinity radius of $r = 150$m. To get more quantitative results, Figure 6.3(a) shows the percentage coverage as we vary the vicinity radius. Naturally, the coverage increases for larger r, resulting in up to 72% coverage for $r = 250$m. Regarding the distribution of virtual locations (i.e., real places that have a corresponding official website as their virtual representation), we divided the areas

of Galway into squares with different side lengths l and counted the number of virtual locations within each square. Figure 6.3(b) shows the percentage of non-empty squares, which naturally increases for larger-sized squares. Empty squares typically cover city parks or purely residential areas. Figure 6.3(c) shows the distribution of non-empty squares for $l = 100$m. Not unexpectedly, the number of virtual locations per square and their respective frequency shows a power-law relationship: while most squares only contain a small set of locations, a few squares will contain a very large number of locations (e.g., city centres, business parks). Given these results, we argue that there are many websites that refer to physical locations, emphasizing the added value of our approach for integrating live data into such websites.

6.2.2. Performance

For the performance experiments, we installed the browser add-on on commodity hardware equipped with an Intel Core 2 Duo processor and 305 GB of disk space. For G-Sensing to be of a practical use, it must not significantly add to a user's bandwidth consumption. We used the LD4S service instance running on an external server in order to support and test a modular and distributed architecture.

On average, Google result pages were around 145 KB in size. When G-Sensing is enabled, the bandwidth consumption is around 175 KB, an increase of just 30 KB (∼20%). Part of the bandwidth consumption can also be attributed to the use of HTTPS rather than HTTP. 20% is a modest but reasonable additional overhead, particularly, since the overhead is comprised of information that is useful to the user. In our ongoing work, we aim to filter the requested live data by tailoring it to the information needs of an individual user (cf. Section 6.3).

Apart from the overhead in terms of required bandwidth, we also measured the average time to request and receive the live data. For this, we first forwarded a query to the DataHub API to retrieve all of the available sensor datasets. This represents the data sources discovery task during a browser start-up. The execution time was 3 milliseconds and returned 20 datasets, out of which three had an open license and also exposed a SPARQL endpoint. Among these, LD4S was actually the only accessible source (as in Chapter 5, Section 5.7), so the average response time for SPARQL queries we execute is actually referring only to queries running on LD4S. Listing 6.1 shows an extract of one of these queries (the namespace prefixes have been omitted for clarity) which is used to

retrieve sensors within a specific time range and near certain location coordinates. LD4S provided a response to the query in Listing 6.1 within 246 milliseconds.

```
SELECT ?sens ?starttime ?endtime ?obs
?foi ?value ?location
{ ?sens spt:obs ?obs.
 ?ov spt:outOf ?sens;
  spt:value ?value;
  spt:tStart ?starttime;
  spt:tEnd ?endtime.
 ?tsp spt:temporalOf ?sens;
  ssn:featureOfInterest ?foi;
  wgs:lat ?latitude;
  wgs:long ?longitude.
FILTER (
xsd:dateTime(?starttime) >=
  '2014-11-30T02:00:00Z'
  ^^xsd:dateTime
  [..]
 && xsd:double(?latitude) <=
  '53.2692120'^^xsd:double)
  [..].
}
```

Listing 6.1: SPARQL query targeting sensor data in a time and location range.

6.3. Discussion & Roadmap

Our experimental results demonstrate that: (a) websites about or referring to real-world locations are a common phenomenon in urban areas; (b) the performance of G-Sensing does not impede on a user's browsing experience in terms of the average response time and additional bandwidth overhead. At present, we also note that there is limited availability of sensor datasets that are both open and public, while some of the SPARQL endpoints for such open and public datasets were inaccessible. However, we expect a larger number of accessible datasets in the future, because of the constant growth of the Linked Open Data cloud, the Internet of Things uptake and because providing public data via a standardized access mechanism is still quite a recent trend.

Beyond search result pages. For the time being, we are showing live data alongside Google search results. Our add-on-based approach can, in practice, allow us to inject

sensor information into any website. For example, we could display information about the parking situation around a restaurant on the restaurant's official website. With this, users can benefit from live data during normal browsing sessions, i.e., while navigating from website to website without relying on explicit search requests.

Extended linkage. So far, we have linked Web content with geolocations using public data crawled from Google Places, where many locations are associated with a URL (typically the websites of hotels, restaurants, shops, etc.). In the next step, we aim to extend these connections by injecting sensor information into other Web pages that also refer to such venues, as explained in Section 6.5. For example, we are currently extending our data repository by crawling user reviews from TripAdvisor. Linking the review URLs to geolocations using our existing Google Places data will enable us to show relevant live data to users who are reading reviews on TripAdvisor. In the long run, we will explore which types of connections are meaningful in a given application context and how such connections can be established. For example, we envision displaying the latest webcam feeds showing a location that is mentioned in a news article. Creating such links in an automatic and reliable manner is a challenging task.

User-centric live data representation. The G-Sensing output relevancy for an end user depends on the user's current interests and on the type (and low level location) of sensor data displayed, e.g., sensor data about occupancy in the clinic waiting room or in the surrounding parking area, rather than sensor data about the temperature of the fridge in the clinic's kitchen. Future releases of our system will include a recommender system that decides whether to display or not the retrieved sensor data according to a prediction of their current relevancy for the user.

6.4. Related Work

Map interface for sensors. In line with the efforts to make sensors accessible from the Web, several projects have focused on overcoming sensor network heterogeneity. They usually create an abstraction layer and visualize sensors on world maps. Microsoft's SensorMap [Nath et al., 2007] mashes up sensor data from a worldwide heterogeneous sensor network (SenseWeb) on a map interface and provides interactive tools to selectively query sensors and visualize data, along with authenticated access to manage sensors. This was followed by similar efforts since then, with GraphOfThings [Phuoc et al., 2014] being the most recent one. We also rely on latitude and longitude coordinates to locate a

sensor device, we provide an alternative perspective. Rather than representing the device location on a map we represent it in relation to its already existing virtual representation in the form of websites. Also, by injecting our system into GOOGLE search results, we actually make sensors more accessible to most Web users.

Virtual versus Real Spaces. The difference between real and virtual places has been analysed in several research areas including philosophy [Kolb, 2008], e-learning [Sutcliffe and Alrayes, 2012], augmented reality [Kalkofen et al., 2013], collaborative software development [Lucia et al., 2008], social networking and communities [Powazek, 2001], etc. More recently, von der Weth et al. [von der Weth and Hauswirth, 2013] proposed a scientific foundation for the problem of mapping the physical presence of people in real places to the online presences of users in virtual places (on websites). Similar to this work, they propose enriching virtual places with real information. However, they rely on users to act as sensors, providing information about those real places in a chat-type browser extension. This research is the closest so far to our approach. We build on top of it, investigating how to exploit sensor devices as opposed to human beings for providing virtual content enriched with live data.

6.5. Conclusions

In this chapter, we have proposed an approach to bridge the gap between the semi-static content offered by websites and the short-lived information offered by sensors. This represents our Contribution **C 4**: improving the data quality for traditional Web content. We were motivated by the Research Question **Q 4. Quality**: *How can contextualised sensors improve the quality of traditional Web content?* While the majority of sensor-related research goes in the direction of dedicated platforms through which users can explore available live data, we aim towards the seamless integration of sensor information into users' everyday browsing behaviours. As our main contribution, G-SENSING offers live data tailored to the information needs of online users of search engines. We facilitate this by providing a browser add-on which injects enriched sensor information into Google search results pages. At the same time, we keep our data source as open and flexible as possible, and approach the sensor data integration problem in innovative ways. Our backend infrastructure exposes a GUI, RESTful API and SPARQL endpoint to enable the annotation, storage and retrieval of semantic sensor data. Our evaluation showed both the potential benefits of G-SENSING and its applicability for large-scale settings.

In ongoing and future work, we focus on two main directions. Firstly, we aim to adapt the live data displayed according to a user's current context – that is, we would expose, for example, different types of sensor data to a user commuting on a bus and a user sitting at home. This requires appropriate context-modelling techniques as well as extending both the add-on and backend infrastructure to support context-dependent content delivery and presentation. Secondly, we want to extend our browser extension to inject live data into any relevant Web page referencing a physical location (e.g., the official websites of hotels, restaurants, businesses) and not just into GOOGLE result pages.

Part IV.

Conclusion

Chapter 7.

Conclusion and Outlook

LD4S is a framework for semantic interlinking of sensor data. We base our work on LD4S, using the building blocks it provides — the foundations of data representation and the layered service oriented architecture — to enhance it further, through good user-facing semantic applications, and to connect it to the large source of Linked Data that is the Web.

The approach we took in this thesis is data-centric. We focused on maintaining and enriching the network of linked sensor data that LD4S enables. The path we took is three-fold, with each direction working towards exploiting the power of linked semantic sensor data. The first direction is internal, on LD4S, through the means of new semantic tools, designed to support, and even more, encourage interlinking. The second direction is external, connecting the Activity Logging tasks, the Web search results and the Sensor Networks to the linked sensor data modelling we proposed. Both directions lead to IoT systems able of automated configuration.

This chapter summarises the work presented in this thesis, reiterating the contributions and presenting a general discussion and insights gathered. We conclude with a list of open questions and directions for future research and a final summary of the work.

7.1. Contributions

This thesis presents five main contributions, as mentioned in Section 1.3. **1.** The first contribution sets the scene for the rest of the work, by creating a model and a provisioning service for linked sensor data, in Chapter 4. This work differentiates from the state of the art in its support for social feedback and sharing and for interlinking customisation.

We identified which information to collect, modelled it using ontologies and process it as OWL. The rest of the work focuses on exploiting the advantages brought by our approach to semantically annotate, model and interlink sensor data, for the purposes of IoT system responsiveness, adaptiveness and merging with the Web of Documents.

2. The second contribution, interlinking with contextual information, looks at the method to search for external data eligible to be linked and how to evaluate the linking in Chapter 4. Since part of the metadata can be context-related, we defined and implemented an algorithm which finds external data to link based on client-defined criteria. Such service is also provisioned by LD4S. We also enabled the evaluation and rating of the created link, embedding comments and rates within the data annotation. In this way, simple reasoning over OWL could filter data based on ratings.

3. The third contribution consists of filtering of relevant contextual information for a given task, in Chapter 5. We completely automate the relevancy-based selection and evaluation of contextual information performed during the interlinking phase. Because relevancy is highly dependent on the task at hand, we focus on a specific task: daily activity logging. We designed and implemented an algorithm that predicts which sensors are going to provide relevant data in the close future for the activity that is currently being logged. The algorithm uses distributional semantics and clustering techniques. We compared the sensors per-activity groups that resulted from our system with publicly available manually annotated data, achieving outstanding results.

4. The fourth contribution is about improving the data quality for traditional Web content, in Chapter 6. We focus on bridging the gap between Sensor Web and traditional Web in which the latter lacks of short-lived but extremely interesting information provided to the average user. Relying on LD4S as a backend, we developed *G-Sensing* as a frontend, a browser plugin that injects live data from sensors into Google search results. We successfully evaluated its performances and suitability of our approach.

7.2. Directions for Future Research

We have presented in this thesis, directions to create context-aware linked sensor data and approaches to exploit their potential. The purpose of enabling and encouraging semantic annotation and interlinking of sensor data is to both improve the sensor discovery to

enrich other kinds of content such as Web content, and to enable automated adaptation of IoT systems.

In Chapter 2, we described the challenges faced when deploying a mobile dynamic sensor network and exemplified potential solutions to them through LD4S design and development. However, we believe there is still much improvement to be made in the are of context-aware applications for the auto-configuration of IoT systems.

We plan to further investigate algorithms and methods to support:

- the filtering of links in linked sensor data according to the users rating that we collect with LD4S;
- a recommender system for filtering the sensor data that we inject in search results with G-Sensing according to the users' personal preferences and the situation in which they are at the moment;
- injection of sensor data into any type of Web page and Web content via G-Sensing beyond Google search results;
- applying reasoning over ontological concepts to derive the labelling of an activity during an activity logging task beyond the per-activity sensor grouping;
- collect users' feedback on the automatically derived semantic annotations as well, for the incremental learning of our system.

In Chapter 2 we described many IoT systems, and one of the recurring challenges is the communication between different sensor networks. We support the reasoning over semantic annotations in order to automatically map concepts used in different networks from distinct ontologies. However, automated reasoning in presence of distinct ontology would further facilitate both the network inter-communication and the final users. Extending LD4S to support such automated reasoning and visualising the results in its GUI will bring new interesting achievements. Since LD4S uses OWL, reasoning is already supported, as explained in Chapter 3, Section 3.2.2.

Finally, a third direction for the future is devising and running a long term, large scale user study, to gather insights into how users really use the current functionalities offered by their LD4S. We have started work in this direction (as explained in Chapter 4, Section 4.4) targeting only a small subset of users because we did not have a record of all the people who had downloaded and deployed LD4S (although numbers are available on GitHub). We plan to reach out to all the users and set up a procedure that collects the

user's email address prior to allowing the download, so to keep a complete user registry. We hope that such a study would help us focus our research on things which have the most impact on the way LD4S is used.

7.3. Summary

The main contributions of this Thesis focus on supporting interlinking of context-aware semantic sensor data and exploiting its advantages.

Conceptually, we present the challenges of designing context-aware adaptive IoT systems, and we discuss options and possible solutions. We also detail three algorithms for predicting sensor relevancy during activity logging tasks through unsupervised hierarchical clustering, for bridging the gap between the traditional Web of Documents and the Semantic Web of Things through cross sensor dataset searches and data injection, for automating the derivation of semantic annotation for devices newly entering a network.

From the implementation point of view, we support the conceptual contributions with corresponding software.

LD4S is a Web service exposing a RESTful API and GUI for creating, storing and browsing linked sensor data. In Chapter 4 we describe the design and implementation of LD4S, as an illustration of possible solution to the challenges found. This was then extended to support the prediction of semantic sensor annotations and of sensor relevancy (for activity logging tasks).

G-Sensing is a system that includes LD4S as a backend and a Mozilla Firefox plugin as a frontend which automatically finds the most recent sensor data that have been observed in the real places represented by their corresponding web pages in search results. In Chapter 6 we describe the algorithm as well as the implementation, which allows for various modes of utilisation, depending on the use case.

We evaluated both implementations and the results are positive:

- A task-based user evaluation of LD4S showed that although users had no prior experience with sensors they found the information clear and the system easy to use. With regards to the sensor relevancy prediction, we compared our unsupervised results with a golden standard of manual annotations on the same dataset. Our

results show an average precision of 89% and an overall accuracy of 69%, thus outperforming the state of the art by 5% and 32% respectively.

- The algorithm to inject sensor data on Web search results was evaluated in terms of suitability and performances. Our results show our algorithm has a broad coverage of places in the real world and its performances are suitable for Web widget or browser plugin.

Appendix A.

Personal Contribution to Publications

This appendix lists the specific thesis contributions made by the author with respect to every co-authored publication.

The ontology described in Section 4.2 and the LD4S web service providing automated semantic annotation and automated interlinking, described in Section 4.3 constituted the main contribution to the following co-authored papers:

- "Annotating Real-World Objects using Semantic Entities" [Hasemann et al., 2013]
- "Data Modeling for Cloud-Based Internet-of-Things Systems" [Leggieri et al., 2012b]
- "True Self-Configuration for the IoT" [Chatzigiannakis et al., 2012]
- "The SSN Ontology of the W3C Semantic Sensor Network Incubator Group" [Compton et al., 201
- "SPITFIRE: Towards a Semantic Web of Things" [Pfisterer et al., 2011b]
- "Unlocking Wireless Sensor Networks" [Richardson et al., 2011]
- "Semantic-Service Provisioning for the Internet of Things" [Pfisterer et al., 2011a]
- "inContext Sensing: LOD augmented sensor data" [Leggieri et al., 2011a]
- "A Contextualised Cognitive Perspective for Linked Sensor Data - Short paper" [Leggieri et al., 2010a]
- "Monitoring Urban Traffic using Semantic Web Services on Smartphones - A Case Study" [Kleine et al., 2015].

The author is the main contributor (i.e., author of content, implementations and evaluations) of the following SPITFIRE deliverables:

- Spitfire: "D2.1 Ontologies for representing sensor information" [Leggieri et al., 2012a]
- Spitfire: "D2.4 Social Feedback and Sharing" [Leggieri et al., 2013a].

The author is a substantial contributor (i.e., author of major portions of content, design, examples and images) of the following OpenIoT and Gambas deliverables:

- OpenIoT: "D3.1.2 Semantic Representations of Internet-Connected Objects" [Leggieri et al., 2013b]
- Gambas: "D4.1.1 RDF Sensor Formalisms and Ontologies" [Leggieri and Parreira, 2013].

The author is the main contributor (i.e., author of content, implementations and evaluations) of the following conference papers:

- "Using Sensors to Bridge the Gap between Real Places and their Web-Based Representations" [Leggieri et al., 2015b]. Here, the second author C. von der Weth, substantially contributed to the evaluation section.
- "Distributional Semantics and Unsupervised Clustering for Sensor Relevancy Prediction" [Leggieri et al., 2015a].

The author is the main contributor (i.e., author of content) of the following book chapter:

- "Interoperability of two RESTful protocols: HTTP and CoAP" [Leggieri and Hausenblas, 2014].

Appendix B.

SPITFIRE: Towards a Semantic Web of Things

Sensors are ubiquitous in infrastructures, appliances, mobile phones, and wireless sensor networks. Their widespread deployment represents a significant financial investment and technical achievement and the data they deliver is capable of supporting an almost unlimited set of high value proposition applications. This is a powerful and profitable confluence of need, capability, and economic opportunity – yet the true potential of sensor technology is massively under-exploited.

A central problem hampering success is that sensors are typically locked into unimodal closed systems. For example, motion detection sensors in a building may be exclusively controlled by the intrusion detection system. Yet the information they provide could be used by many other applications, e.g., placing empty buildings into an energy-conserving sleep mode or locating empty meeting rooms. Unlocking valuable sensor data from closed systems has the potential to revolutionise how we live. To realise this potential, a service infrastructure is needed to connect sensors to the Internet and publish their output in well-understood, machine-processable formats on the Web thus making them accessible and usable at large scale under controlled access.

So far, the sensor world and the Web world have been largely disconnected, requiring the human in the loop to find, integrate and use information and services from both worlds in a meaningful way. Publishing sensor-related data on the Web would help to find relevant information by directly accessing sensor data, i.e., by directly observing the real world, integrated with related information from the Web. Already today, smart phone applications such as CenceMe [Miluzzo et al., 2008] exist that infer the activity of the person wearing the phone from sensor data and publish this in the Web. Another example

are energy consumption sensors that end-users can install in their house to measure energy consumption of appliances, for example to compare their energy consumption with that of other, similar households to identify opportunities for saving energy. To do this easily, with open interfaces and data formats, and at large scale, technologies from the Web need to be customised for and integrated with their relevant counterparts on the Internet of Things (IoT). This means that application experts who are able to publish Web pages today should have the same easy-to-use technologies at hand to publish sensor descriptions, sensor data and make use of sensor outputs without requiring deep knowledge of embedded computing. In particular, we believe that users are primarily interested in real-world entities (things, places, and people) and their high-level states (empty, free, sitting, walking, ...) rather than in individual sensors and their raw output data. Therefore, the infrastructure must provide appropriate abstractions to map sensors and their raw output to real-world entities and their status representation.

Real-world entities are rarely useful when considered in isolation – the ability to put multiple entities into a common semantic context is needed. For example, we want to reason about rooms being in the same building, belonging to the same company, with nearby parking spots. This requires a machine-readable representation of world knowledge and appropriate reasoning capabilities. Further, this representation needs to be unified - while most sensor data published so far on the Web relies on heterogeneous data models and serializations. In addition to discovery and query facilities on static properties of those machine-readable representations of sensors and real-world entities, specialized search approaches to support queries on the dynamically changing state of sensors or entities consisting of many sensors (possibly integrated with static data), will be required, e.g., which rooms in a building are currently occupied.

There are efforts to realize a Semantic Sensor Web including the SENSEI [Villalonga et al., 2010a], SemSorGrid4Env [Partners, 2011], Exalted [Partners, 2013], and 52 North projects [Partners, 2014], as well as work by the Kno.e.sis Center [Patni et al., 2010], CSIRO [Compton et al., 2009], and the Spanish Meteorological Agency [AEMET, 2014]. Most notably, the Open Geospatial Consortium's (OGC) Sensor Web Enablement (SWE) [OGC - Open Geospatial Consortium, 2010] project builds a framework to publish and access sensor data using XML-based protocols and APIs. The choice of XML, however, ties SWE to system-specific schemas, providing neither semantic interoperability nor a basis for reasoning. This problem is in the focus of the Semantic Sensor Web which proposes annotating sensor data with semantic meta-data, whose meaning is machine-understandable through vocabulary definitions, i.e., an ontology [Sheth et al., 2008]. By annotating sensor-related features such as the

network, deployment, data formats, etc., it becomes possible to automate further tasks, e.g., deployment, maintenance, and integration.

However, these efforts have limitations which we address in SPITFIRE: There is no general-purpose approach compatible with the growing body of semantic world knowledge available as Linked Open Data (LOD) on the Web; existing efforts are either too sensor-centric or too knowledge-centric, i.e., they do not provide comprehensive, integrated abstractions for things, their high-level states, and how they are linked to sensors; and a number of important services are missing in existing efforts, notably support for semi-automatically creating Linked Data representations of sensors and things, as well as efficient search for things based on their current states.

SPITFIRE [Pfisterer et al., 2011b] addresses these limitations by providing 1. vocabularies to integrate descriptions of sensors and things with the LOD cloud 2. semantic entities as an abstraction for things with high-level states inferred from embedded sensors 3. semi-automatic generation of semantic sensor descriptions 4. efficient search for sensors and things based on their current states. In addition, SPITFIRE integrates these ingredients into a unified service infrastructure to ease adoption of the Semantic Web of Things for end-users and developers. On top of this infrastructure, applications are assembled by issuing search requests for matching (real or aggregated) sensor services and by invoking found services directly.

This section proceeds with the description of an exemplary use case that will be used to illustrate the state of the art with respect to integration of sensors into the Web upon which SPITFIRE builds, followed by a description of the novel contributions of SPITFIRE and a brief discussion of an existing operational prototype.

Use Case and Requirements Due to an emergency, a travelling salesman drives to his company headquarters to hold an ad-hoc meeting. For that, he must find a currently free room in the headquarters that are dispersed over a large area in order to hold an ad hoc meeting. After that, he informs his colleagues and searches for a parking spot close to the building.

Imagine that sensors, which are connected to the Internet, measure the state of real-world entities such as meeting rooms and parking spots. Internet-connectivity not only requires network-level integration (IP), but also application-level integration to enable structured access to sensor data. To enable automatic reasoning about sensors (e.g., finding free parking spots close to meeting room), these sensors, their output, and their

embedding into the real world must be *described in a machine-readable format* that is compatible with data formats used to describe existing world knowledge in the Web. Not only syntax and semantics of such a description must be defined, but efficient mechanisms to *annotate newly deployed sensors* with appropriate descriptions are required.

Users are primarily interested in *real-world entities* (e.g., meeting room) and their *high-level states* (e.g., room occupied) rather than sensors (e.g., sensor 536) and their raw output (e.g., motion detected at time T). Therefore, appropriate mechanisms to establish an explicit mapping of sets of sensors to real-world entities they are monitoring (e.g., all motion detection sensors in a certain room) must be provided. Further, the raw output of these sensors (e.g., motion detection events) must be mapped to a high-level state (e.g., room occupied). Often, this involves fusing the output of multiple sensors (e.g., multiple motion sensors are needed to cover a large room) or even scheduling sensors for energy efficiency (e.g., only one out of two available battery-powered motion sensors is required to cover a smaller room).

Finally, the user wants to *search for real-world entities by their current state* (e.g., empty meeting rooms). Often, such search requests refer not only to the output of sensors, but also to further machine-readable information that is available elsewhere in the Web (e.g., company maps, meeting schedules, calendars). The search engine needs to integrate these different static and dynamic data sources in a seamless way.

Realizing the above use case on an Internet scale requires

- that the sensors are connected to the Internet,
- that machines can discover and understand the semantics of the data returned by the sensors, and
- a technique to find the sensors that could provide the relevant data.

This section briefly discusses the state-of-the-art in relation to this with a focus on Internet-scale, Web-based technologies upon which we build our approach, employing the use case as an example.

Connecting Sensors to the Internet and the Web Integrating resource-constrained sensors into the Internet is difficult since ubiquitously deployed Internet protocols such as HTTP, TCP or even IP are too complex and resource-demanding. To achieve integration,

light-weight alternatives are required that can easily be converted from/to Internet protocols.

Only recently, two such alternatives are gaining momentum: 6LoWPAN and CoAP. 6LoWPAN [Montenegro et al., 2007] is a light-weight IPv6 adaptation layer allowing sensors to exchange IPv6 packets with the Internet. Currently, only UDP is specified as TCP is considered too resource consuming. CoAP (Constrained Application Protocol [Frank et al., 2012]) is a draft by IETF's CoRE working group, which deals with Constrained RESTful Environments. It provides a light-weight alternative to HTTP using a binary representation and a subset of HTTP's methods (GET, PUT, POST, and DELETE). In addition, CoAP provides some transport reliability using acknowledgements and retransmissions. For a seamless integration, reverse proxies may convert 6LoWPAN and CoAP to TCP and HTTP so that sensor data can be accessed using these omnipresent protocols. Also, Internet-based clients could directly use CoAP on top of UDP.

6LoWPAN in combination with CoAP allows sensors to be queried from the Internet as they can provide so-called RESTful web services. Those are services following the Web's REST (REpresentational State Transfer) principles [Fielding, 2000]. Resources (e.g., sensors) are addressed using standard URIs and data can be returned in different representations (e.g., HTML or RDF) using HTTP content negotiation.

RESTful services are queried and manipulated using the aforementioned four HTTP methods. For instance, an application could query the state of a sensor by sending a GET request to the sensor (e.g., `http://ipv6-address-or-dns-name/room-sensor`. The sensor replies with its value encoded in a, possibly proprietary, encoding (e.g., in plain text: "*occupied*" or any format the sensor supports). For an exhaustive discussion of 6LoWPAN, CoAP, and RESTful services, we refer the reader to [Shelby, 2010].

To discover the services hosted on a CoAP server, the CoRE Link Format specification defines how Web Linking described in RFC5988 is used by CoAP servers. Clients use a well-known URI (`/.well-known/core`) to retrieve a list of resources. For instance, the room sensor device could return *</room-sensor>;ct=0;rt="ex:RoomSensor"* to indicate that the resource `/room-sensor` returns the content type *text/plain* (indicated by $ct = 0$) and that the resource type is *ex:RoomSensor*. The latter is a concept from an ontology (e.g., the W3C SSN-XG sensor ontology (to which we contributed [Compton et al., 2012b])

as described in the following section. Note that concepts appearing in different ontologies can be automatically mapped[1].

RESTful services (i.e., the operations provided, their parameters and return values) can be described using, for example, Web Service Definition Language (WSDL) version 2.0. For example, RESTful versions of OGC's Sensor Observation Services have been proposed and are currently under consideration by the Sensor Web Enablement group.

Linked Sensor Data The integration of sensors into the Internet using CoAP/HTTP already enables many applications in which developers query and process data provided by a well-known set of sensors. However, such manual integration does not scale. What is required is a "machine-understandable" description of sensors and the data they produce. Semantic Web [Lee et al., 2001] technologies fulfil this requirement as they enable machines to understand, process, and interlink data using structured descriptions of resource and Linked Open Data as the framework makes this integration both immediate and meaningful through the inclusion of semantic links into a resource's machine-readable description.

The predominant technique for *machine-readable representations of knowledge* on the Web is the Resource Description Framework (RDF), which represents knowledge as *(subject, predicate, object)*-triples (e.g., *Sensor3 is-in ParkingSpot41* or *ParkingSpot41 is-in Berlin*). A set of triples forms a graph where subjects and objects are vertices and predicates are edges. From the graph formed by these two triples, one can infer that *Sensor3* is in Berlin by exploiting the knowledge (contained in so-called ontologies), then *is-in* is a transitive property. Such knowledge is often expressed using OWL (Web Ontology Language), one of the main languages (with RDF Schema) to define ontologies on the Web.

It is imperative to use non-ambiguous identifiers for subjects, predicates, and objects to guarantee uniqueness on an Internet-scale, which is achieved by encoding them as URIs. The above triple could be expressed in a graph like the one in Figure 3.2 where the following URIs may be used as subject, predicate and object:

- a subject (<http://example.com/sensors/sensor3>),
- a predicate (<http://www.loa-cnr.it/ontologies/DUL.owl#hasLocation>), and
- an object (<http://example.com/parkingSpot/spot41>).

[1] http://www.w3.org/DesignIssues/RDF-XML

The Linked Data model does not enforce special URIs but encourages the use of widely-used URIs so that a densely interlinked graph emerges. *Ontologies* play an important role in defining the URIs for a specific application domain and their relation to each other as they "standardize" agreed, conceptual knowledge. For example, an ontology could define a generic *sensor* (e.g., `http://purl.oclc.org/NET/ssnx/ssn#Sensor`), an occupancy detection sensor (e.g., `http://example.com/ontology/spitfire.owl#Occupancy`), and define that occupancy sensor is a sub-class of sensor, which creates a relation between the two URIs. Semantic search engines queried for *sensors* at a certain location could therefore specifically return information on *occupancy detection sensors*. The CoAP link-format we use (RFC5988) allows to specify URIs eventually pointing to semantic definitions, i.e., support for semantic annotation of links inside a sensor network. Additionally, to make these semantic descriptions available on the Web, we could imagine to annotate pages describing sensors using RDFa or SA-REST, so that the same document is used for humans and machines.

Search for Sensors Assuming that sensors are described by such RDF triples, a search service can find sensors based on meta-data such as sensor type, location, or accuracy. For instance, applications could ask for parking spots in Berlin to calculate the city's availability of car parking places. Such queries can be expressed in SPARQL and the aforementioned question could be answered using the (simplified) SPARQL query in Listing B.1. In the query, question marks indicate variables (e.g., "node" and "spot"), while "spots" is an aggregate value.

Figure B.1.: SPARQL query requesting all occupancy sensors located at parking spots in Berlin.

The variables in a SPARQL query are matched against triples in databases (triple stores) and are bound to the matching fields in the matching triples. That is, the query finds subjects that are sensors observing occupancy that are located in a spot that is a parking spot located in Berlin. There are a number of existing efforts to support semantic sensor discovery but they are not as comprehensive as us. For instance, the authors in [Jirka et al., 2009b] do not expose Linked Data while the authors in [Pschorr et al., 2010] do not exploit the hierarchical and structured relations which are relevant even for such simple queries as above. To further exploit these annotations, we could also use faceted browsers such as MIT Simile's Exhibit, where facets for identi-

fying parking places could be location, availability, but also static information such as price-range.

Appendix C.

LD4S – Event Model-F Ontology

The explicit modelling of events and event-based systems are increasingly gaining widespread attention by research and industry. The detection of an event is the most intuitive interpretation a human could give to a sensor reading. In fact, intelligence-collecting devices like sensors, lead to an ubiquity of events being recognized and communicated and, thus, they require multiple systems to be connected for managing events. This results in complex and distributed event-based systems, characterized by taking events as input and providing events as output.

Motivation: Domain events may be very complex and may be linked to a variety of aspects such as time and space, objects and persons involved, as well as structural relationships like mereological, causal, and correlate relationships. Different event-based components and systems can hardly be integrated or communicate with each other, because of the ad-hoc, idiosyncratic event models that they use, whose semantic interpretation becomes a challenging task. Since the existing event models were all developed ad-hoc and lacking formal semantics, the Event Model-F ontology was realized [Scherp et al., 2009a].

The purpose then, is to provide a formal representation of events in a model that allows easy interchange of event information between different event-based components and systems.

Overview: The Event Model-F ontology is based on the upper level ontology Dolce+DnS Ultralite, and provides comprehensive support for both all the structural aspects of events, e.g., mereological, causal, and correlational relationships, and non structural ones, e.g.,

time and space, objects and persons involved. The ontology provides flexible means for both event composition, modelling of event causality and correlation. It also enables representing different interpretations of the same event.

The foundational event model F is developed in a pattern-oriented approach, modularized in different ontologies, and can be easily extended by domain specific ontologies.

It also covers all the requirements that a common event model should support:

- Constitutive Aspect: The constitutive aspect describes the living and non- living objects participating in an event such as people, animals, and other material objects.
- Temporal Aspect: The temporal aspect covers the temporal extension of an event. It can be modelled using absolute or relative representations of time.
- Spatial Aspect: The spatial aspect is in charge of capturing the spatial dimension of objects participating in the event. This can be also modelled using absolute or relative positioning.
- Experiential Aspect: The experiential aspect comprises the annotation of events with sensor data such as media data.
- Structural Aspect: The structural aspect considers the arrangement of events in mereological, causal, and correlative relationships. Events may be and usually are made up of other events. Thus, the common event model shall support the modelling of mereological relationships between events. Causality requires the modelling of causes and effects, and should support the integration and use of different causal theories. Correlation refers to two events that have a common cause. Causality is very difficult to discover and, hence, often unknown; while correlation is typically easy to observe.
- Event Interpretations: Structural relations between events such as causality and correlation can be a matter of subjectivity and interpretation. For example, in a law-suit the parties involved may each claim that the other one is at fault. A common event model should be prepared to support such different interpretations of the same event.

We chose to align our ontology to it, because of this very completeness and detailed modularization. Moreover it includes the Description and Situation (DnS) Ontology Design Pattern, that we were interested in using in our ontology, since we decided to align with Dolce+DnS Ultralite which is based on it.

Main Patterns The formal representation of the experiential aspect, is already possible using existing approaches, e.g., the Core Ontology for Multimedia

Established ontologies for time and space, and others can be used to support the spatial and temporal aspects of the dul-aligned ontology modules for temporal relations and spatial relations.

The remaining aspects and the requirement for event interpretation are represented by specialized instantiations of the DnS ontology pattern. In the following, we explain the ontology patterns of the Event Model F including graphical illustrations of them. Classes defined by the Event Model F are highlighted to show the alignment with classes of DUL that are drawn with white background.

The Participation Pattern

One aspect of an event is given by the objects participating in an event such as persons. The participation pattern of the event model F enables to express this constitutive aspect of events formally. As shown in Figure C.1, participation is expressed by an *event: Event-Participation-Situation* that satisfies an *event: Event-Participation-Description*. The situation includes the *dul: Event* being described and the *dul: Objects* being participants of this event. The *event: Event-Participation-Description* classifies the described event and its participants by the concepts *event: Described-Event* and *event: Participant*.

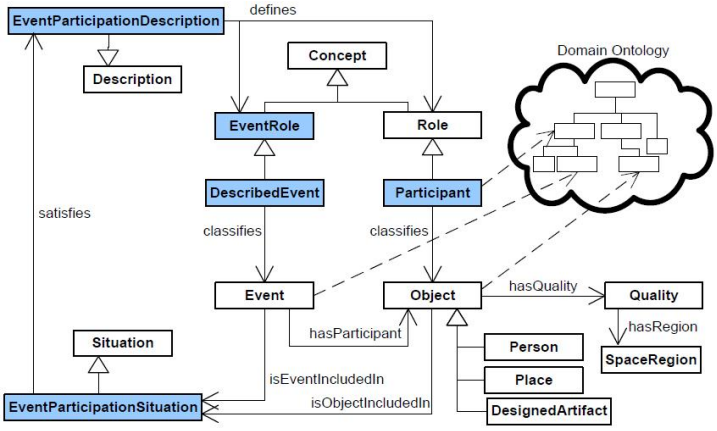


Figure C.1.: Participation Pattern [Scherp et al., 2009a] in which the Event Model F ontology is aligned with the Dolce+DnS Ultralite.

For instance, some domain ontology may define the role of a person being affected by some emergency case, i.e., the emergency subject, and the role describing the rescue staff

such as firemen. As indicated by Figure C.1, the described event, participating objects, and their roles can be defined in some domain ontology so that the participation pattern can be applied to express participation, with respect to arbitrary application domains, e.g., emergency response.

(De-)Composition Pattern

Events are commonly considered at different abstraction levels depending on the view and the knowledge of a spectator. For instance, the local event of a flooded toilet room, may be considered as such or as part of a larger event of a University building flooding in which many such (smaller) incidents occur. The composition pattern enables to express such relations as the composition of events. Here, the composite event is the *whole* and the component events are its *parts*. Formally, an *event: Event-Composition-Situation* includes one instance of an event that has the *event: Event-Role* of an *event: Composite* event and one or many events considered as *event: Component*s of that event, as shown in Figure C.2. Accordingly, an *event: Event-Composition-Situation* satisfies a *event: Composition-Description* that defines the concepts *event: Composite* and *event: Component* for classifying the composite event and its component events.

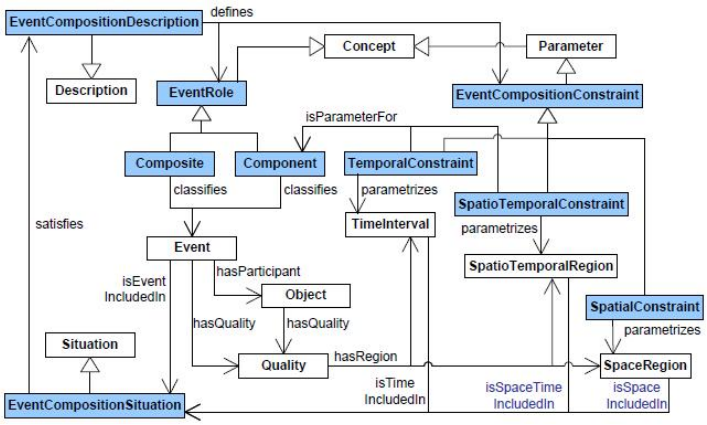


Figure C.2.: Composition Pattern [Scherp et al., 2009a] in which the Event Model F ontology is aligned with the Dolce+DnS Ultralite.

Causality Pattern

Causality is the traditional philosophical problem investigating whether any special tie that binds causes and effects together, exists or not. The pattern defines two *event:*

Event-Roles called *event: Cause* and *event: Effect* which classify *dul: Event*s, as shown in Figure C.3.

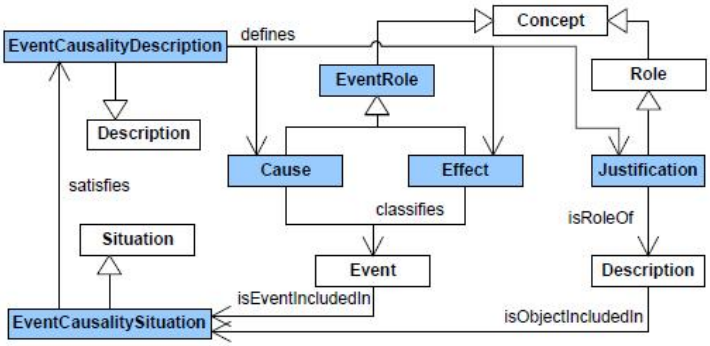


Figure C.3.: Causality Pattern [Scherp et al., 2009a] in which the Event Model F ontology is aligned with the Dolce+DnS Ultralite.

As defined above, causes and effects are events. Thus, we assume that objects inherently involved in the causal relationships are properly associated to the cause and effect by using the participation pattern from Section C.

Correlation Pattern

We call a set of events correlated, if they occur at the same time (or share some overlap) and have a common cause. However, there exists no causal relationship between the two events. The common cause may origin from a single or a chain of multiple preceding cause-effect relationships. Correlation also differs from co-occurrence where two or more events just randomly happen at the same time, but do not have a common cause. Figure C.4 illustrates the Correlation pattern, in which the role *event: Correlate* classifies the events that are correlated.

Achievement by Examples

In order to show the advantages that can be achieved thanks to the alignment of this ontology with Dul and then, indirectly, with our ontology, we can continue the example of Section D.1. In this scenario Henning was dealing with a flood in the male toilet on the first floor of the TUBS building.

If an event detection system mines the cause of this flood being a clogged pipe, then we can semantically describe this finding as shown in Listing C.1.

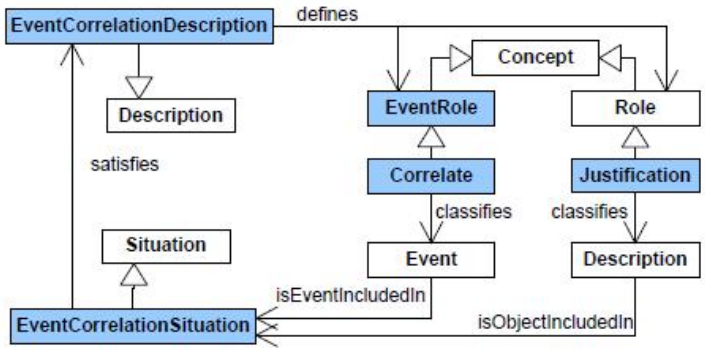


Figure C.4.: Correlation Pattern [Scherp et al., 2009a] in which the Event Model F ontology is aligned with the Dolce+DnS Ultralite.

```
:ToiletFloodEvent a spt:Activity ;
   a ssn:Observation .
:Blockage rdfs:subClassOf event:EventRole .
:CloggedPipe rdfs:subClassOf :Blockage .
:FloodEffect rdfs:subClassOf :Blockage .

:SPITFIRECloggedPipe a :CloggedPipe ;
   dul:classifies :ToiletFloodEvent .
:SPITFIREFlood a :FloodEffect ;
   dul:classifies :ToiletFloodEvent .
spt:Activity rdfs:subClassOf dul:Event .
```

Listing C.1: Example of using the Event Model F ontology to describe the cause of a sensor-detected event.

Afterwards, Henning wants to detect eventual subsequent events that might have been found correlated to this flood. Then he just needs to run a SPARQL query, as shown in Listing C.2.

```
SELECT ?events
 WHERE{
  ?events a dul:Event .
  ?type rdfs:subClassOf :Blockage .
  ?correlation a ?type ;
  dul:classifies ?events .
}
```

Listing C.2: SPARQL query that selects all the events that are correlated with the one happened in the first floor toilet, because sharing the same blockage kind of causality. If the event detector system has lead to create a correlation concept, then it means that these events also happened at the same time.

Appendix D.

LD4S – Semantic Sensor Network Ontology

Our ontology [Leggieri et al., 2010a] targets four main objectives:

1. Describe sensors and sensor-related data.
2. Support efficiency, which is critical in networks of constrained resource devices, in terms of in-network energy saving.
3. Support future interoperability with other ontologies.
4. Support the extension of human awareness about reality.

Here follows the design decision undertaken in order to achieve each one of the objectives listed above, respectively:

1. Alignment with a robust, cross-domain and comprehensive Sensor and Stimuli ontology, i.e., the W3C Semantic Sensor Network (SSN) ontology (Section D.1), whose realization we also contributed.
2. Extending the SSN ontology with specific concepts that are necessary for our purposes, especially in relation with our use case (see Section 4.2.1).
3. Alignment with an upper-level ontology, i.e., Dolce+DnS Ultralite (see Appendix E).
4. Alignment with an event ontology, i.e., Event Model-F (see Appendix C).

Then our ontology called the *SPITFIRE* ontology, has been developed using *Inheritance* (Section 4.2.3) from other ontologies and though we added new concepts, no inconsistency was introduced. Indeed, the new concepts have been *aligned* (Section 4.2.3)

with the other ontologies. The source code of the SPITFIRE ontology is available in Appendix F.

In this section we start describe the core part of our vocabulary, which is the SSN ontology. Then we describe each module and concept of our SPITFIRE ontology, followed by a description of the upper-level ontology and our alignment with it. Finally we will describe the Event Model F and how it also aligns to the same upper ontology, witnessing a real demonstration of how much this alignment simplify interoperability.

Each section ends with practical examples meant to show the actual achievements gained by the introduction of the specific ontology, in our semantic description.

D.1. Ontology for Sensors

Representing sensor data on the Semantic Web, requires all aspects of sensors to be described, i.e., capabilities, physical properties, observations, network characteristics, etc. To overcome common limits of pre-existing XML-based formats [OGC - Open Geospatial Consortium, 2010] and the fragmentation of sensor ontologies into specific domains or applications, the W3C Semantic Sensor Network Incubator Group (SSN-XG) developed a semantic sensor network ontology [Compton et al., 2012b]. We decided to use it as the basis for the SPITFIRE ontology, motivated by the reasons that follow below.

- Completeness: All the basic aspects of sensor-related and sensor data are taken into consideration, and the ontology allows the user to further describe them by integrating external ontologies.
- Alignment with Dolce+DnS Ultralite [Gangemi, 2010]: Ontology alignment with foundational ontologies ensures robustness of the ontology hierarchy structure and supports future interoperability with other ontologies.
- Likeliness to be integrated by other domain-specific external ontologies, and subsequently to make the integration process easier.
- Community within W3C: Potential further standardization opportunities and practical impact.

This ontology describes sensors and observations, and related concepts. It does not describe domain concepts, time, locations, etc. these are intended to be included from

other ontologies via OWL imports. The concepts and structure of the ontology were discussed in the group's meetings and on the mailing list, which were attended by several people from DERI, so that we actively contributed to the development of the SSN-XG ontology.

Objectives The SSN-XG worked on two main objectives:

1. the development of ontologies for describing sensors,
2. the extension of the Sensor Model Language (SensorML), one of the four Sensor Web Enablement (SWE) languages [OGC - Open Geospatial Consortium, 2010], to support semantic annotations. The SWE project is part of the Open Geospatial Consortium (OGC) activity, towards enabling web access to sensors.

The first objective, ontologies for sensors, provides a framework for describing sensors. These ontologies allow classification and reasoning on the capabilities and measurements of sensors, the provenance of measurements and the connection of a number of sensors as a macroinstrument. Following W3C recommendation, OWL 2 DL (Section 3.2.2) is the selected language for ontology specification. The sensor ontologies, to some degree, reflect the OGC standards and, given ontologies that can encode sensor descriptions, mapping between the ontologies and OGC models is an important topic addressed by the SSN-XG. The second objective, of semantic annotation of sensor descriptions and services that support sensor data exchange and sensor network management, serves a similar purpose to that offered by the semantic annotation of Web services.

Motivation The creation of the SSN-XG Incubator Group and the definition of the two main objectives listed above, were motivated by several factors, listed below.

- The opportunity for several W3C member organizations working on Sensor Ontologies, Semantic Sensor Web and Semantic Sensor Networks applications to merge their research effort in this area,
- The recognition that the legacy mechanisms used to embed domain-specific vocabularies in several Sensor Web Enablement (SWE) standards developed by the OGC (SensorML, SWE common) should be replaced by approaches based on the semantic web languages developed by W3C, in particular OWL DL,

- The sentiment that the development of a Semantic Sensor Network ontology and of mechanisms to support semantic annotations could improve interoperability and integration of the services using these standards, as well as facilitate reasoning, classification and other types of assurance and automation not included in the OGC standards.

Alignment A proposal to align the SSN ontology with the Dolce Ultralite (dul) upper ontology was made, on the basis of some preliminary alignment work done by one of the group participants, using a the Stimulus-Sensor-Observation (SSO) Ontology Design Pattern. The rationale behind this proposal was to facilitate reuse and interoperability. In fact, while the initially developed SSN-XG ontology can already be used as vocabulary for some use cases, other application areas require a more rigid conceptualization to support semantic interoperability. Therefore, we introduce a realization of the pattern based on the classes and relations provided by Dolce Ultralite. This ontology can be either directly used, e.g., for Linked Sensor Data, or integrated into more complex ontologies as a common ground for alignment, matching, translation, or interoperability in general.

The work done by the XG on these objectives, is presented in the next sections. During the course of the XG, the group also identified four principal classes of use cases that helped to prioritize parts of the ontology for development. One of these use cases is presented in Section D.1. The Semantic Sensor Network ontology revolves around the central Stimulus-Sensor-Observation pattern, that is presented in Section D.1. Before describing this main pattern and all the useful use cases enabled by the SSN-XG ontology, we depict an overview of its development in Section D.1.

Overview

The Group, recognizing the interoperability and broader applicability benefits of a collaborative effort, has developed a formal OWL DL ontology for modelling sensor devices (and their capabilities), systems and processes. The development was informed by a thorough review of previous sensor ontologies (included in this report), and the concurrent development of an informal vocabulary of the main terms, drawing on earlier vocabularies like the OGC/SWE ones e.g., Sensor Model Language (SensorML) and Observation and Measurement (O&M).

The ontology is based around concepts of *Systems*, *Processes* and *Observations*. It supports the description of the physical and processing structure of sensors. Sensors are not constrained to physical sensing devices: rather a sensor is anything that can estimate or calculate the value of a phenomenon. Then either a device or computational process or a combination of them, could play the role of a sensor. The representation of a sensor in the ontology links together what it measures (the domain phenomena), the physical sensor (the device) and its functions and processing (the models).

The ontology is available as a single OWL file: SSN ontology and a semi-automatically generated documentation derived from it is also provided as a standalone document. Additional annotations have been added to split the ontology into thematic "modules" which are introduced in the following paragraphs.

Modules: Figure D.1 shows the several conceptual modules that have been built on top of the Stimulus-Sensor-Observation pattern, to cover key sensor concepts.

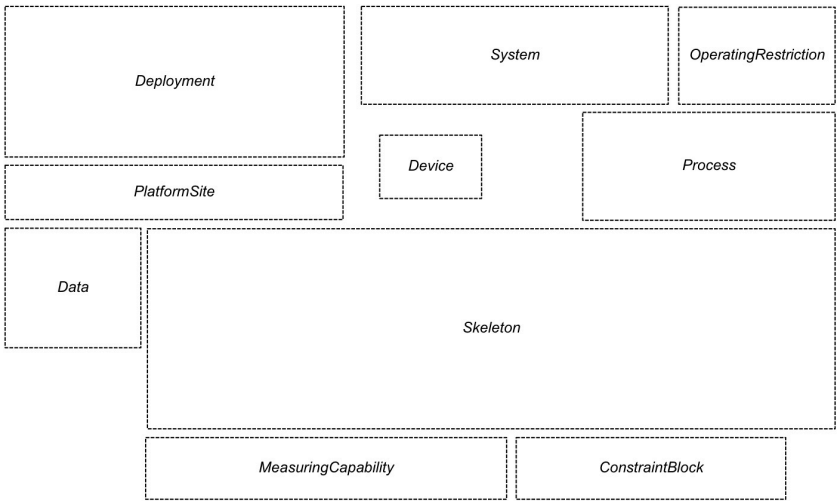


Figure D.1.: Overview of the Semantic Sensor Network ontology modules

Relationships between them, are illustrated in Figure D.2, which contains an overview of the main classes and properties inside the ontology modules.

The ontology can be used for a focus on, either any or a combination of, a number of perspectives:

- A sensor perspective, with a focus on what senses, how it senses, and what is sensed;
- A data or observation perspective, with a focus on observations and related metadata;

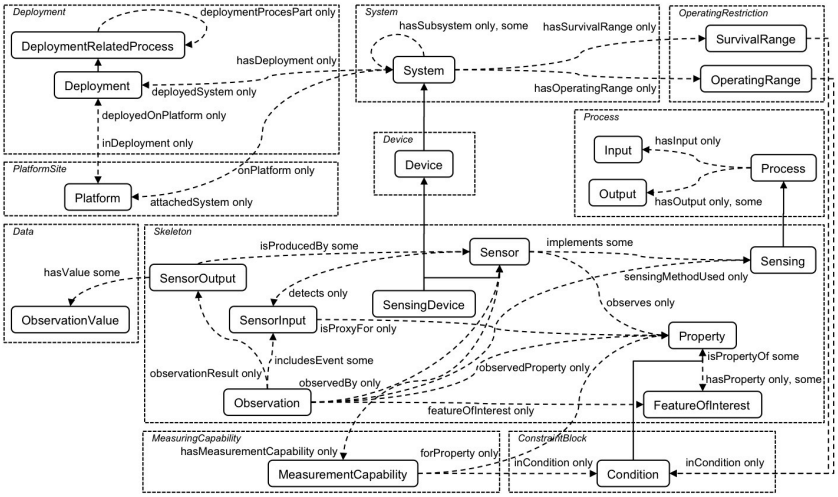


Figure D.2.: Overview of the Semantic Sensor Network ontology classes and properties [Compton et al., 2012b].

- A system perspective, with a focus on systems of sensors; or,
- A feature and property perspective, with a focus on features, properties of them, and what can sense those properties.

The modules as described here allow further refining or grouping of these views on sensors and sensing. The description of sensors may be detailed or abstract. The ontology does not include a hierarchy of sensor types; these definitions are left for domain experts, and for example could be a simple hierarchy or a more complex set of definitions based on the workings of the sensors.

The Skeleton module

The relation between the three ontologies (the SSN ontology contained at the end of the first phase, the core skeleton and the dul-aligned version) is best thought of as layers or modules. The core *Skeleton* module (also referred to as ontology design pattern) represents the initial conceptualization as a lightweight, minimalistic, and flexible ontology with a minimum ontological commitment. It is built around the *Stimulus-Sensor-Observation* (SSO) Ontology Design Pattern, illustrated in Figure D.3, which aims at all kind of sensor or observation based ontologies and vocabularies for the Semantic Sensor Web [Sheth et al., 2008] and, especially, Linked Data [Berners-Lee, 2006a]. The pattern is developed by following the principle of minimal ontological commitments to make it reusable for a variety of application areas. It is not aligned to any other top-level

ontology and introduces a minimal set of classes and relations centred around the notions of stimuli, sensor, and observations. Based on the work of Quine, the skeleton defines stimuli as the (only) link to the physical environment. Empirical science observes these stimuli using sensors to infer information about environmental properties and construct features of interest.

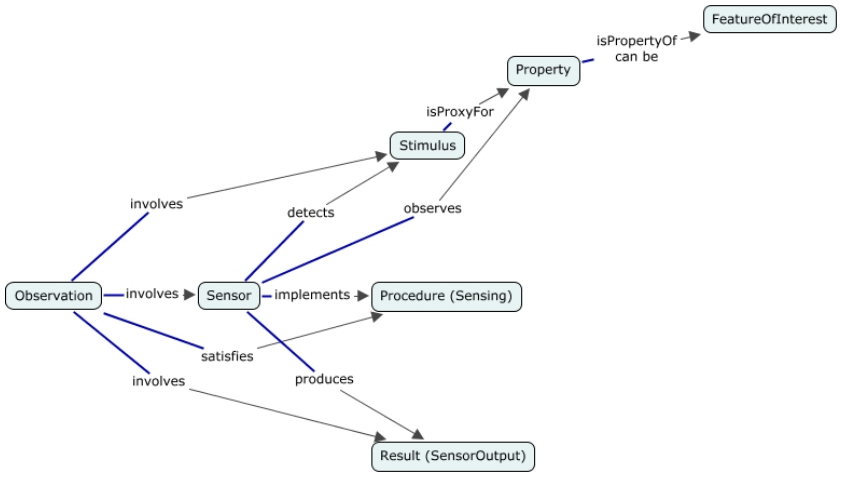


Figure D.3.: The Stimulus-Sensor-Observation Ontology Design Pattern [Compton et al., 2012b].

Definition for Stimuli: Stimuli are detectable changes in the environment, i.e., in the physical world. They are the starting point of each measurement as they act as triggers for sensors. Stimuli can either be directly or indirectly related to observable properties and, therefore, to features of interest. They can also be actively produced by a sensor to perform observations. The same types of stimulus can trigger different kinds of sensors and be used to reason about different properties. Nevertheless, a stimulus may only be usable as proxy for a specific region of an observed property.

The System Module

This section describes how to create a System object and uses a simple example to show how to model a system composed of sensors in the SSN ontology.

Definition for System: System is a unit of abstraction for pieces of infrastructure (and we largely care that they are) for sensing. A system has components, its subsystems, which are other systems.

In the SSN ontology a System is modelled as an instance of the class *ssn:System*. The relationship between an *ssn:System* instance and other sub-systems it eventually contains, is established by the predicate *ssn:has-Sub-System*.

Listing D.1 shows an example that declares *:SN-Node-TSB-ABC01* as a sensor network node i.e., a "mote", which includes both a Temperature sensor and a Humidity sensor (this is allowed because *ssn:Sensing-Device* is a sub-class of *ssn:System*).

```
:SN-Node-TSB-ABC01 a ssn:System ;
  ssn:hasSubSystem :TemperatureSensor-TSB-ABC01 ;
  ssn:hasSubSystem :HumiditySensor-TSB-ABC01 .

:TemperatureSensor-TSB-ABC01 a ssn:SensingDevice .

:HumiditySensor-TSB-ABC01 a ssn:SensingDevice .
```

Listing D.1: Example of using the SSN vocabulary to describe a System and two of its sub-systems (sensor nodes).

The resource *:SN-Node-TSB-ABC01* is intended as a proxy for any of the Systems deployed. The best practise would be to describe the Deployment itself as an instance of the class *ssn:Deployment* and establish a relationship between it and *:SN-Node-TSB-ABC01*, by using the predicate *ssn:has-Deployment*. Also the mote, should be related with the sensor Platform on which it has been deployed, which is an instance of the class *ssn:Platform*. This kind of relationship can be established by the predicate *ssn:on-Platform*.

The Measuring Module

Definition for Sensors: Sensors are physical objects that perform observations, i.e., they transform an incoming stimulus into another, often digital, representation. Sensors are not restricted to technical devices but also include humans as observers. A clear distinction needs to be drawn between sensors as objects and the process of sensing. We assume that objects are sensors while they perform sensing, i.e., while they are deployed. Furthermore, we also distinguish between the sensor and a procedure, i.e., a description, which defines how a sensor should be realised and deployed to measure a certain observable property. Similarly, to the capabilities of particular stimuli, sensors can only operate in certain conditions. These characteristics are modelled as observable properties of the sensors and includes their survival range or accuracy of measurement under defined external conditions. Finally, sensors can be combined to sensor systems and

networks. Many sensors need to keep track of time and location to produce meaningful results and, hence, are combined with further sensors to sensor systems such as weather stations.

The ontology defines several properties for instances of the class *ssn: Sensor*:

- *ssn: observes*: points to a property observed by a sensor (e.g., temperature, acceleration, wind speed). An object of this property must be an instance of the class ssn:Property.
- *ssn: has-Measurement-Capability*: Points to the description of the sensor's measurement capability expressed as an instance of the class
- *ssn: Measurement-Capability*. The description of a measurement capability includes such parameters as frequency, accuracy, measurement range, etc.

The class *ssn: Sensor* can represent any object with the sensing capability (e.g., in some cases a human observer can be a sensor). In most scenarios the sensors are implemented as devices. The *ssn: Device* is described in Section D.1.

A description of a Sensor is created by defining an instance of the class *ssn: Sensing-Device*. For example, in Listing D.2 a semantic description represents a concrete Sensor (accelerometer) attached to a kitchen tool (knife).

```
:ExTiltAccelerometer a ssn:SensingDevice ;
  ssn:observes ucumphysic:acceleration ;
  ssn:hasMeasurementCapability :ExTiltAccelCapab ;
  ssn:onPlatform :Knife_123 .
```

Listing D.2: Example of using the SSN vocabulary to describe a Sensor as a Sensing Device.

Note that the SSN ontology does not contain a vocabulary of possible properties which can be measured by sensors. Specific instances of the class *ssn: Property* have to be created by the user or (preferably) imported from an existing ontology, as shown in Listing D.3.

```
:smart-knife a owl:Ontology ;
...
owl:imports <http://purl.oclc.org/NET/muo/ucum/> ;
owl:imports <http://purl.oclc.org/NET/ssnx/ssn> .
:MyQuality rdfs:subClassOf muo:PhysicalQuality;
   rdfs:subClassOf ssn:Property .
```

Listing D.3: Example of incorporating the physical properties from the MyMobileWeb ontology, represented by the class *muo:PhysicalQuality*, in the SSN vocabulary, to describe the observed Property.

An instance of the class *ssn: Sensing-Device* represents one concrete physical object. It is possible that a use case deals with many sensors sharing common attributes, e.g., sensors measuring a specific property or sensing devices of the same model, which have the same measurement capabilities. In order to describe such groups of sensors with common properties, it is possible to define subclasses of the class *ssn: Sensor* with restricted property values.

Definition for Observed Properties: Properties are qualities that can be observed via stimuli by a certain type of sensors. They inhere in features of interest and do not exist independently. This does not imply that they do not exist without observations, our domain is restricted to those observations for which sensors can be implemented based on certain procedures and stimuli. To minimise the amount of ontological commitments related to the existence of entities in the physical world, observed properties are the only connection between stimuli, sensors, and observations on the one hand, and features of interests on the other hand.

Definition for Feature of Interest: Features of Interest are entities in the real world that are the target of sensing. As entities are reifications, the decision of how to carve out fields of sensory input to form such features is arbitrary to a certain degree and, therefore, has to be fixed by the observation (procedure).

The Measuring Capability Module

The measurement capabilities of a sensor are described as a set of measurement properties of a sensor, instances of the class *ssn: Measurement-Capability*. Possible measurement properties of a sensor are represented as subclasses of the class *ssn: Measurement-Property*. Currently, the ontology defines the following types of measurement properties:

- *ssn: Drift*: A, continuous or incremental, change in the reported values of observations over time for an unchanging quality.

- *ssn: Sensitivity*: Sensitivity is the quotient of the change in a result of sensor and the corresponding change in a value of a quality being observed.
- *ssn: Selectivity*: Selectivity is a property of a sensor whereby it provides observed values for one or more qualities such that the values of each quality are independent of other qualities in the phenomenon, body, or substance being investigated.
- *ssn: Accuracy*: The closeness of agreement between the value of an observation and the true value of the observed quality.
- *ssn: Measurement-Range*: The set of values that the sensor can return as the result of an observation under the defined conditions with the defined measurement properties. If no conditions are specified or the conditions do not specify a range for the observed qualities, the measurement range is to be taken as the condition for the observed qualities.
- *ssn: Detection-Limit*: An observed value for which the probability of falsely claiming the absence of a component in a material is α^2, given a probability α of falsely claiming its presence.
- *ssn: Precision*: The closeness of agreement between replicate observations on an unchanged or similar quality value, i.e., a measure of a sensor's ability to consistently reproduce an observation.
- *ssn: Response-Time*: The time between a change in the value of an observed quality and a sensor (possibly with specified error) settling on an observed value.
- *ssn: Frequency*: The smallest possible time between one observation and the next.
- *ssn: Latency*: The time between a request for an observation and the sensor providing a result.
- *ssn: Resolution*: The smallest difference in the value of a quality being observed that would result in perceptibly different values of observation results.

One instance of *ssn: Measurement-Capability* can describe a set of measurement properties linked by the property *ssn: has-Measurement-Property* and connected to a property using *ssn: for-Property* (a sensor can observe a number of properties and this allows measurement capabilities to be defined for each property). The conditions, in which these measurement properties are valid, are specified using the property *ssn: in-Condition* and expressed using an instance of the class *ssn: Condition*. The sensor ontology defines

conditions as *ssn: Property* (i.e. observable conditions that affect the operation of the sensor) but as with all properties doesn't define any further structure: an imported domain vocabulary must be used for this purpose. An instance of *ssn: Sensor* with multiple values for the property *ssn: has-Measurement-Capability*, represents different measurement capabilities depending on conditions.

In order to describe measurement properties of one specific sensor, it is necessary to define one or several instances of the class *ssn: Measurement-Capability* and use the property *ssn: has-Measurement-Capability* to link the sensor with its measurement capabilities. For example, in case of an accelerometer sensor attached to a knife, this can be described as illustrated in Listing D.4.

```
:ExTiltAccelerometer
  a :WiTilt30Accelerometer ;
  ssn:hasMeasurementCapability :ExTiltAccelCapab .
```

Listing D.4: Example of incorporating the physical properties from the MyMobileWeb ontology, represented by the class *muo:PhysicalQuality*, in the SSN vocabulary, to describe the observed Property [Compton et al., 2012b].

Please note that an instance of the class *ssn: Measurement-Capability* describes measurement properties of a specific physical sensor object. If it is necessary to describe measurement capabilities of a class of sensors, then it is necessary to define a restriction on the property *ssn: has-Measurement-Capability* for a particular subclass of ssn:Sensor which describes sensors of a specific type.

If all sensors of the same class have exactly the same measurement capabilities, then it is sufficient to define one instance of the class *ssn: Measurement-Capability*. Sometimes it is necessary to describe a range of possible measurement capabilities. In this case, one needs to define a subclass of the class *ssn: Measurement-Capability* where certain properties are restricted. For example, in Listing D.5, a superclass for all measurement capabilities of accelerometer sensors is described.

```
:AccelerationMeasurementCapability
  rdfs:subClassOf ssn:MeasurementCapability ;
  rdfs:subClassOf :bnode1 .
:bnode1 a owl:Restriction ;
 owl:onProperty ssn:forProperty ;
 owl:hasValue ucumphysic:acceleration .
```

Listing D.5: Example of declaring measurement capabilities that are shared between a specific kind of sensors (accelerometer).

The Observation Module

Definition for Observation: Observations act as the nexus between incoming stimuli, the sensor, and the output of the sensor, i.e., a symbol representing a region in a dimensional space. Therefore, we regard observations as social, not physical, objects. Observations can also fix other parameters such as time and location. These can be specified as parts of observation procedure. The same sensor can be positioned in different ways and, hence, collect data about different properties. In many cases, sensors perform additional processing steps or produce single results based on a series of incoming stimuli. Therefore, observations are rather contexts for the interpretation of the incoming stimuli than physical events.

The class *ssn: Observation* in the ontology provides the structure to represent a single observation. An observation is a situation that describes an observed feature, an observed property, a sensor and method of sensing used and a value observed for the property: that is, an observation describes a single value attributed to a single property by a particular sensor. Observations of multiple features or multiple properties of the one feature should be represented as either compound properties, features and values or as multiple observations, grouped in some appropriate structure.

The SSN ontology defines several properties for instances of the class *ssn: Observation*:

- *ssn: feature-Of-Interest*: points to the observed feature of interest. A feature of interest can be any observed real-world phenomenon (e.g., geographic entity, entity, etc.).
- *ssn: observed-Property*: points to the specific quality (properties in the ontology are qualities that can be observed by a sensor; qualities, on the other hand, can also abstract, qualities of abstract things, or in some other way not able to be sensed) of the feature of interest which was observed (e.g., temperature, acceleration, or speed).
- *ssn: observed-By*: points to a sensor which produced the observation (an instance of the class *ssn: Sensor*).
- *ssn: sensing-Method-Used*: points to a method used to produce the observation (an instance of the class *ssn: Sensing*). This could describe, for example, a particular way in which the sensor is used to make the observation.

- *ssn: observation-Result*: points to a result of the observation expressed as an instance of the class *ssn: Sensor-Output*.
- *ssn: quality-Of-Observation*: points to the adjudged quality of the result. This is complementary to the measurement capability information expressed for the sensor itself (Section D.1).
- *ssn: observation-Result-Time*: points to the time when the observation result became available.
- *ssn: observation-Sampling-Time*: points to the time when the observation result applies to the feature of interest.

The last two properties are defined as object properties, as the SSN ontology does not prescribe a specific format for the representation of time instants.

The result of an observation is expressed by an instance of the class *ssn: Sensor-Output*. The ontology defines the following properties applicable to the class:

- *ssn: is-Produced-By*: points to a sensor which produced the output (an instance of the class *ssn: Sensor*).
- *ssn: has-Value*: points to the actual value of the observation, e.g., "30C", "60 mph", etc. This is expressed as an instance of the class *ssn: Observation-Value*. The ontology does not restrict the format of an observation value: the actual properties can be defined by the user or imported from a third-party ontology.

Information about the time at which the observation has been made, known has the Sampling Time and the time at which the result is available, can be attached to the *ssn: Observation* class. This can be done with the help of the *ssn: observation-Sampling-Time* and *ssn: observation-Result-Time* properties.

In order to describe an observation made by a sensor, an instance of the class *ssn: Observation* should be used. For example, in Listing D.6 we have a sensor which is attached to a knife and measures its acceleration to capture the time when the user is cutting. To represent its observations, we define a class *:AccelerationObservation* as a subclass of the class *ssn: Observation*.

```
1 :AccelerationObservation
2   rdfs:subClassOf ssn:Observation ;
3   rdfs:subClassOf :bnode1 ;
4   rdfs:subClassOf :bnode2 ;
```

```
5    rdfs:subClassOf :bnode3 .
6  :bnode1 a owl:Restriction ;
7    owl:onProperty ssn:observationResult ;
8    owl:allValuesFrom :AccelerationSensorOutput .
9  :bnode2 a owl:Restriction ;
10   owl:onProperty ssn:observedBy ;
11   owl:allValuesFrom :Accelerometer .
12 :bnode3 a owl:Restriction ;
13   owl:onProperty ssn:observedProperty ;
14   owl:allValuesFrom ucumphysic:acceleration .
```

Listing D.6: Example of using the SSN vocabulary to describe an Observation made by a Sensor [Compton et al., 2012b].

Definition for Result or Sensor Output: The result is a symbol representing a value as outcome of the observation. Results can act as stimuli for other sensors and can range from counts and Booleans, to images, or binary data in general.

The Deploy Module

The ssn:System class is an abstraction for parts of a sensing infrastructure. The ssn:Sensor class in the ontology provides the structure to represent a concrete sensing object. Sensor can represent any object with the sensing capability (e.g., in some cases a human observer can be also a sensor). However, in many scenarios the sensors are devices. The ssn:Device class describes a device and inherits all the properties of the ssn:System class. The following provides an overview of the main classes and properties related to deployment of a network of sensors in the ontology:

- *ssn: has-Deployment*: Points to deployment description of sensor expressed as an instance of the ssn:Deployment class. The description of a Deployment refers to ssn:System and it is also a subclass of ssn:DeploymentRelatedProcess and inherits all the properties from this class.
- The *ssn: Deployment-Related-Process* class groups various Processes related to Deployment. For example, it includes installation, maintenance and deployment features.
- The *ssn: System* class for parts of a sensing infrastructure. A system has components, its subsystems, which are other systems. A system is deployed on a Platform.

- *ssn: deployed-On-Platform* points to platform on which the system is deployed.
- *ssn: Platform* includes different components that can be attached to Sensor and also different features such as measurement properties, operating properties and system settings.
- *ssn: deployed-System* provides relation between a deployment and a deployed system.

A deployment of a system is created by defining an instance of the *ssn: Deployment* class, as shown in Listing D.7.

```
:ABC01Deployment
  a ssn:Deployment ;
  ssn:deployedOnPlatform :UNiS-TSBPlatform ;
  ssn:deployedSystem :SN-Node-TSB-ABC01 .
:UNiS-TSBPlatform a ssn:Platform .
:SN-Node-TSB-ABC01 a ssn:System .
```

Listing D.7: Example of using the SSN vocabulary to describe a Deployment of a System on a Platform.

The Platform Site Module

The SSN ontology defines relationships between classes such as *ssn: Deployment*, *ssn: System* and *ssn: Platform*. However it does not provide some aspects such as spatial attributes for the Platform class. There are three options to represent locations.

When the Dolce Ultralite alignment is enforced, an *ssn: Platform* is a *dul:PhysicalObject* and as such may have a *dul:hasLocation* relation to a *dul:PhysicalPlace* location. A PhysicalPlace is an abstraction of a real-world place.

As well as the relative locations above, Dolce Ultralite also allows absolute locations. The location of an entity is an observable aspect of the entity and is thus an *ssn: Property*, properties have values thanks to the predicate *ssn: has-Value*, represented as regions, in this case a *dul:SpaceRegion*. Hence, a sensor or platform can be given, for example, an absolute latitude and longitude, a location relative to another co-ordinate, or any other sort of value for location. Of course, if this method is to be used often, sub properties of *ssn: has-Value* could be defined, e.g., *:hasLatLong*, *:hasAbsoluteLocation*, *:hasCoordinates*, etc., to provide descriptive names for locations, depending on the method used.

The third option is to define or import a further method for representing locations.

The example in Listing D.8shows how location attributes are defined for a platform by using concepts from the Dolce Ultralite ontology.

```
:UNiS-TSBPlatform
  a ssn:Platform ;
  dul:hasLocation :PhysicalPlace_UNiSTestBED-BA03A .
:PhysicalPlace_UNiSTestBED-BA03A a dul:PhysicalPlace ;
 dul:isLocationOf :UNiS-TSBPlatform .
```

Listing D.8: Example of using the SSN vocabulary to describe a Platform and its location.

The Operating Restriction Module

The operational and survival restrictions can be described for an instance of the class *ssn: System*. The operational properties are referred to by using *ssn: has-Operating-Range*, which in turn *ssn: hasOperatingProperty* such as *ssn: Maintenance-Schedule* and *ssn: Operating-Power-Range*. The survival properties are referred to by using *ssn: hasSurvivalRange* and include properties (*ssn: has-Survival-Property*) such as *ssn: Battery-Life-Time* and *ssn: System-Life-Time*.

The main classes and properties to describe the Operating Restrictions for a system are shown in Figure D.4.

An Operational Restriction can be defined in a Condition. The predicate *ssn: in-Condition* relates an instance of the *ssn: Measurement-Capability* class to on of the *ssn: Condition* class. *ssn: Condition* represents ranges for qualities that act as conditions on a system / sensor's operation. For example, the accuracy features of a sensor represented by *ssn: Survival-Range* and *ssn: Operating-Range* are defined in a certain temperature, e.g., 25 degree Celsius.

The Device Module

In most scenarios the sensors are implemented as devices. The class *ssn: Device* 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). 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*.

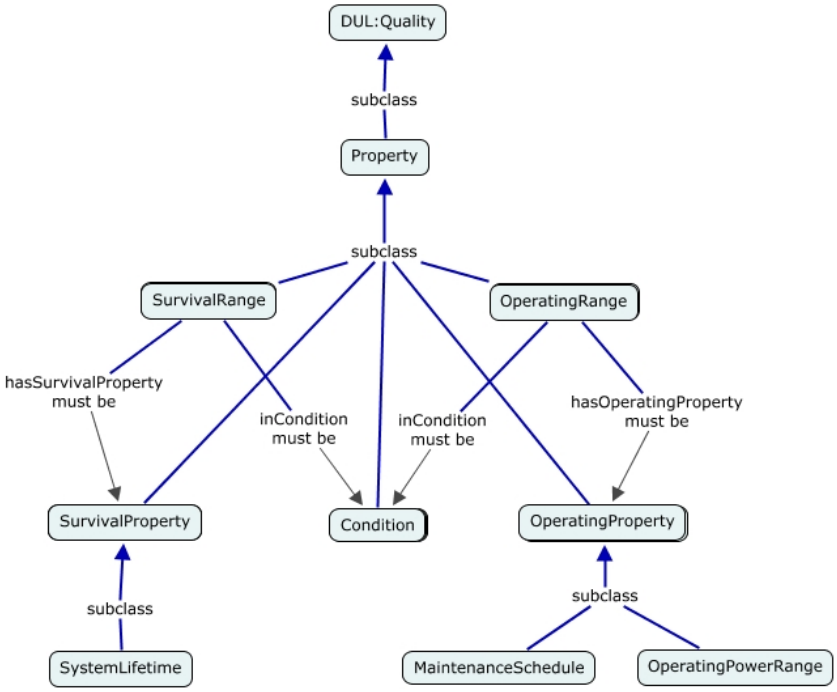


Figure D.4.: Main classes and properties of the Operating Restriction Module from the SSN vocabulary [Compton et al., 2012b].

```
:SN-Node-TSB-ABC01
  a ssn:System ;
  ssn:hasOperatingProperty :TSBOperatingPowerRange .
:TSBOperatingPowerRange a ssn:OperatingPowerRange ;
 ssn:hasValue :Current_Draw_Idle .
:Current_Draw_Idle a dul:Amount ;
 dul:hasDataValue "'21"' ;
 muo:measuredIn ucumunit:microAmpere .
:PhysicalPlace_UNiSTestBED-BA03A a dul:PhysicalPlace ;
 dul:isLocationOf :UNiS-TSBPlatform .
```

Listing D.9: Example of using the SSN vocabulary to describe a Restriction on an Operating Property of a System.

An example of device is included in The Measuring Module Section D.1 and is used in The MeasuringCapability Module Section D.1, to illustrate how a *ssn: Measurement-Capability* can be specified.

The Energy Module

This module is a placeholder for possible extensions for SSN ontology users wishing to model the energy management aspects of a sensor network. It contains two classes *ssn: Battery-Life-Time* and *ssn: Operating-Power-Range*.

The example in Figure D.5 illustrates how to define the lifespan of a battery in function of the current drawn from it. Generally, more than one ssn:BatteryLifetime instance will be used to model how the battery performs in various conditions. Here in particular, it is shown how to define a lifetime of 20 hours for a sensor node battery when it is used to deliver an electric current of 65 milliamperes.

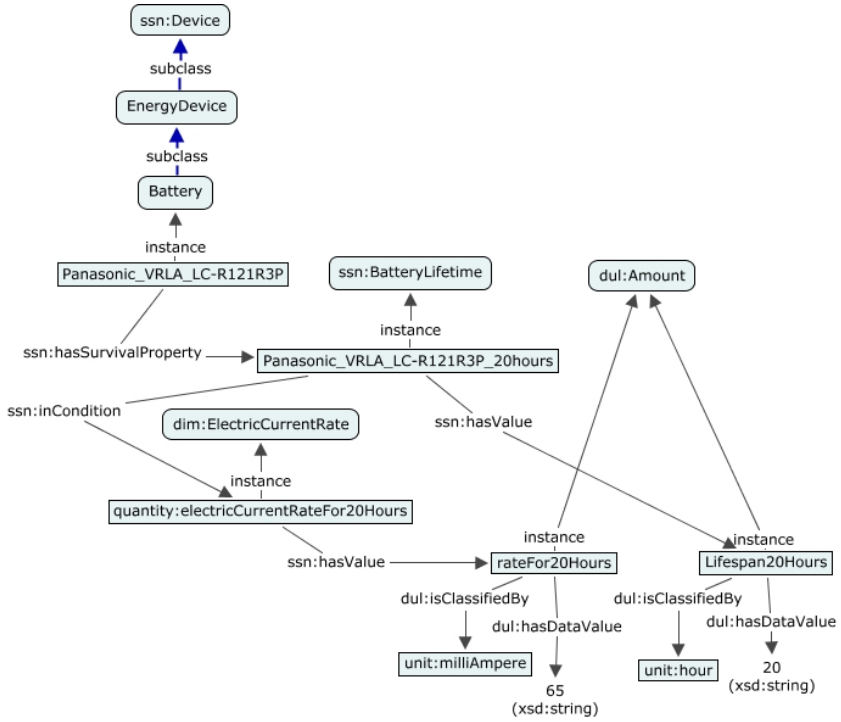


Figure D.5.: Example showing how to use the SSN vocabulary to define a lifetime of 20 hours for a sensor node battery when it is used to deliver an electric current of 65 milliamperes [Compton et al., 2012b].

Other Modules

Additional modules like the Process and the Constraint Block one, are just acknowledged here but not described in detail for reason of brevity.

Definition for Procedure: Procedure is a description of how a sensor works, i.e., how a certain type of stimuli is transformed to a digital representation, perhaps a description of the scientific method behind the sensor. Consequently, sensors can be thought of as implementations of sensing methods where different methods can be used to derive information about the same type of observed property. Sensing methods can also be

used to describe how observations where made: e.g., how a sensor was positioned and used. Simplifying, one can think of sensing as recipes for observing.

Implementation of the SSO patter:

Figure D.6 illustrates the changes applied to the Stimulus-Sensor-Observation (SSO) Ontology Design Pattern to include the classes and relations already present in the SSN ontology. In particular, several shortcut properties have been added to provide users with more options to create links between the main classes i.e., *ssn: Observation*, *ssn: Sensor*, *ssn: Stimulus*, *ssn: Property* and *ssn: Feature-Of-Interest*.

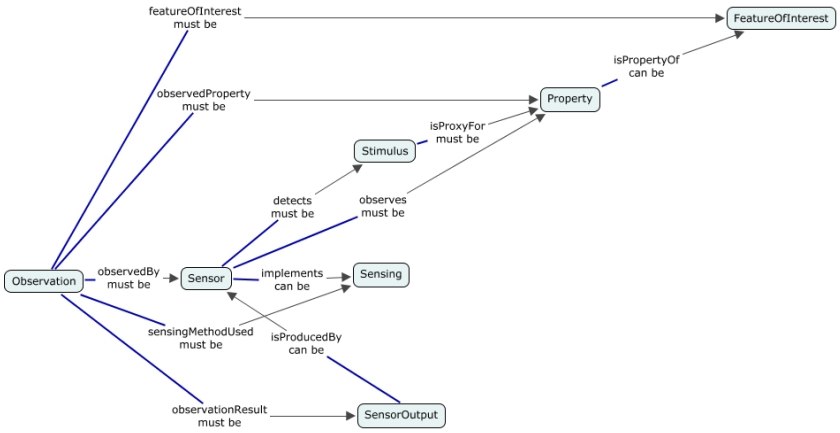


Figure D.6.: Overview of the Semantic Sensor Network ontology modules [Compton et al., 2012b].

Also, a few class names have been changed to match the choices previously made for the SSN ontology

- Result has been replaced by *ssn: Sensor-Output*,
- Procedure has been replaced by *ssn: Sensing*,
- SensorInput has been kept as a class equivalent to *ssn: Stimuli*.

Alignment with Dolce Ultralite

To ease the interpretation of the used primitives as well as to boost ontology alignment and matching, the SSO pattern has been aligned to the Ultralite version of the Dolce foundational ontology and refined to match the content of the SSN ontology.

Note that for this reason, new classes and relations are introduced based on subsumption and equivalence. For instance, the first pattern uses the generic involves relation, while the Dolce-aligned version distinguishes between events and objects and, hence, uses *dul:includesEvent* and *dul:includesObject*, respectively.

Each class of the SSN ontology is then defined as a subclass of an existing dul class and related to other SSN and dul classes. New types of relations are only introduced when the domain or range have to be changed, in all other cases the relations from dul are reused. The aim of the resulting extension to dul is to preserve all ontological commitments defined before. Figure D.7 depicts an overview of the alignment of all the SSN concepts with Dolce Ultralite.

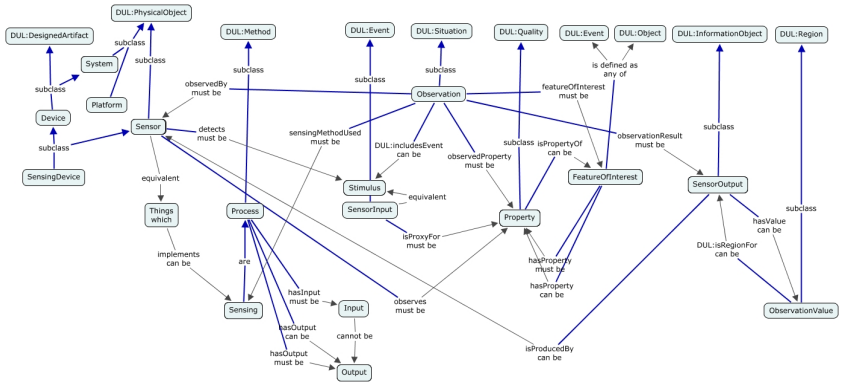


Figure D.7.: Alignment of the Semantic Sensor Network ontology to Dolce Ultralite [Compton et al., 2012b].

In particular Sensors are defined as subclasses of physical objects *dul:PhysicalObject*, as shown in Listing D.10. Therefore, they have to participate in at least one *dul:Event* such as their deployment. This is comparable to the ontological distinction between a human and a human's life. Sensors are related to their sensing method and observations using the *dul:implements* and *dul:isObjectIncludedIn* relations, respectively.

```
1 ssn:Sensors rdf:type owl:Class ;
2  rdfs:subClassOf <http://www.loa-cnr.it/ontologies/DUL.owl#
      PhysicalObject> ;
3  rdfs:isDefinedBy <http://purl.oclc.org/NET/ssnx/ssn#> .
```

Listing D.10: Example of using the OWL DL 2 language to define a subclass of the concept *dul:PhysicalObject*.

The class *ssn: Observation* is specified as a subclass of *dul:Situation*, which in turn is a subclass of *dul:Social-Object*. The required relation to stimuli, sensors, and results can be modelled using the *dul:includes-Event*, *ssn: observed-By* and *ssn: observation-Result*

relationships, respectively. Observation procedures can be integrated by *dul:sensing-Method*.

ssn: Observed-Property is defined as a subclass of *dul:Quality*. Types of Properties, such as temperature or pressure should be added as subclasses of *ssn: Property* instead of individuals. A new relation called *ssn: is-Property-Of* is defined as a subrelation of *dul:is-Quality-Of* to relate a property to a feature of interest.

Features of interest can be events or objects but not qualities and abstracts to avoid complex questions such as whether there are qualities of qualities. The need to introduce properties for qualities is an artefact of reification and confuses qualities with features or observations. For instance, accuracy is not a property of a temperature but the property of a sensor or an observation procedure.

The *ssn: Sensor-Output* class (Result in the SSO pattern) is modelled as a subclass of *dul:Information-Object*. The management of the concrete data value is introduced through a *ssn: has-Value* relationships to a *dul:Region* and then through the data property *dul:has-Region-Data-Value* in conjunction with some xsd data type.

Achievement by Examples

A sample use case in which the SSN ontology can show its potential, is the Data Discovery and Linking one. We can imagine one of the SPITFIRE partners, e.g., Henning, in charge of the Sensor Network deployed in the TUBS building, which includes 30 sensor nodes. He wants to find all the observations that meet certain criteria, and possibly link them to other external data sources.

For instance, a flood has occurred in the first floor male toilet; so he wants to

- find all the observations related to water consumption and water pressure,
- tide the available information, in a specific bounding box (or in a specific region) and obtained in the last 24 hours,
- link them to the economic assets that could be affected by a potential flood event.

Also, Henning works under the assumption that although the local Sensor Dataset is the primary data source used, there could be other data sources (sensor-based or not) that could dynamically appear in the regions of interest, which he might not control, but which can provide useful information.

Hence, the primary actor in this use case, is the one with operational functions, e.g., emergency response, who will benefit from access to information that is embellished with real-time representation of water pressure and water leakage; and who may benefit from integrating this with other existing datasets in order to support multi-criteria decisions and operations.

Henning will need to simply run a SPARQL query like the one shown in Listing D.11, taking advantage of several SSN concepts and predicates. Also external datasets can be queried by using different techniques, satisfying Henning who wanted to collect information even from external eventual data sources he is not in control, e.g., individual private mobile sensor platforms, owned by other researchers on the same floor. In this example the external datasets are listed as named graphs. Alternatives are graphs and federated queries, in which the query is forwarded directly to external SPARQL endpoints and only the results are imported locally, while in case of either graphs or named graphs, the whole graphs are imported locally and the query is run locally on their triples.

```
SELECT ?reading
  FROM NAMED <http://example.org/localDataset1>
 FROM NAMED <http://example.org/externalDataset1>
 WHERE{
  ?reading a ssn:SensorOutput .
  ?o servation a ssn:Observation ;
   ssn:observationResult ?reading ;
   ssn:observedBy ?node ;
   ssn:observationResultTime ?date ;
  ssn:observedProperty ?prop .
  {?prop a :WaterConsumption .}
  UNION
  {?prop a :WaterLeakage .}
   ?node a ssn:Sensor ;
  ssn:onPlatform ?platform .
  ?platform a ssn:Platform ;
  dul:hasLocation :TUBSMaleToilet1 .
  FILTER (
      ?date > "2012-03-30"^xsd:date &&
     ?date < "2012-03-31"^xsd:date
   )
  }ORDER BY ?observation DESC(?date)
```

Listing D.11: SPARQL query that selects all the observations produced by any of the sensor nodes located in the first floor male toilet *:TUBSMaleToilet1*, as long as both collected in the time range between the 30th and the 31st of March 2012, and observing either Water Consumption or Water Leakage.

This SPARQL query will return a list of the searched sensor observation URIs, in either JSON, XML, RDF or HTML formats. An excerpt of the possible results in JSON format is shown in Listing D.12.

```
{"head":{"vars":["reading"]
  },"results":{
     "bindings": [
     {"reading":{
       "type":"uri",
       "value":"http://example.org/node6/observation/obs1"}
     },{"reading":{
       "type":"uri",
       "value":"http://example.org/node7/observation/obs2"}
      },{"reading":{
       "type":"uri",
       "value":"http://example.org/node5/observation/obs1"}
      },{"reading":{
       "type":"uri",
       "value":"http://example.org/node4/observation/obs11"}
      },{"reading":{
       "type":"uri",
       "value":"http://example.org/node2/observation/obs40"}
      },{"reading":{
       "type":"uri",
       "value":"http://example.org/node3observation/obs32"}
      },{"reading":{
       "type":"uri",
       "value":"http://example.org/node1/observation/obs45"}
      }]}}
```

Listing D.12: JSON results to the SPARQL query in Listing D.11. It consists in a list of the URIs of those sensor observations that have been collected in a specific place and at a certain time range, while observing either Water Consumption or Water Leakage.

Conclusions

The SSN ontology is a very comprehensive and robust base to describe the majority of sensor-related information. However, since it has been defined to be cross-domain, it lacks specific information that we need for our own purposes, to realize our scenarios that involve low-level Sensor Network management and Energy Consumption monitoring. Consequently we extended the SSN ontology with our application-specific concepts, which we aligned to Dolce Ultralite as well. The SPITFIRE vocabulary is described in Section 4.2.2.

Appendix E.

Dolce+DnS Ultralite Ontology

An upper-level ontology (also known as either top-level or foundation ontology) is an ontology which describes very general concepts that are the same across all knowledge domains (Section 3.2.2). An important function of an upper ontology is to support very broad semantic interoperability between a large number of ontologies which are accessible ranking under this upper ontology. As the rank metaphor suggests, it is usually a hierarchy of entities and associated rules (both theorems and regulations) that attempts to describe those general entities that do not belong to a specific problem domain.

Dolce: The Descriptive Ontology for Linguistic and Cognitive Engineering (Dolce) is the first module of the WonderWeb foundational ontologies library. As implied by its acronym, Dolce has a clear cognitive bias, in that it aims at capturing the ontological categories underlying natural language and human commonsense. The categories it introduces are thought of as cognitive artefacts, which are ultimately depending on human perception, cultural imprints and social conventions.

DnS: The Descriptions and Situations (DnS) ontology, is a constructivist ontology that pushes Dolce's descriptive stance even further. It assumes Dolce as a ground top-level vocabulary, and exploits Content ontology Design Patterns (CPs), which provide a framework to annotate focused fragments of a reference ontology, i.e., the parts of an ontology containing the types and relations that underlie expert reasoning in given fields or communities.

Dolce+DnS Ultralite: Both Dolce and DnS are particularly devoted to the treatment of social entities [Gangemi, 2010], e.g., organisations, collectives, plans, norms, and

information objects. Dolce+DnS Ultralite (dul) is a lighter OWL axiomatisation of Dolce and DnS combined together, which also

- simplifies the names of many classes and properties,
- adds extensive inline comments,
- thoroughly aligns to the repository of Content patterns [1].

The final result is a lightweight, easy-to-apply foundational ontology for modelling either physical or social contexts. Moreover dul's simplification, greatly speeds up consistency checking and classification of OWL domain ontologies that are plugged to it, without significant loss in expressivity. It is also available in modules, called *content ontology design patterns*, which can be applied independently in the design of domain ontologies. Specifically we chose Dolce Ultralite because of both its large uptake and its support to describe situations.

Alignment with our Ontology

To ease the interpretation of the used primitives as well as to boost ontology alignment and matching, the SPITFIRE vocabulary has been aligned to Dolce Ultralite. Each class of the SPITFIRE ontology is then defined as a subclass of an existing dul class and related to other SSN and dul classes. New types of relations are only introduced when the domain or range have to be changed, in all other cases the relations from dul are reused. The aim of the resulting extension to dul is to preserve all ontological commitments defined before.

Figure E.1 depicts an overview of the alignment of all our concepts with Dolce Ultralite, while predicates are omitted.

Achievement by Examples

One of the main advantages achieved by aligning the SPITFIRE vocabulary with Dul is the interoperability. This brings several advantages. For instance, our IBBT partners in SPITFIRE, who need to semantically describe the robots that they have, on which sensors have been attached, rather than keeping a separated ontology for robots, they would be able to easily plug their robot ontology to the dul hierarchy.

[1] http://ontologydesignpatterns.org/wiki/Ontology_Design_Patterns_._org_%28ODP%29

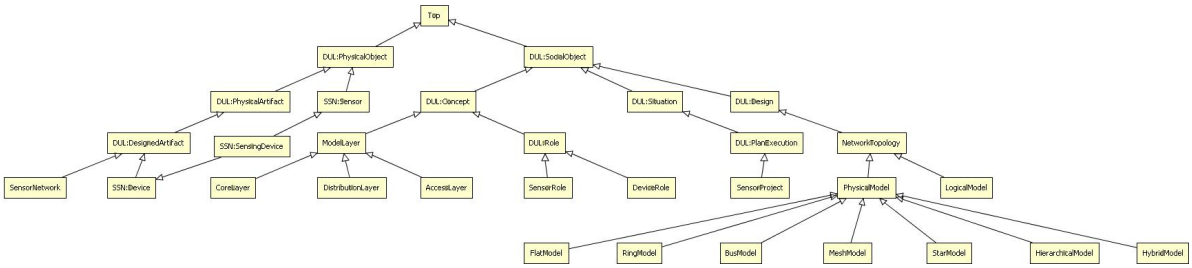


Figure E.1.: Alignment of the SPITFIRE ontology to Dolce Ultralite. Those terms not preceded by any namespace are taken from the SPITFIRE ontology.

By aligning with dul they will be automatically aligned with our own vocabulary, as well. In particular a *:Robot* will be a subclass of *dul:Designedartefact*, like *spitfire:SensorNetwork* and *ssn:Device* are. Consequently, whenever a user would like to retrieve the amount of physical designed artefacts, will automatically and seamlessly get both Devices, SensorNetworks and Robots, included in the search results, by running a SPARQL query like the one shown in Listing E.1.

```
1 SELECT ?objects
2 WHERE{
3 {
4   ?objects a dul:DesignedArtifact .}
5   UNION
6   {?objects a ?class .
7    ?class rdfs:subClassOf dul:DesignedArtifact}
8 }
```

Listing E.1: SPARQL query that selects all the available instance of physical designed artefacts, including instances of their subclasses.

Appendix F.

Contextualised Sensor Ontology – source code

Listing F.1 shows the SPITFIRE ontology source code. It includes a specification of both the classes, the creators and the legal copyrights. Details about the development of the SPITFIRE Ontology can be found in Section 4.2.2. It also includes definitions of the Object and Data Properties.

The RDF source code has been serialized using the Turtle syntax and, for clarity reasons, the initial declaration of namespace has been omitted: such namespaces are the same used throughout this document.

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix spt: <http://spitfire-project.eu/ontology/ns/> .
@prefix spt-c: <http://spitfire-project.eu/
  ontology/ns/context-types#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix event: <http://events.semantic-multimedia.org/
   ontology/2008/12/15/model.owl#> .
@prefix muo: <http://www.w3.org/2001/XMLSchema#> .
@prefix dul: <http://www.loa-cnr.it/ontologies/DUL.owl#> .
@prefix ao: <http://purl.org/ontology/ao/
   associationontology.html#> .
@base <http://spitfire-project.eu/ontology/ns/> .

```

```
<http://spitfire-project.eu/ontology/ns> a owl:Ontology ;
   <http://purl.org/dc/elements/1.1/creator>
      <http://myr.altervista.org/foaf.rdf#me> ;
   <http://purl.org/dc/elements/1.1/creator>
      "Alexandre Passant"^^xsd:string ;
  <http://purl.org/dc/elements/1.1/creator>
     "Michael Hausenblas"^^xsd:string ;
   <http://purl.org/dc/elements/1.1/rights>
      "Copyright 2010 - 2012 SPITFIRE." ;
   rdfs:seeAlso "http://spitfire-project.eu" ;
   <http://purl.org/dc/elements/1.1/title>
      "SPITFIRE Ontology"@en ;
   <http://purl.org/dc/elements/1.1/identifier>
      "http://spitfire-project.eu/ontology/ns" ;
   <http://www.w3.org/Consortium/Legal/2002/license>
   <http://www.w3.org/Consortium/Legal/2002
   /copyright-software-20021231.html> .


#    ////////////////////////////////
#    //
#    // Classes
#    //
#    ////////////////////////////////


:SensorNetwork rdf:type owl:Class ;
       rdfs:subClassOf dul:DesignedArtifact ;
   rdfs:isDefinedBy
   <http://spitfire-project.eu/ontology/ns> .


:NetworkTopology rdf:type owl:Class ;
       rdfs:subClassOf dul:Design ;
   rdfs:isDefinedBy
   <http://spitfire-project.eu/ontology/ns> .


:HierarchicalModel rdf:type owl:Class ;
       rdfs:subClassOf spt:NetworkTopology ;
   rdfs:isDefinedBy
   <http://spitfire-project.eu/ontology/ns> .

```

```
62
63 :MeshModel rdf:type owl:Class ;
64         rdfs:subClassOf spt:NetworkTopology ;
65    rdfs:isDefinedBy
66    <http://spitfire-project.eu/ontology/ns> .
67
68
69 :FlatModel rdf:type owl:Class ;
70         rdfs:subClassOf spt:NetworkTopology ;
71    rdfs:isDefinedBy
72    <http://spitfire-project.eu/ontology/ns> .
73
74
75 :ModelLayer rdf:type owl:Class ;
76         rdfs:subClassOf dul:Design ;
77    rdfs:isDefinedBy
78    <http://spitfire-project.eu/ontology/ns> .
79
80
81 :AccessLayer rdf:type owl:Class ;
82         rdfs:subClassOf spt:ModelLayer ;
83    rdfs:isDefinedBy
84    <http://spitfire-project.eu/ontology/ns> .
85
86
87 :DistributionLayer rdf:type owl:Class ;
88         rdfs:subClassOf spt:ModelLayer ;
89    rdfs:isDefinedBy
90    <http://spitfire-project.eu/ontology/ns> .
91
92
93 :CoreLayer rdf:type owl:Class ;
94         rdfs:subClassOf spt:ModelLayer ;
95    rdfs:isDefinedBy
96    <http://spitfire-project.eu/ontology/ns> .
97
98
99 :NetworkLink rdf:type owl:Class ;
100          rdfs:subClassOf dul:PhysicalObject ;
101    rdfs:isDefinedBy
102    <http://spitfire-project.eu/ontology/ns> .
103
104
```

```
105 :NetworkLink rdf:type owl:Class ;
106          rdfs:subClassOf dul:Quality ;
107    rdfs:isDefinedBy
108    <http://spitfire-project.eu/ontology/ns> .
109
110
111 :LinkActivity rdf:type owl:Class ;
112          rdfs:subClassOf dul:Quality ;
113    rdfs:isDefinedBy
114    <http://spitfire-project.eu/ontology/ns> .
115
116
117 :SensorRole rdf:type owl:Class ;
118   rdfs:subClassOf dul:Role ;
119    rdfs:isDefinedBy <http://spitfire-project.eu/ontology/ns> .
120
121
122 :DeviceRole rdf:type owl:Class ;
123    rdfs:subClassOf dul:Role ;
124    rdfs:isDefinedBy
125    <http://spitfire-project.eu/ontology/ns> .
126
127
128 :SensorProjectTopic rdf:type owl:Class ;
129   rdfs:isDefinedBy
130   <http://spitfire-project.eu/ontology/ns> .
131
132
133 :Activity rdfs:subClassOf dul:Event ;
134     rdfs:subClassOf ao:LikeableAssociation ;
135     rdfs:isDefinedBy
136     <http://spitfire-project.eu/ontology/ns> .
137
138
139 :Mood a owl:Class ;
140     rdfs:isDefinedBy
141     <http://spitfire-project.eu/ontology/ns> .
142
143 :PlatformTemporalProperty a owl:Class ;
144     rdfs:isDefinedBy
145     <http:/spitfire-project.eu/ontology/ns> ;
146     rdfs:subClassOf :TemporalProperty .
147
```

```
148 :SensorTemporalProperty a owl:Class ;
149     rdfs:isDefinedBy
150     <http:/spitfire-project.eu/ontology/ns> ;
151     rdfs:subClassOf :TemporalProperty .
152
153 :TemporalProperty a owl:Class ;
154     rdfs:isDefinedBy
155     <http:/spitfire-project.eu/ontology/ns> .
156
157
158 #     ////////////////////////////////
159 #     //
160 #     // Object properties
161 #     //
162 #     ////////////////////////////////
163
164
165 :belongsToNetwork rdf:type owl:ObjectProperty ;
166   rdfs:subPropertyOf dul:isPartOf ;
167    rdfs:domain dul:PhysicalObject ;
168    rdfs:range spt:SensorNetwork  ;
169    rdfs:isDefinedBy
170    <http://spitfire-project.eu/ontology/ns> .
171
172
173 :describesNetwork rdf:type owl:ObjectProperty ;
174   rdfs:subPropertyOf dul:describes ;
175    rdfs:domain spt:NetworkTopology  ;
176    rdfs:range spt:SensorNetwork  ;
177    rdfs:isDefinedBy
178    <http://spitfire-project.eu/ontology/ns> .
179
180
181 :describesModel rdf:type owl:ObjectProperty ;
182   rdfs:subPropertyOf dul:describes ;
183    rdfs:domain spt:NetworkTopology  ;
184    rdfs:range spt:ModelLayer  ;
185    rdfs:isDefinedBy
186    <http://spitfire-project.eu/ontology/ns> .
187
188
189 :isLayerOf rdf:type owl:ObjectProperty ;
190   rdfs:subPropertyOf dul:isPartOf ;
```

```
191    rdfs:domain spt:ModelLayer  ;
192    rdfs:range spt:NetworkTopology  ;
193    owl:inverseOf spt:layer ;
194    rdfs:isDefinedBy
195    <http://spitfire-project.eu/ontology/ns> .
196
197
198 :layer rdf:type owl:ObjectProperty ;
199   rdfs:subPropertyOf dul:hasPart ;
200    rdfs:domain spt:NetworkTopology  ;
201    rdfs:range spt:ModelLayer  ;
202    owl:inverseOf spt:isLayerOf ;
203    rdfs:isDefinedBy
204    <http://spitfire-project.eu/ontology/ns> .
205
206
207 :belongsToLayer rdf:type owl:ObjectProperty ;
208   rdfs:subPropertyOf dul:isPartOf ;
209    rdfs:domain dul:PhysicalObject ;
210    rdfs:range spt:ModelLayer  ;
211    rdfs:isDefinedBy
212    <http://spitfire-project.eu/ontology/ns> .
213
214
215 :networkRole rdf:type owl:ObjectProperty ;
216   rdfs:subPropertyOf dul:hasRole ;
217    rdfs:domain dul:PhysicalObject ;
218    rdfs:range dul:Role  ;
219    rdfs:isDefinedBy
220    <http://spitfire-project.eu/ontology/ns> .
221
222
223 :hasLink rdf:type owl:ObjectProperty ;
224   rdfs:subPropertyOf dul:hasComponent ;
225    rdfs:domain dul:PhysicalObject  ;
226    rdfs:range spt:NetworkLink  ;
227    owl:inverseOf spt:isLinkOf ;
228    rdfs:isDefinedBy
229    <http://spitfire-project.eu/ontology/ns> .
230
231
232 :isLinkOf rdf:type owl:ObjectProperty ;
233   rdfs:subPropertyOf dul:isComponentOf ;
```

```
234     rdfs:domain spt:NetworkLink  ;
235     rdfs:range dul:PhysicalObject  ;
236     owl:inverseOf spt:hasLink ;
237     rdfs:isDefinedBy
238     <http://spitfire-project.eu/ontology/ns> .
239
240
241 :linkQuality rdf:type owl:ObjectProperty ;
242   rdfs:subPropertyOf dul:hasQuality ;
243     rdfs:domain spt:NetworkLink ;
244     rdfs:range spt:LinkQuality  ;
245     owl:inverseOf spt:isQualityOf ;
246     rdfs:isDefinedBy
247     <http://spitfire-project.eu/ontology/ns> .
248
249
250 :isLinkQualityOf rdf:type owl:ObjectProperty ;
251   rdfs:subPropertyOf dul:isQualityOf ;
252     rdfs:domain spt:LinkQuality  ;
253     rdfs:range spt:NetworkLink ;
254     owl:inverseOf spt:linkQuality ;
255     rdfs:isDefinedBy
256     <http://spitfire-project.eu/ontology/ns> .
257
258
259 :linkQualityValue rdf:type owl:DatatypeProperty ;
260   rdfs:subPropertyOf dul:hasDataValue ;
261     rdfs:domain spt:LinkQuality  ;
262     rdfs:range xsd:double ;
263     rdfs:isDefinedBy
264     <http://spitfire-project.eu/ontology/ns> .
265
266
267 :priorityLevel rdf:type owl:DatatypeProperty ;
268     rdfs:domain dul:Role ;
269     rdfs:range xsd:integer ;
270     rdfs:isDefinedBy
271     <http://spitfire-project.eu/ontology/ns> .
272
273
274 :linkActivity rdf:type owl:ObjectProperty ;
275     rdfs:domain spt:NetworkLink ;
276     rdfs:range spt:LinkActivity ;
```

```
277    owl:inverseOf spt:isLinkActivityOf ;
278    rdfs:isDefinedBy
279    <http://spitfire-project.eu/ontology/ns> .
280
281
282 :isLinkActivityOf rdf:type owl:ObjectProperty ;
283    rdfs:domain spt:LinkActivity ;
284    rdfs:range spt:NetworkLink ;
285    owl:inverseOf spt:linkActivity ;
286    rdfs:isDefinedBy
287    <http://spitfire-project.eu/ontology/ns> .
288
289
290 :startActivity rdf:type owl:DatatypeProperty ;
291    rdfs:domain spt:LinkActivity  ;
292    rdfs:range xsd:dateTime ;
293    rdfs:isDefinedBy
294    <http://spitfire-project.eu/ontology/ns> .
295
296
297 :endActivity rdf:type owl:DatatypeProperty ;
298    rdfs:domain spt:LinkActivity>  ;
299    rdfs:range xsd:dateTime ;
300    rdfs:isDefinedBy
301    <http://spitfire-project.eu/ontology/ns> .
302
303
304 :linkActivityValue rdf:type owl:DatatypeProperty ;
305    rdfs:domain spt:LinkActivity  ;
306    rdfs:range xsd:integer ;
307    rdfs:isDefinedBy
308    <http://spitfire-project.eu/ontology/ns> .
309
310
311 :isProjectTopicOf rdf:type owl:ObjectProperty ;
312   rdfs:domain spt:SensorProjectTopic ;
313    rdfs:range spt:SensorNetwork  ;
314    owl:inverseOf spt:partOfProjectTopic ;
315    rdfs:isDefinedBy
316    <http://spitfire-project.eu/ontology/ns> .
317
318
319 :partOfProjectTopic rdf:type owl:ObjectProperty ;
```

```
320     rdfs:domain spt:SensorNetwork  ;
321     rdfs:range spt:SensorProjectTopic ;
322     owl:inverseOf spt:isProjectTopicOf ;
323     rdfs:isDefinedBy
324     <http://spitfire-project.eu/ontology/ns> .
325
326
327 :savedEnergy rdf:type owl:ObjectProperty ;
328     rdfs:domain dul:PhysicalObject ;
329     rdfs:range spt:SavedEnergy ;
330     owl:inverseOf spt:isSavedEnergyOf ;
331     rdfs:isDefinedBy
332     <http://spitfire-project.eu/ontology/ns> .
333
334
335 :isSavedEnergyOf rdf:type owl:ObjectProperty ;
336     rdfs:domain spt:SavedEnergy ;
337     rdfs:range dul:PhysicalObject ;
338     owl:inverseOf spt:savedEnergy ;
339     rdfs:isDefinedBy
340     <http://spitfire-project.eu/ontology/ns> .
341
342
343 :archive rdf:type owl:ObjectProperty ;
344     rdfs:domain <http://purl.org/net/provenance/ns#DataItem>  ;
345     owl:inverseOf spt:archiveOf ;
346     rdfs:isDefinedBy
347     <http://spitfire-project.eu/ontology/ns> .
348
349
350 :archiveOf rdf:type owl:ObjectProperty ;
351     rdfs:range <http://purl.org/net/provenance/ns#DataItem> ;
352     owl:inverseOf spt:archive ;
353     rdfs:isDefinedBy
354     <http://spitfire-project.eu/ontology/ns> .
355
356
357 :trigger rdf:type owl:ObjectProperty ;
358     rdfs:domain dul:Event  ;
359     rdfs:range <http://xmlns.com/foaf/0.1/Agent> ;
360     owl:inverseOf spt:triggerOf ;
361     rdfs:isDefinedBy
362     <http://spitfire-project.eu/ontology/ns> .
```

```
363
364
365 :triggerOf rdf:type owl:ObjectProperty ;
366    rdfs:range dul:Event  ;
367    rdfs:domain <http://xmlns.com/foaf/0.1/Agent> ;
368    owl:inverseOf spt:triggerOf ;
369    rdfs:isDefinedBy
370    <http://spitfire-project.eu/ontology/ns> .
371
372
373 :mood rdf:type owl:ObjectProperty ;
374    rdfs:domain spt-c:Status ;
375    rdfs:range :Mood ;
376    owl:equivalentProperty ao:mood ;
377    rdfs:isDefinedBy
378    <http://spitfire-project.eu/ontology/ns> .
379
380 :temporal a owl:ObjectProperty ;
381    rdfs:range :TemporalProperty .
382    rdfs:isDefinedBy
383    <http://spitfire-project.eu/ontology/ns> .
384
385
386
387
388 #    ////////////////////////////////
389 #    //
390 #    // Data properties
391 #    //
392 #    ////////////////////////////////
393
394 :savedEnergy rdf:type owl:DatatypeProperty ;
395    rdfs:subPropertyOf dul:hasDataValue ;
396    rdfs:domain spt:SavedEnergy  ;
397    rdfs:range xsd:double ;
398    rdfs:isDefinedBy
399    <http://spitfire-project.eu/ontology/ns> .
400
401 :message rdf:type owl:DatatypeProperty ;
402    rdfs:domain spt-c:Status ;
403    rdfs:range xsd:string ;
404    rdfs:isDefinedBy
405    <http://spitfire-project.eu/ontology/ns> .
```

Listing F.1: Part of the RDF Source Code of the SPITFIRE ontology serialized in Turtle. In this portion of the code, classes are defined.

Bibliography

[Aberer et al., 2006] Aberer, K., Hauswirth, M., and Salehi, A. (2006). A Middleware for Fast and Flexible Sensor Network Deployment. In *Proceedings of 32nd International Conference on Very Large Data Bases*.

[Abowd et al., 1999] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*.

[Abowd and Mynatt, 2000] Abowd, G. D. and Mynatt, E. D. (2000). Charting past, present and future research in ubiquitous computing. *ACM Transaction Human-Computer Interaction*.

[Adida et al., 2008] Adida, B., Birbeck, M., McCarron, S., and Pemberton, S. (2008). RDFa in XHTML: Syntax and Processing — A collection of attributes and processing rules for extending XHTML to support RDF. W3C Recommendation, W3C. http://www.w3.org/TR/rdfa-syntax/.

[AEMET, 2014] AEMET (2014). Agencia estatal de meteorologia. http://www.aemet.es.

[Aggarwal, 2008] Aggarwal, C. C. (2008). On Unifying Privacy and Uncertain Data Models. In *ICDE Conference*.

[Aggarwal, 2009] Aggarwal, C. C. (2009). *Managing and Mining Uncertain Data*. Springer.

[Agranat, 1998] Agranat, I. (1998). Engineering web technologies for embedded applications. *IEEE Internet Computing*, 2.

[Ahn and Kim, 2006] Ahn, S. and Kim, D. (2006). Proactive context-aware sensor networks. *Wireless Sensor Networks*.

[Akyildiz et al., 2002] Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38.

[Alliance, 2014] Alliance, I. (2014). Enabling the internet of things. http://www.ipso-alliance.org/.

[Anand and Roy, 2003] Anand, R. and Roy, C. (2003). A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. *Middleware 2003*, page 998.

[Ashton, 2009] Ashton, K. (2009). That internet of things thing in the real world things mat ter more than ideas. *RFID Journal*.

[Asin and Gascon, 2012] Asin, A. and Gascon, D. (2012). 50 sensor applications for a smarter world. Technical report, Libelium Comunicaciones Distribuidas.

[Atzori et al., 2010] Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: a survey. *Computer Networks*, 54.

[Badii et al., 2010] Badii, A., Crouch, M., and Lallah, C. (2010). A context-awareness framework for intelligent networked embedded systems. In *Proceedings of the Third International Conference on Advances in Human-Oriented and Personalized Mechanisms, Technologies and Services (CENTRIC)*.

[Baroni and Zamparelli, 2010] Baroni, M. and Zamparelli, R. (2010). Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-10)*.

[Beckett, 2004] Beckett, D. (2004). RDF/XML Syntax Specification (Revised). W3C Recommendation, W3C. http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[Beckett, 2007] Beckett, D. (2007). Turtle — Terse RDF Triple Language. http://www.dajobe.org/2004/01/turtle/.

[Belissent, 2010] Belissent, J. (2010). Getting clever about smart cities: new opportunities require new business models. Technical report, Forrester Research.

[Bellavista et al., 2013] Bellavista, P., Corradi, A., Fanelli, M., and Foschini, L. (2013). A survey of context data distribution for mobile ubiquitous systems. *ACM Computing Surveys*, XX.

[Bermudez et al., 2009] Bermudez, L., Delory, E., O'Reilly, T., and del Rio Fernandez, J. (2009). Ocean Observing Systems Demystified. In *Proceedings of OCEANS 2009 - Marine Technology for Our Future: Global and Local Challenges*.

[Berners-Lee, 2006a] Berners-Lee, T. (2006a). Linked Data. Design issues for the world wide web, World Wide Web Consortium. http://www.w3.org/DesignIssues/LinkedData.html.

[Berners-Lee, 2006b] Berners-Lee, T. (2006b). Notation 3. http://www.w3.org/DesignIssues/Notation3.

[Berners-Lee et al., 1994] Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., and Secret, A. (1994). The World-Wide Web. *Communications of the ACM*, 37(8).

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The Semantic Web. *Scientific American*, 284(5).

[Bimschas et al., 2011] Bimschas, D., Hasemann, H., Hauswirth, M., Karnstedt, M., Kleine, O., Kröller, A., Leggieri, M., Mietz, R., Pagel, M., Passant, A., Pfisterer, D., Römer, K., and Truong, C. (2011). SPITFIRE: Toward a Semantic Web of Things. *IEEE CommMag*.

[Bizer, 2009] Bizer, C. (2009). The Emerging Web of Linked Data. *IEEE Intelligent Systems*.

[Bizer et al., 2007] Bizer, C., Cyganiak, R., and Heath, T. (2007). How to Publish Linked Data on the Web. http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/.

[Bizer et al., 2009] Bizer, C., Heath, T., and Berners-Lee, T. (2009). Linked data - the story so far. *Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS)*.

[Bizer et al., 2010] Bizer, C., Jentzsch, A., and Cyganiak, R. (2010). State of the LOD Cloud. http://www4.wiwiss.fu-berlin.de/lodcloud/state.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. M. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of COLT*.

[Bonnet et al., 2001] Bonnet, P., Gehrke, J., and Seshadri, P. (2001). Towards sensor database systems. In *Mobile Data Management*. Springer Berlin Heidelberg.

[Botts et al., 2006] Botts, M., Percivall, G., Reed, C., and Davidson, J. (2006). OGC Sensor Web Enablement: Overview and High Level Architecture. In *Proceedings of GeoSensor Networks, 2nd International Conference, GSN 2006*.

[Boulos et al., 2011] Boulos, M. N. K., Resch, B., Crowley, D. N., Breslin, J. G., Sohn, G., Burtner, R., Pike, W. A., Jezierski, E., and Chuang, K.-Y. S. (2011). Crowdsourcing, citizen sensing and sensor web technologies for public and environmental health surveillance and crisis management: trends, OGC standards and application examples. *International Journal of Health Geographics*.

[Brock, 2001] Brock, D. L. (2001). The electronic product code (epc) a naming scheme for physical objects. Technical report, Auto-ID Center.

[Broekstra and Kampman, 2003] Broekstra, J. and Kampman, A. (2003). The SeRQL Query Language. Technical report, Aduna.

[Broering et al., 2011a] Broering, A., Echterhoff, J., Jirka, S., Simonis, I., Everding, T., Stasch, C., Liang, S., and Lemmens, R. (2011a). New generation sensor web enablement. *Sensors*, 11(3):2652–2699.

[Broering et al., 2011b] Broering, A., Remke, A., and Lasnia, D. (2011b). SenseBox A Generic Sensor Platform for the Web of Things. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*.

[Br¨oring et al., 2011] Br¨oring, A., Echterhoff, J., Jirka, S., Simonis, I., Everding, T.,

Stasch, C., Liang, S., and Lemmens, R. (2011). New Generation Sensor Web Enablement. *Sensors*.

[Brown, 1996] Brown, P. J. (1996). *The stick-e document: a framework for creating context-aware applications*. Electronic Publishing.

[Burrell et al., 2004] Burrell, J., Brooke, T., and Beckwith, R. (2004). Vineyard computing: sensor networks in agricultural production. *IEEE Pervasive Computing*.

[Bychkovskiy et al., 2003] Bychkovskiy, V., Megerian, S., Estrin, D., and Potkonjak, M. A. (2003). Collaborative approach to in-place sensor calibration. In *IPSN Conference*.

[Caceres and Friday, 2012] Caceres, R. and Friday, A. (2012). Ubicomp systems at 20: progress, opportunities, and challenges. *IEEE Pervasive Computing*.

[Capra et al., 2003] Capra, L., Emmerich, W., and Mascolo, C. (2003). Carisma: context-aware reflective middleware system for mobile applications. *IEEE Transactions on Software Engineering*.

[Carvalho et al., 2014] Carvalho, D., Call, C., Freitas, A., and Curry, E. (2014). Easyesa: A low-effort infrastructure for explicit semantic analysis. In *Demonstration Paper in Proceedings of the 13th International Semantic Web Conference (ISWC)*.

[Castelli et al., 2009] Castelli, G., Mamei, M., Rosi, A., and Zambonelli, F. (2009). Extracting high-level information from location data: the w4 diary example. *Mobile Network Applications*.

[Chatzigiannakis et al., 2012] Chatzigiannakis, I., Hasemann, H., Karnstedt, M., Kleine, O., Kroeller, A., Leggieri, M., Pfisterer, D., Roemer, K., and Truong, C. (2012). True self-configuration for the iot. In *IEEE IoT Challenge Competition at the Internet of Things International Conference for Industry and Academia*.

[Chaves and Decker, 2010] Chaves, L. W. F. and Decker, C. (2010). A survey on organic smart labels for the internet-of-things. In *Proceedings of the Seventh International Conference on Networked Sensing Systems (INSS)*.

[Chellappa et al., 2008] Chellappa, R., Subrahmanian, V. S., and Udrea, O. (2008). Machine Recognition of Human Activities: A survey. *IEEE Circuits and Systems Society*.

[Chen et al., 2008] Chen, G., Li, M., and Kotz, D. (2008). Data-centric middleware for context-aware pervasive computing. *Pervasive Mobile Computing*.

[Chen et al., 2004] Chen, H., Finin, T., and Joshi, A. (2004). An ontology for context-aware pervasive computing environments. *Knowledge Engineering Review*, 18(3).

[Chen et al., 2007] Chen, M. H. G., Yau, N., and Estrin, D. (2007). Sharing sensor network data. Technical report, CENS.

[Chen et al., 2010] Chen, Y., Guo, J., and Hu, X. (2010). The research of internet of things supporting technologies which face the logistics industry. In *Proceedings of the International Conference on Computational Intelligence and Security (CIS)*.

[Cherniack et al., 2003] Cherniack, M., Balakrishnan, H., and Balazinska, M. (2003). Scalable distributed stream processing. In *Proceedings of the CIDR Conference*.

[Chong et al., 2011] Chong, G., Zhihao, L., and Yifeng, Y. (2011). The research and implement of smart home system based on internet of things. In *Proceedings of the International Conference on Electronics Communications and Control (ICECC)*.

[Chong et al., 2009] Chong, S. K., McCauley, I., Loke, S. W., and Krishnaswamy, S. (2009). Context-aware sensors and Data Muling. In *9th International Conference on Intelligent Transport System Telecommunications*.

[Compton et al., 2012a] Compton, M., Barnaghi, P., Bermudez, L., Castro, R. G., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W. D., Phuoc, D. L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., and Taylor, K. (2012a). The SSN Ontology of the Semantic Sensor Networks Incubator Group. *JWS*.

[Compton et al., 2012b] Compton, M., Barnaghi, P., Bermudez, L., Garca-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W. D., Phuoc, D. L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., and Taylor, K. (2012b). The {SSN} ontology of the {W3C} semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17(0):25 – 32.

[Compton et al., 2009] Compton, M., Neuhaus, H., Tran, K.-N., and Taylor, K. (2009). Reasoning about Sensors and Compositions. In *2nd International Workshop on Semantic Sensor Networks, at 8th International Semantic Web Conference*, volume 522 of *CEUR Workshop Proceedings*, pages 33–48. CEUR-ws.org.

[Corp., 2014] Corp., I. (2014). The wireless identification and sensing platform (wisp). http://wisp5.wikispaces.com/.

[Craven et al., 1998] Craven, M., DiPasquo, D., Freitag, D., Mitchell, A. M. T. M., Nigam, K., and Slattery, S. (1998). Learning to extract symbolic knowledge from the world wide web. In *Proceedings of AAAI-98*.

[Cruz and Xiao, 2003] Cruz, I. F. and Xiao, H. (2003). The role of ontologies in data integration. *Journal of Engineering Intelligent Systems*.

[CTI et al., 2011] CTI, NUIG, and TUBS (2011). Semantic entities framework and concepts. Deliverable 3.1.

[de Freitas et al., 2005] de Freitas, R., Neto, B., and da Graca Campos Pimentel, M. (2005). Toward a domain-independent semantic model for context-aware computing. http://dx.doi.org/10.1109/LAWEB.2005.43.

[Delin, 2001] Delin, K. (2001). The Sensor Web: A Macro-Instrument for Coordinated Sensing. In *Sensors*.

[Delin et al., 1999] Delin, K., Jackson, S., and Some, R. (1999). Sensor Webs. *NASA Tech. Briefs*.

[Deshpande et al., 2004] Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J., and Hong, W. (2004). Model-driven data acquisition in sensor networks. In *VLDB*.

[Dey, 2001] Dey, A. (2001). Understanding and Using Context. In *Proceedings of Personal and Ubiquitous Computing*.

[Dey and Abowd, 2000] Dey, A. K. and Abowd, G. D. (2000). Towards a better understanding of context and context-awareness. In *Human Factors Computation System Workshop*.

[Dey et al., 2001] Dey, A. K., Abowd, G. D., and Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*.

[Dohr et al., 2010] Dohr, A., Modre-Opsrian, R., Drobics, M., Hayn, D., and Schreier, G. (2010). The internet of things for ambient assisted living. In *Proceedings of the Seventh International Conference on Information Technology: New Generations (ITNG)*.

[Duquennoy et al., 2009] Duquennoy, S., Grimaud, G., and Vandewalle, J. J. (2009). Smews: Smart and mobile embedded web server. In *Proceedings of the Complex, Intelligent and Software Intensive Systems (CISIS 09)*.

[Ejigu et al., 2007] Ejigu, D., Scuturici, M., and Brunie, L. (2007). Semantic approach to context management and reasoning in ubiquitous context-aware systems. In *Proceedings of the 2nd International Conference on Digital Information Management*.

[E.M. et al., 2004] E.M., T., S., I., and K., L. (2004). Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive Computing*.

[Etzioni et al., 2004] Etzioni, O., Cafarella, M., Downey, D., Popescu, A. M., Soderland, T. S. S., Weld, D., and Yates, A. (2004). Methods for domain-independent information extraction from the web: An experimental comparison. In *Proceedings of AAAI-04*.

[Fernyhough et al., 2000] Fernyhough, J. H., Cohn, A. G., and Hogg, D. (2000). Constructing qualitative event models automatically from video input. *Image and Vision Computing*.

[Fielding, 2000] Fielding, R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine.

[Firner et al., 2011] Firner, B., Moore, R. S., Howard, R., Martin, R. P., and Zhang, Y. (2011). Poster: Smart buildings sensor networks and the internet of things. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*.

[Forecasting, 2011] Forecasting, B. R. M. (2011). Sensors: Technologies and global markets. Technical report, BCC Research.

[F.Paganelli and Giuli, 2011] F.Paganelli and Giuli, D. (2011). An Ontology-Based System for Context-Aware and Configurable Services to Support Home-Based Continuous Care. *IEEE Transactions on Information Technology in Biomedicine*.

[Frank et al., 2012] Frank, B., Shelby, Z., Hartke, K., and Bormann, C. (2012). Constrained Application Protocol (CoAP).

[Franklin and Flachsbart, 1998] Franklin, D. and Flachsbart, J. (1998). All gadget and no representation makes jack a dull environment. In *Proceedings of the AAAI Spring Symposium on Intelligent Environments*.

[Franklin et al., 2005] Franklin, M. J., Jeffery, S. R., Krishnamurthy, S., and Reiss, F. (2005). Design considerations for high fan-in systems: The hifi approach. In *Proceedings of the 2nd CIDR Conference*, pages 290–304.

[Freitas et al., 2011] Freitas, A., Curry, E., Oliveira, J. G., and O'Ryain, S. (2011). A distributional structured semantic space for querying RDF data. *International Journal of Semantic Computing*.

[Gabrilovich and Markovitch, 2007] Gabrilovich, E. and Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proc. of the 20th Intl. Joint Conf. on Artificial Intelligence*.

[Gangemi, 2010] Gangemi, A. (2010). Ontology:DOLCE+DnS Ultralite. http://ontologydesignpatterns.org/wiki/Ontology:DOLCE%2BDnS_Ultralite.

[Garlan et al., 2002] Garlan, D., Siewiorek, D., Smailagic, A., and Steenkiste, P. (2002). Project aura: Toward distraction-free pervasive computing. *IEEE Pervasive Computing*.

[Gayathri et al., 2014] Gayathri, K. S., Elias, S., and Shivashankar, S. (2014). An ontology and pattern clustering approach for activity recognition in smart environments. In *Proceedings of the Third International Conference on Soft Computing for Problem Solving Advances in Intelligent Systems and Computing*.

[Ghosh and Das, 2008] Ghosh, A. and Das, S. (2008). Coverage and connectivity issues in wireless sensor networks: a survey. *Pervasive and Mobile Computing*, 4.

[Gibbons et al., 2003a] Gibbons, P., Karp, B., Ke, Y., Nath, S., and Seshan, S. (2003a). Irisnet: An Architecture for a Worldwide Sensor Web. In *IEEE Pervasive Computing*.

[Gibbons et al., 2003b] Gibbons, P. B., Karp, B., Ke, Y., Nath, S., and Seshan, S. (2003b). Irisnet: An architecture for a worldwide sensor web. *IEEE ComSoc*.

[Giusto et al., 2010] Giusto, D., Iera, A., Morabito, G., and Atzori, L. (2010). *The Internet of Things*. Springer.

[Gluhak and Schott, 2007] Gluhak, A. and Schott, W. (2007). A wsn system architecture to capture context information for beyond 3g communication systems. In *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP)*.

[Gonzalez et al., 2006] Gonzalez, H., Han, J., Li, X., and Klabjan, D. (2006). Warehousing and Analyzing Massive RFID Data Sets. In *ICDE Conference*.

[Gordon et al., 2011] Gordon, D., Hanne, J.-H., Berchtold, M., Miyaki, T., and Beigl, M. (2011). Recognizing Group Activities using Wearable Sensors. In *Mobiquitous 2011*.

[Grant and Beckett, 2004] Grant, J. and Beckett, D. (2004). RDF test cases. W3C Recommendation, W3C. http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/.

[Grefenstette and Sadrzadeh, 2011] Grefenstette, E. and Sadrzadeh, M. (2011). Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-11)*.

[Gu et al., 2005] Gu, T., Pung, H. K., and Zhang, D. Q. (2005). A service-oriented middleware for building context-aware services. *Joournal of Network Computing Applications*.

[Gu et al., 2004] Gu, T., Wang, X. H., Pung, H. K., and Zhang, D. Q. (2004). An ontology-based context model in intelligent environments. In *Proceedings of CNDS*.

[Guillemin and Friess, 2009] Guillemin, P. and Friess, P. (2009). Internet of things strategic research roadmap. Technical report, Cluster of European Research Projects on the Internet of ThingsCERP IoT.

[Guinard et al., 2010] Guinard, D., Fischer, M., and Trifa, V. (2010). Sharing using social networks in a composable web of things. In *Proceedings of the 1st IEEE International Workshop on the Web of Things (WoT 2010) at IEEE PerCom 2010*, Mannheim, Germany.

[Guinard and Trifa, 2009] Guinard, D. and Trifa, V. (2009). Towards the Web of Things: Web Mashups for Embedded Devices. In *Proceedings of WWW Conference 2009*.

[Guinard et al., 2011] Guinard, D., Trifa, V., Mattern, F., and Wilde, E. (2011). From the internet of things to the web of things: Resource oriented architecture and best practices. *Architecting the Internet of Things*.

[Gupta and Srivasatava, 2004] Gupta, A. and Srivasatava, M. (2004). Developing auto-id solutions using sun java system rfid software. Technical report, Oracle Technology Network.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*.

[Harris and Shadbolt, 2005] Harris, S. and Shadbolt, N. (2005). SPARQL query processing with conventional relational database systems. In *Web Information Systems Engineering WISE 2005 Workshops*.

[Hasemann et al., 2013] Hasemann, H., Kleine, O., Kroeller, A., Leggieri, M., and Pfisterer, D. (2013). Annotating Real-World Objects using Semantic Entities. In *Proceedings of the 10th European Conference on Wireless Sensor Networks*.

[Hasemann et al., 2012] Hasemann, H., Krller, A., and Pagel, M. (2012). Rdf provisioning for the internet of things. In *Proceedings of the Internet of Things 2012 International Conference (IoT 2012)*.

[Hayes and Gutierrez, 2004] Hayes, J. and Gutierrez, C. (2004). Bipartite graphs as intermediate model for RDF. In *Proceedings of the International Semantic Web Conference*.

[Heinzelman et al., 2004] Heinzelman, W. B., Murphy, A. L., Carvalho, H. S., and Perillo, M. A. (2004). Middleware to support sensor network applications. *IEEE Network*, 18.

[Holohan and Schukat, 2010] Holohan, E. and Schukat, M. (2010). Authentication using virtual certificate authorities - a new security paradigm for wireless sensor networks. In *Proceedings of the 9th IEEE International Symposium on Network Computing and Applications (IEEE NCA10)*.

[Honle et al., 2005] Honle, N., Kappeler, U. P., Nicklas, D., Schwarz, T., and Grossmann, M. (2005). Benefits of Integrating Meta Data into a Context Model. In *Proceedings of the Pervasive Computing and Communications Workshops at the Third IEEE International Conference on Pervasive Computing*.

[Howe, 2006] Howe, J. (2006). *The Rise of Crowdsourcing*. Wired.

[Huebsch et al., 2005] Huebsch, R., Chun, B., Hellerstein, J. M., Loo, B. T., Maniatis, P., Roscoe, T., Shenker, S., Stoica, I., and Yumerefendi, A. R. (2005). The architecture of pier: an internet-scale query processor. In *Proceedings of the CIDR Conference*.

[Huebsch et al., 2003] Huebsch, R., Hellerstein, J. M., Lanham, N., and Thau, B. (2003). Querying the internet with pier. In *Proceedings of the 29th VLDB Conference*.

[Hull et al., 1997] Hull, R., Neaves, P., and Bedford-Roberts, J. (1997). Towards situated computing. In *Proceedings of the First International Symposium on Wearable Computers*.

[Huynh, 2008] Huynh, D. T. G. (2008). *Human activity recognition with wearable sensors*. Ph.D. Thesis. TU Darmstadt, Germany.

[iAnywhere Inc., 2004] iAnywhere Inc., S. (2004). White paper: Manage data successfully with rfid anywhere edge processing. Technical report, Sybase.

[Idehen, 2008] Idehen, K. (2008). In perpetual pursuit of context. http://www.openlinksw.com/blog/~kidehen/?id=1356&title=In%20Perpetual%

20Pursuit%20of%20Context.

[Inc., 2012] Inc., G. (2012). Gartner's hype cycle special report for 2011. http://www.gartner.com/technology/research/hype-cycles/.

[Inc., 2013] Inc., G. (2013). Google trends. http://www.google.com/trends.

[Inc., 2011] Inc., M. (2011). Mongodb.

[Institutes, 2011] Institutes, C. (2011). Smart networked objects and internet of things. Technical report, Carnot Institutes' Information Communication Technologies and Micro Nano Technologies alliance.

[Issarny et al., 2007] Issarny, V., Caporuscio, M., and Georgantas, N. (2007). A perspective on the future of middleware-based software engineering. *IEEE Computer Society*.

[Jacob et al., 2006] Jacob, C., Linner, D., Steglich, S., and Radusch, I. (2006). Bio-inspired context gathering in loosely coupled computing environments. *Bio-Inspired Models of Network Information and Computing Systems*.

[Jadhav et al., 2010] Jadhav, A., Purohit, H., Kapanipathi, P., Ananthram, P., Ranabahu, A., Nguyen, V., Mendes, P. N., Smith, A. G., Cooney, M., and Sheth, A. (2010). Twitris 2.0 : Semantically Empowered System for Understanding Perceptions From Social Data. In *ISWC Semantic Web Challenge 2010*.

[Jeffrey et al., 2006a] Jeffrey, S. R., Alonso, G., Franklin, M., Hong, W., and Widom, J. (2006a). A pipelined framework for online cleaning of sensor data streams. In *ICDE Conference*.

[Jeffrey et al., 2006b] Jeffrey, S. R., Alonso, G., Franklin, M., Hong, W., and Widom, J. (2006b). Declarative Support for RFID Data Cleaning. In *Pervasive Conference*.

[Jeffrey et al., 2006c] Jeffrey, S. R., Garofalakis, M., and Franklin, M. J. (2006c). Adaptive Cleaning for RFID Data Streams. In *VLDB Conference*.

[Jirka et al., 2009a] Jirka, S., Broering, A., and Stasch, C. (2009a). Discovery mechanisms for the sensor web. *Sensors*, 9.

[Jirka et al., 2009b] Jirka, S., Broring, A., and Stasch, C. (2009b). Discovery Mechanisms for Sensor Web. In *Sensors*.

[Juels, 2004] Juels, A. (2004). Minimalist Cryptography for RFID Tags. In *Conference on Security in Communication Networks*.

[Juels and Brainard, 2004] Juels, A. and Brainard, J. (2004). Soft Blocking: Flexible Blocker Tags on the Cheap. In *Workshop on Privacy in the Electronic Society (WPES 04)*.

[Juels et al., 2003] Juels, A., Rivest, R., and Szydlo, M. (2003). The Blocker Tag: Selective Blocking of RFID tags for Consumer Privacy. In *ACM Conference on*

Computer and Communication Security.

[Kagal et al., 2003] Kagal, L., Finin, T., and Joshi, A. (2003). A Policy-based Approach to Security for the Semantic Web. In *Proceedings of the International Semantic Web Conference*.

[Kagal et al., 2004] Kagal, L., Paolucci, M., Srinivasan, N., Denker, G., Finin, T., and Sycara, K. (2004). Authorization and Privacy for Semantic Web Services. *IEEE Intelligent Systems*, 19.

[Kalkofen et al., 2013] Kalkofen, D., Veas, E., Zollmann, S., Steinberger, M., and Schmalstieg, D. (2013). Adaptive Ghosted Views for Augmented Reality. In *IEEE International Symposium on Mixed and Augmented Reality* .

[Karvounarakis et al., 2002] Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., and Scholl, M. (2002). RQL: A Declarative Query Language for RDF. In *Proceedings of the International World-Wide Web Conference (WWW2002)*.

[Katasonov et al., 2008] Katasonov, A., Kaykova, O., Khriyenko, O., Nikitin, S., and Terziyan, V. (2008). Smart semantic middleware for the internet of things. In *Proceedings of the ICINCO Conference*.

[Khoussainova et al., 2006] Khoussainova, N., Balazinska, M., and Suciu, D. (2006). Towards Correcting Input Data Errors Probabilistically Using Integrity Constraints. In *Fifth International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE06)*.

[Kim, 2008] Kim, H. (2008). Protection against packet fragmentation attacks at 6lowpan adaptation layer. In *Proceedings of the International Conference on Convergence and Hybrid Information Technology*.

[Kim et al., 2005] Kim, Y., Kim, B., Lee, J., and Lim, H. (2005). The path index for query processing on RDF and RDF Schema. In *ICACT Conference*.

[Kim et al., 2008] Kim, Y.-T., Jeong, Y.-S., Hwang, J.-H., Lee, K.-J., Tolentino, R. S., Lee, S.-H., and Park, G.-C. (2008). A design and implementation of usn-based mobile web services framework. In *Proceedings of the 2nd IEEE International Conference on Future Generation Communication and Networking (FGCN)*.

[Kinoshita et al., 2004] Kinoshita, S., Hoshino, F., Komuro, T., Fujimura, A., and Ohkubo, M. (2004). Low-cost RFID privacy protection scheme. *IPS Journal*, 45.

[Kleine et al., 2015] Kleine, O., Ebers, S., and Leggieri, M. (2015). Monitoring urban traffic using semantic web services on smartphones - a case study. In *Proceedings of the IEEE SECON 2015, SWANSITY - 2nd Workshop on Smart Wireless Access Networks for Smart City*.

[Klyne and Carroll, 2004] Klyne, G. and Carroll, J. J. (2004). Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, W3C.

http://www.w3.org/TR/rdf-concepts/.

[Ko and Sim, 2008] Ko, K.-E. and Sim, K.-B. (2008). Development of context aware system based on bayesian network driven context reasoning method and ontology context modeling. In *Proceedings of the International Conference on Control, Automation and Systems*.

[Kolb, 2008] Kolb, D. (2008). *Sprawling Place*. University of Georgia Press.

[Korel and Koo, 2010] Korel, B. T. and Koo, S. G. M. (2010). A Survey on Context-Aware Sensing for Body Sensor Networks. *Wireless Sensor Network*.

[Kortuem et al., 2010] Kortuem, G., Kawsar, F., Fitton, D., and Sundramoorthy, V. (2010). Smart objects as building blocks for the internet of things. *IEEE Internet Computing*, 14.

[Kwon et al., 2010] Kwon, O., Song, Y. S., Kim, J. H., and Li, K.-J. (2010). Sconstream: A spatial context stream processing system. In *Proceedings of the International Conference on Computational Science and Its Applications*.

[Kwon et al., 2014] Kwon, Y., Kang, K., and Bae, C. (2014). Unsupervised learning for human activity recognition using smartphone sensors. *Expert Systems with Applications*, 41.

[Labs, 2012] Labs, S. R. L. (2012). Future retail center. http://www.sap.com/corporate-en/our-company/innovation/research/livinglabs/futureretail/index.epx.

[Landauer and Dumais, 1997] Landauer, T. K. and Dumais, S. T. (1997). A solution to platos problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*.

[Lassila and Swick, 1999] Lassila, O. and Swick, R. R. (1999). Resource Description Framework (RDF) Model and Syntax Specification. W3C Recommendation, W3C. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[Le-Phuoc et al., 2009] Le-Phuoc, D., Polleres, A., Hauswirth, M., Tummarello, G., and Morbidoni, C. (2009). Rapid prototyping of semantic mash-ups through semantic web pipes. In *Proceedings of the 18th international conference on World wide web*.

[Lee et al., 2001] Lee, T., Hendler, J., Lassila, O., et al. (2001). The semantic web. *Scientific American*, 284(5):34–43.

[Leggieri et al., 2015a] Leggieri, M., Davis, B., and Breslin, J. (2015a). Distributional Semantics and Unsupervised Clustering for Sensor Relevancy Prediction. In *Proceedings of the 11th IEEE International Wireless Communication and Mobile Computing Conference (IWCMC)*.

[Leggieri and Hausenblas, 2014] Leggieri, M. and Hausenblas, M. (2014). Interoperability of two RESTful protocols: HTTP and CoAP.

[Leggieri et al., 2013a] Leggieri, M., Karnstedt, M., and Hauswirth, M. (2013a). SPITFIRE Deliverable D2.4.

[Leggieri and Parreira, 2013] Leggieri, M. and Parreira, J. X. (2013). GAMBAS Deliverable D4.1.1.

[Leggieri et al., 2010a] Leggieri, M., Passant, A., and Hauswirth, M. (2010a). A contextualised cognitive perspective for linked sensor data. In *Proceedings of the Semantic Sensor Network Workshop at the International Semantic Web Conference (ISWC)*.

[Leggieri et al., 2010b] Leggieri, M., Passant, A., and Hauswirth, M. (2010b). A contextualised cognitive perspective for linked sensor data - short paper. In *Proceedings of the 3nd International Workshop on Semantic Sensor Networks Workshop at ISWC2010*.

[Leggieri et al., 2011a] Leggieri, M., Passant, A., and Hauswirth, M. (2011a). inContextSensing: LOD augmented sensor data. In *Proceedings of the Posters and Demo session at the International Semantic Web Conference (ISWC)*.

[Leggieri et al., 2011b] Leggieri, M., Passant, A., and Hauswirth, M. (2011b). inContextSensing: LOD augmented sensor data. In *ISWC Posters and Demos*.

[Leggieri et al., 2012a] Leggieri, M., Passant, A., Karnstedt, M., and Hauswirth, M. (2012a). SPITFIRE Deliverable D2.1.

[Leggieri et al., 2012b] Leggieri, M., Serrano, M., and Hauswirth, M. (2012b). Data modeling for cloud-based internet-of-things systems. In *Proceedings of the IEEE International Conference on Internet of Things 2012*.

[Leggieri et al., 2015b] Leggieri, M., von der Weth, C., and Breslin, J. (2015b). Using Sensors to Bridge the Gap between Real Places and their Web-Based Representations. In *Proceedings of the 10th IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*.

[Leggieri et al., 2013b] Leggieri, M., von der Weth, C., and Serrano, M. (2013b). OpenIoT Deliverable D3.1.2.

[Lemahieu, 1999] Lemahieu, W. (1999). Mesh: an object-oriented approach to hypermedia modeling and navigation. In *Proceedings of Informatiewetenschap*.

[Levy, 1999] Levy, P. (1999). *Collective intelligence: Mankind's emerging world in cyberspace*. Perseus Publishing.

[Lewis and Gale, 1994] Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94*.

[Li and Dustdar, 2011] Li, F. and Dustdar, S. (2011). Incorporating unsupervised learning in activity recognition. In *AAAI workshops at the 25th AAAI conference on artificial intelligence*.

[Li et al., 2010] Li, F., Sehic, S., and Dustdar, S. (2010). Copal: An adaptive approach

to context provisioning. In *In 6th International Conference on Wireless and Mobile Computing, Networking and Communications*. IEEE Computer Society.

[Li et al., 2008] Li, Z., Zhou, X., Qing, H., and Li, S. (2008). Model and implementation of context-aware sensor networks. In *Proceedings of the International Symposium on Information Science and Engineering*.

[Lih, 2009] Lih, A. (2009). *The Wikipedia Revolution: How a Bunch of Nobodies Created the World's Greatest Encyclopedia*. Hyperion.

[Lim and Dey, 2010] Lim, B. Y. and Dey, A. K. (2010). Toolkit to support intelligibility in context-aware applications. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*.

[Lin, 2011] Lin, L. (2011). Application of the internet of thing in green agricultural products supply chain management. In *Proceedings of the International Conference on Intelligent Computation Technology and Automation (ICICTA)*.

[Lin et al., 2004] Lin, T., Zhao, H., Wang, J., Han, G., and Wang, J. (2004). An embedded web server for equipment. In *Proceedings. 7th International Symposium on Parallel Architectures, Algorithms and Networks*.

[Loke, 2009] Loke, S. W. (2009). Incremental awareness and compositionality: A design philosophy for context-aware pervasive systems. In *Pervasive and Mobile Computing*.

[Lu and Neng, 2010] Lu, T. and Neng, W. (2010). Future internet: The internet of things. In *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*.

[Lucia et al., 2008] Lucia, A. D., Francese, R., Passero, I., and Tortora, G. (2008). SLMeeting: supporting collaborative work in Second Life. In *Working Conference on Advanced Visual Interfaces*.

[Lund and Burgess, 1996] Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers*.

[Madden et al., 2003] Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. (2003). The design of an acquisitional query processor for sensor networks. In *Proceedings of the SIGMOD Conference*.

[Mainwaring et al., 2002] Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., and Anderson, J. (2002). Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*.

[Malan et al., 2004] Malan, D., Fulford-jones, T., Welsh, M., and Moulton, S. (2004). Codeblue: An ad hoc sensor network infrastructure for emergency medical care. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*.

[Malhorta et al., 2012] Malhorta, A., Hausenblas, M., and Herman, I. (2012). W3c rdb2rdf working group. http://www.w3.org/2001/sw/rdb2rdf/.

[Mayer and Guinard, 2011] Mayer, S. and Guinard, D. (2011). An extensible discovery service for smart things. In *Proceedings of the 2nd International Workshop on the Web of Things*.

[Mccrae et al., 2012] Mccrae, J., Aguado-De-Cea, G., Buitelaar, P., Cimiano, P., Declerck, T., Gómez-Pérez, A., Gracia, J., Hollink, L., Montiel-Ponsoda, E., Spohr, D., and Wunner, T. (2012). Interchanging lexical resources on the semantic web. *Lang. Resour. Eval.*, 46(4):701–719.

[Miluzzo et al., 2008] Miluzzo, E., Lane, N. D., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S. B., Zheng, X., and Campbell, A. T. (2008). Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application. In *in Proceedings of the International Conference on Embedded Networked Sensor Systems (SenSys*.

[Montenegro et al., 2007] Montenegro, G., Kushalnagar, N., Hui, J., and Culler, D. (2007). Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard).

[Moodley and Simonis, 2006] Moodley, D. and Simonis, I. (2006). A New Architecture for the Sensor Web: The SWAP Framework. In *Proceedings of 5th International Semantic Web Conference (ISWC 2006)*.

[Moses, 2012] Moses, A. (2012). Lg smart fridge tells you what to buy, cook and eat. http://www.smh.com.au/digital-life/hometech/lg-smart-fridge-tells-you-what-to-buy-cook-and-eat-20120110-1ps9z.html.

[Narayanan and McIlraith, 2002] Narayanan, S. and McIlraith, S. A. (2002). Simulation, verification and automated composition of web services. In *Proceedings of the 11th international conference on World Wide Web*.

[Nath et al., 2006] Nath, S., Liu, J., and Zhao, F. (2006). Challenges in building a portal for sensors worldwide. In *Proceedings of WSW at SenSys*.

[Nath et al., 2007] Nath, S., Liu, J., and Zhao, F. (2007). Sensormap for wide-area sensor webs. *IEEE Computer*.

[Nickerson and Lu, 2004] Nickerson, B. G. and Lu, J. (2004). A language for wireless sensor webs. In *Proceedings of the Second Annual Conference on Communication Networks and Services Research*.

[Nittel, 2009] Nittel, S. (2009). A Survey of Geosensor Networks: Advances in Dynamic Environmental Monitoring. In *Sensors*.

[Obitko, 2007] Obitko, M. (2007). Specification of conceptualization.

[of Business, 2009] of Business, G. T. G. L. (2009). The epcglobal architecture framework. http://www.epcglobalinc.org.

[of the ETP EPOSS, 2005] of the ETP EPOSS, W. G. R. (2005). Internet of things in 2020 road map for the future. Technical report, European Commission.

[OGC - Open Geospatial Consortium, 2010] OGC - Open Geospatial Consortium (2010). Sensor Web Enablement (SWE).

[Olifer and Olifer, 2005] Olifer, N. and Olifer, V. (2005). *Computer Networks: Principles, Technologies and Protocols for Network Design*. John Wiley and Son.

[Oren et al., 2008] Oren, E., Delbru, R., Catasta, M., Cyganiak, R., and Tummarello, G. (2008). Sindice.com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3.

[Ostermaier et al., 2010] Ostermaier, B., Roemer, K., Mattern, F., Fahrmair, M., and Kellerer, W. (2010). A real-time search engine for the web of things. In *Proceedings of the International Conference on Internet of Things 2010 (IoT 2010)*.

[Paek et al., 2010] Paek, J., Kim, J., and Govindan, R. (2010). Energy-efficient rate-adaptive gps-based positioning for smartphones. In *Proceedings of MobiSys 2010*.

[Page et al., 2009] Page, Kevin, R., Roure, D. C. D., Martinez, Kirk, Sadler, Jason, D., and Kit, O. Y. (2009). Linked Sensor Data: RESTfully serving RDF and GML. In *Semantic Sensor Networks workshop at ISWC (SSN09)*.

[Pan et al., 2013] Pan, J., Staab, S., Assmann, U., Ebert, J., and Zhao, Y. (2013). *Ontology-Driven Software Development*. Springer.

[Partners, 2014] Partners, . N. (2014). 52 north initiative for geospatial open source software gmbh. http://52north.org.

[Partners, 2013] Partners, E. (2013). Exalted: Expanding lte for devices. http://www.ict-exalted.eu/.

[Partners, 2011] Partners, S. (2011). Semsorgrid4env: Semantic sensor grids for rapid application development for environmental management. http://www.semsorgrid4env.eu/.

[partners all, 2011] partners all, S. (2011). Use case definition and report on the first experiment. Deliverable 4.1.

[Pascoe, 1998] Pascoe, J. (1998). Adding generic contextual capabilities to wearable computers. In *Proceedings of the Second International Symposium on Wearable Computers*.

[Patel et al., 2009] Patel, P., Jardosh, S., Chaudhary, S., and Ranjan, P. (2009). Context aware middleware architecture for wireless sensor network. In *Proceedings of the IEEE International Conference on Services Computing*.

[Patni et al., 2010] Patni, H., Henson, C., and Sheth, A. (2010). Linked Sensor Data. In *International Symposium on Collaborative Technologies and Systems (CTS 2010)*.

[Pfisterer et al., 2011a] Pfisterer, D., Bimschas, D., Kleine, O., Roemer, K., Truong, C., Mietz, R., Kroeller, A., Hasemann, H., Karnstedt, M., Passant, A., Leggieri, M., and Hauswirth, M. (2011a). Semantic-Service Provisioning for the Internet of Things. In *Proceedings of the Workshop on Semantic Services for the Internet of Things (SSIT) at Kommunikation in Verteilten Systemen*.

[Pfisterer et al., 2011b] Pfisterer, D., Roemer, K., Bimschas, D., Hasemann, H., Hauswirth, M., Karnstedt, M., Kleine, O., Leggieri, M., Kroeller, A., Mietz, R., Pagel, M., Passant, A., Richardson, R., and Truong, C. (2011b). SPITFIRE: Towards a Semantic Web of Things. *IEEE Communication magazine*.

[Philipose et al., 2004] Philipose, M., Fishkin, K. P., Perkowitz, M., Patterson, D. J., Fox, D., Kautz, H., and Haehnel, D. (2004). Inferring activities from interactions with objects. *IEEE Pervasive Computing*.

[Phuoc et al., 2014] Phuoc, D. L., Quoc, H. N. M., Ngo, Q. H., Nhat, T. T., and Hauswirth, M. (2014). Enabling live exploration on the graph of things. In *Semantic Web Challenge at the Internaltional Semantic Web conference*.

[Powazek, 2001] Powazek, D. M. (2001). *Design for Community: The Art of Connecting Real People in Virtual Places.* New Riders, a Division of Pearson Technology Group.

[Prud'hommeaux and Seaborne, 2008] Prud'hommeaux, E. and Seaborne, A. (2008). SPARQL query language for RDF. W3C Recommendation, W3C. http://www.w3.org/TR/rdf-sparql-query/.

[Pschorr et al., 2010] Pschorr, J., Henson, C., Patni, H., and Sheth, A. P. (2010). Sensor discovery on linked data. Technical report, Kno.e.sis.

[Ra et al., 2010] Ra, M. R., Paek, J., Sharma, A. B., Govindan, R., Krieger, M. H., and Neely, M. J. (2010). Energy-delay tradeoffs in smartphone applications. In *Proceedings of MobiSys 2010*.

[Ramparany et al., 2011] Ramparany, F., Benazzouz, Y., Gadeyne, J., and Beaune, P. (2011). Automated context learning in ubiquitous computing environments. In *Semantic Sensor Networks Workshop colocated with the International Semantic Web Conference*.

[Ranganathan et al., 2004a] Ranganathan, A., Al-Muhtadi, J., and Campbell, R. H. (2004a). Reasoning about uncertain contexts in pervasive computing environments. In *IEEE Pervasive Computing*.

[Ranganathan et al., 2004b] Ranganathan, A., Mcgrath, R. E., Campbell, R. H., and Mickunas, M. D. (2004b). Use of ontologies in a pervasive computing environment. *Knowledge Engineering Review*, 18(3).

[Riboni and Bettini, 2009] Riboni, D. and Bettini, C. (2009). Context-aware activity recognition through a combination of ontological and statistical reasoning. In *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing*.

[Richardson et al., 2011] Richardson, R., Hauswirth, M., Passant, A., Leggieri, M., Karnstedt, M., Kroeller, A., Hasemann, H., Kleine, O., Pfisterer, D., Roemer, K., and Truong, C. (2011). Unlocking Wireless Sensor Networks. In *Proceedings of the eChallenges e-2011 Conference*.

[Rocha et al., 2009] Rocha, A. R. D., Delicato, F. C., de Souza, J. N., Gomes, D. G., and Pirmez, L. (2009). A semantic middleware for autonomic wireless sensor networks. In *Proceedings of the International Workshop on Middleware for Ubiquitous and Pervasive Systems*.

[Rodden et al., 1998] Rodden, T., Chervest, K., Davies, N., and Dix, A. (1998). Exploiting context in hci design for mobile systems. In *Proceedings of the Workshop on Human Computer Interaction with Mobile Devices*.

[Roemer et al., 2010] Roemer, K., Ostermaier, B., Mattern, F., Fahrmair, M., and Kellerer, W. (2010). Real-Time Search for Real-World Entities: A Survey. *Proceedings of The IEEE*, 98.

[Rogers, 2006] Rogers, Y. (2006). Moving on from Weisers vision of calm computing: engaging ubicomp experiences. In *Proceedings of Ubiquitous Computing (UbiComp)*.

[Roman et al., 2002] Roman, M., Hess, C., Cerqueira, R., Ranganathan, A., Campbell, R. H., and Nahrstedt, K. (2002). A middleware infrastructure for active spaces. *IEEE Pervasive Computing*.

[Rooney et al., 2006] Rooney, S., Bauer, D., and Scotton, P. (2006). Techniques for integrating sensors into the enterprise network. *IEEE Transactions on Network and Service Management*.

[Ryan et al., 1997] Ryan, N. S., Pascoe, J., and Morse, D. R. (1997). Enhanced reality fieldwork: the context-aware archaeological assistant. *Computer Applications in Archaeology*.

[Sanchez et al., 2006] Sanchez, L., Lanza, J., Olsen, R., Bauer, M., and Girod-Genet, M. (2006). A generic context management framework for personal networking environments. In *Proceedings of the Workshop at the 3rd Annual International Conference on Mobile and Ubiquitous Systems*.

[Sang et al., 2010] Sang, Y., Shen, H., Inoguchi, Y., Tan, Y., and Xiong, N. (2010). Secure data aggregation in wireless sensor networks: A survey. *Wireless Sensor Network Technologies for the Information Explosion Era*, 278.

[Scheglmann et al., 2013] Scheglmann, S., Groener, G., Staab, S., and Laemmel, R. (2013). Incompleteness-aware programming with rdf data. In *Proceedings of the 2013 Workshop on Data Driven Functional Programming (DDFP)*, Rome, Italy.

[Scherp et al., 2009a] Scherp, A., Franz, T., Saathoff, C., and Staab, S. (2009a). F–a model of events based on the foundational ontology dolce+dns ultralight. In *K-CAP*.

[Scherp et al., 2009b] Scherp, A., Franz, T., Saathoff, C., and Staab, S. (2009b). F–a model of events based on the foundational ontology dolce+dns ultralight. In *K-CAP*.

[Scherp et al., 2011] Scherp, A., Saathoff, C., Franz, T., and Staab, S. (2011). Designing Core Ontologies. *Applied Ontology*.

[Schilit et al., 1994] Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications*.

[Schilit and Theimer, 1994] Schilit, B. and Theimer, M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*.

[Schmidt, 2008] Schmidt, A. (2008). From Sensors to Context and Activity. Tutorial.

[Scoble and Israel, 2013] Scoble, R. and Israel, S. (2013). *Age of Context: Mobile, Sensors, Data and the Future of Privacy*. CreateSpace Independent Publishing Platform.

[Seaborne, 2004] Seaborne, A. (2004). RDQL – A Query Language for RDF. W3C Member Submission, W3C.

[Shelby, 2010] Shelby, Z. (2010). Embedded web services. *Wireless Communications, IEEE*, 17(6):52 –57.

[Sheth et al., 2008] Sheth, A., Henson, C., and Sahoo, S. (2008). Semantic Sensor Web. *IEEE Internet Computing, July/August 2008*, pages 78–83.

[Sheth et al., 2013] Sheth, A. P., Barnaghi, P. M., Strohmaier, M., Jain, R., and Staab, S. (2013). Physical-Cyber-Social Computing. Technical report, Dagstuhl Reports.

[Shneidman et al., 2004] Shneidman, J., Pietzuch, P., Ledlie, J., Roussopoulos, M., Seltzer, M., and Welsh, M. (2004). Hourglass: An infrastructure for connecting sensor networks and applications. Technical report, Harvard University.

[Srisathapornphat et al., 2001] Srisathapornphat, C., Jaikaeo, C., and chung Shen, C. (2001). Sensor information networking architecture and applications. *IEEE Personal Communications*, 8.

[Stasch et al., 2009] Stasch, C., Janowicz, K., Broering, A., Reis, I., and Kuhn, W. (2009). A Stimulus-Centric Algebraic Approach to Sensors and Observations. In *In Lecture Notes in Computer Science, Proceedings of 3rd International Conference on Geosensor Networks*.

[Strang et al., 2003] Strang, T., Linnhoff-Popien, C., and Frank, K. (2003). Cool: A context ontology language to enable contextual interoperability. *Ifip International Federation For Information Processing*.

[Stratonovich, 1960] Stratonovich, R. L. (1960). Conditional Markov Processes. *Theory*

of Probability and its Applications.

[Sundmaeker et al., 2010a] Sundmaeker, H., Guillemin, P., Friess, P., and Woelffle, S. (2010a). Vision and challenges for realising the internet of things. Technical report, Cluster of European Research Projects on the Internet of ThingsCERP IoT.

[Sundmaeker et al., 2010b] Sundmaeker, H., Guillemin, P., Friess, P., and Woelfflé, S. (2010b). *Vision and Challenges for Realising the Internet of Things*. European Commission - Information Society and Media DG.

[Sutcliffe and Alrayes, 2012] Sutcliffe, A. and Alrayes, A. (2012). Investigating user experience in second life for collaborative learning. *International Journal of Human-Computer Studies*.

[Tan et al., 2008] Tan, C. C., Sheng, B., Wang, H., and Li, Q. (2008). Microsearch: When search engines meet small devices. In *Proceedings of the 6th International Conference on Pervasive Computing*.

[Tapia et al., 2004] Tapia, E. M., Intille, S. S., and Larson, K. (2004). Activity recognition in the home setting using simple and ubiquitous sensors. In *Proceedings of PERVASIVE 2004*.

[Teillet, 2008] Teillet, P. (2008). Sensor Webs: A Geostrategic Technology for Integrated Earth Sensing. In *Proceedings of International Workshop Sensing a Changing World*.

[Thrun and Mitchell, 1995] Thrun, S. and Mitchell, T. (1995). Lifelong robot learning. *Robotics and Autonomous Systems*.

[Turney and Pantel, 2010] Turney, P. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research (JAIR-10)*.

[Union, 2005] Union, I. T. (2005). Itu internet reports 2005: The internet of things. Technical report, International Telecommunication Union (ITU).

[van Bunningen et al., 2005] van Bunningen, A., Feng, L., and Apers, P. (2005). Context for ubiquitous data management. In *International Workshop on Ubiquitous Data Management*.

[van Zyl et al., 2009] van Zyl, T., Simonis, I., and McFerren, G. (2009). The Sensor Web: Systems of Sensor Systems. *International Journal on Digital Earth*, 2.

[Villalonga et al., 2010a] Villalonga, C., Bauer, M., Huang, V., Bernat, J., and Barnaghi, P. (2010a). Modeling of sensor data and context for the real world internet. In *PerCom Workshops*.

[Villalonga et al., 2010b] Villalonga, C., Bauer, M., Huang, V., Bernat, J., and Barnaghi, P. (2010b). Modeling of Sensor Data and Context for the Real World Internet. In *8th International Conference on Pervasive Computing and Communications Workshops*.

[von der Weth and Hauswirth, 2013] von der Weth, C. and Hauswirth, M. (2013). Find-

ing information through integrated ad-hoc socializing in the virtual and physical world. In *IEEE/WIC/ACM International Conference on Web Intelligence*.

[W3C RDF Data Access Working Group members, 2010] W3C RDF Data Access Working Group members (2010). SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/.

[Wang et al., 2010] Wang, H., Tan, C., and Li, Q. (2010). Snoogle: A search engine for pervasive environments. *IEEE Transactions on Parallel and Distributed Systems*, 21.

[Wang et al., 2011] Wang, Y. W., Yu, H. L., and Li, Y. (2011). Internet of things technology applied in medical information. In *Proceedings of the International Conference on Consumer Electronics Communications and Networks (CECNet)*.

[Ward et al., 1997] Ward, A., Jones, A., and Hopper, A. (1997). A new location technique for the active office. *IEEE Personal Communications*.

[Wawrzoniak et al., 2004] Wawrzoniak, M., Peterson, L., and Roscoe, T. (2004). Sophia: An information plane for networked systems. *ACM SIGCOMM Computer Communication Review*.

[Weiser, 1991] Weiser, M. (1991). The computer for the 21st century. *Scientific American*.

[Welbourne et al., 2009] Welbourne, E., Battle, L., Cole, G., Gould, K., Rector, K., Raymer, S., Balazinska, M., and Borriello, G. (2009). Building the internet of things using rfid. *IEEE Internet Computing*.

[Wilson, 2005] Wilson, D. (2005). *Assistive Intelligent Environments for Automatic Health Monitoring*. Ph.D. Thesis. Carnegie Mellon University, Pittsburgh.

[Wyatt et al., 2005] Wyatt, D., Philipose, M., and Choudhury, T. (2005). Unsupervised activity recognition using automatically mined common sense. In *In AAAI*, pages 21–27.

[Xingchen and Rajkumar, 2007] Xingchen, C. and Rajkumar, B. (2007). Service oriented sensor web. In *Sensor Networks and Configuration*. Springer Berlin Heidelberg.

[Xue et al., 2008] Xue, W., Pung, H., Ng, W., and Gu, T. (2008). Data management for context-aware computing. In *Proceedings of the IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*.

[Yap et al., 2005] Yap, K. K., Srinivasan, V., and Motani, M. (2005). Max: human-centric search of the physical world. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*.

[Zaslavsky et al., 2012] Zaslavsky, A., Perera, C., and Georgakopoulos, D. (2012). Sensing as a service and big data. In *International Conference on Advances in Cloud Computing (ACC-2012)*.

[Zhan et al., 2009] Zhan, Y., Wang, S., Zhao, Z., Chen, C., and Ma, J. (2009). A mobile

device oriented framework for context information management. In *Proceedings of the IEEE Youth Conference on Information Computing and Telecommunication*.

[Zhuang et al., 2010] Zhuang, Z., Kim, K. H., and Singh, J. P. (2010). Improving energy efficiency of location sensing on smartphones. In *Proceedings of MobiSys 2010*.

[Zorzi et al., 2010] Zorzi, M., Gluhak, A., Lange, S., and Bassi, A. (2010). From today's intranet of things to a future internet of things: a wireless- and mobility-related view. *IEEE Wireless Communications*, 17.