Semantically-enriched and semi-Autonomous collaboration framework for the Web of Things

Design, implementation and evaluation of a multi-party collaboration framework with semantic annotation and representation of sensors in the Web of Things and a case study on disaster management

Mohammad AMIR

Submitted for the Degree of

Doctor of Philosophy

Faculty of Engineering & Informatics

University of Bradford

2015

Abstract

Mohammad Amir

Semantically-enriched and semi-Autonomous collaboration framework for the Web of Things

Design, implementation and evaluation of a multi-party collaboration framework with semantic annotation and representation of sensors in the Web of Things and a case study on disaster management.

Keywords: Semantic Web, Web of Things, Multi-party Collaboration Framework, Semantic Annotation, Semantic Sensor Network Ontology (SSN), Cloud Computing, Service-oriented Architecture (SoA), Resource-based Data Model, Resource-oriented Access Control, Disaster Management.

This thesis proposes a collaboration framework for the Web of Things based on the concepts of Service-oriented Architecture and integrated with semantic web technologies to offer new possibilities in terms of efficient asset management during operations requiring multi-actor collaboration. The motivation for the project comes from the rise in disasters where effective cross-organisation collaboration can increase the efficiency of critical information dissemination. Organisational boundaries of participants as well as their IT capability and trust issues hinders the deployment of a multi-party collaboration framework, thereby preventing timely dissemination of critical data. In order to tackle some of these issues, this thesis proposes a new collaboration framework consisting of a resource-based data model, resource-oriented access control mechanism and semantic technologies utilising the Semantic Sensor Network Ontology that can be used simultaneously by multiple actors without impacting each other's networks and thus increase the efficiency of disaster management and relief operations. The generic design of the framework enables future extensions, thus enabling its exploitation across many application domains. The performance of the framework is evaluated in two areas: the capability of the access control mechanism to scale with increasing number of devices, and the capability of the semantic annotation process to increase in efficiency as

more information is provided. The results demonstrate that the proposed framework is fit for purpose.

Acknowledgements

First and foremost, I thank Allah the Almighty for giving me the capability to undertake this task and then for giving me patience and perseverance to see it to completion. Indeed, all praises are to and for Allah, the Most Beneficent, the Most Merciful.

I would then like to thank my supervisor Dr. P. Pillai for helping and guiding me through my years of study, and for constantly encouraging my work and helping me develop as a researcher. Your advice has always been appreciated.

At the same time, I would like to thank Prof. Fun Hu in whose lab I had the privilege of working on and developing the SAW framework. Your input on my publications and research work has been immensely valuable.

I would also like to thank my colleague Kirils Bibiks for lending me some of his time to help develop the client-side simulation setup for my framework.

I cannot close this chapter without thanking my family. I could not have seen this through especially without the support of my mother and my wife. Thank you for supporting me and for encouraging me and keeping me steadfast on this long journey.

Table of Contents

List of Figures

List of Tables

Abbreviations

ABAC	Authorization-Based Access Control
API	Application Programming Interface
CPPM	Cascading Permissions Policy Model
DF	Datafeeds or data feeds or feeds
DM	Disaster Management
DP	Datapoints or data points or points
DS	Datastreams or data streams or streams
GDD	Generic Device Definition
IAM	Identity & Access Management
INGO	International Governmental Organisation
IoT	Internet of Things
KM	Knowledge Management
KMS	Knowledge Management System
NGO	National Governmental Organisation
OGC	Open Geospatial Consortium
OOP	Object Oriented Programming
OSGi	Open Service Gateway initiative
OWL	Ontology Web Language
RBAC	Role-Based Access Control
RDBMS	Relational Database Management System
RDF	Resource Description Framework
REST	Representational State Transfer
SAW	Semantically-enriched and semi-Autonomous collaboration framework for the Web of Things
SGN	Sensor Gateway Node
SoA	Service-oriented Architecture
SSN	Semantic Sensor Network (Ontology)
SWE	Sensor Web Enablement
TBAC	Token-Based Access Control
UBAC	User-Based Access Control
WoT	Web of Things

Chapter 1: Introduction

1.1 Overview

This thesis investigates the feasibility of and proposes an integration of semantic technologies with the Web of Things (WoT) based on the concept and principles of the Service-Oriented Architecture (SoA) to realise a distributed and semi-autonomous collaboration framework. This framework will be tailored towards applications requiring multi-department collaboration (e.g. disaster management (DM) and relief scenarios, next-generation interactive environments (cities, airports, shopping malls, etc.)) where effective, timely and accountable asset management and information dissemination is a key requirement to the success of the mission. These situations may warrant immediate collaboration amongst heterogeneous actors, for example, DM scenarios which may arise suddenly and without notice from natural phenomena like earthquakes, floods and tornadoes; as well as from man-made situations like armed skirmishes, massacres and even large-scale wars. To account for the dynamic landscape of these events and the likelihood of massive asset deployment for administration and management purposes, the framework needs to be capable of managing and delegating information flow amongst the various actors. At the same time, the framework needs to retain the flexibility to provide the relevant data and information to an array of actors with varying interests, roles and responsibilities (e.g., police department, fire department, disease outbreak management, etc.). Due to the heterogeneity of the possible application domains, the framework will need to be generic in design but extensible in nature so that it can be tailored towards a particular application by augmenting additional functionalities, thus the modular approach adopted by

leveraging the principles of SoA. The framework from hereon in will be abbreviated to "*SAW*" which stands for "*Semantically-enriched & semi-Autonomous collaboration framework for the WoT*".

1.2 Problem Statement & Motivation

Today, web services are becoming prominent and web mashups are becoming the norm. It is now common for major corporations to offer public APIs (e.g. Facebook, Twitter, Google, Live, Yahoo, Amazon, EBay, Dropbox, GitHub, etc.) and expose their web services so that other web applications can use their data for building mashups [1]. Mashups can be defined as a compounded representation of a set of information formed by extracting and extrapolating related and linked data from other sources on the web. Examples of these mashups are today seen in offerings like GUI Widgets on Internet-of-Things (IoT) providers (e.g. Xively (formerly Cosm)), Yahoo Pipes, Google Maps and Shopping, and many others alike such as the online project collaboration systems (e.g. Redbooth). Modern webapps are starting to request an increasing amount of personal user data and "permissions" (see Figure 1-1) from other prominent service and identity providers that the user may be affiliated with. The webapps, in return for access to this enriched data, are able to provide a seamless and enhanced end-user experience. This leads to the enablement of a wide-range of services ranging from simple file-sharing with friends to a more complex and controlled process of enabling contacts to participate and collaborate in a project (e.g. Basecamp, Redbooth). All of this can now be achieved easily and readily, without having to download any software or register an additional account. Essentially, the advent of webapps is highlighting the growing value of open

data, the benefits that can be reaped from cross-organisation collaboration and data sharing, and the enhanced end-user experience that can be delivered due to the enriching of raw data when it is processed to derive meaningful and valuable information [2]. SAW envisions a similar revolution for an all-purpose collaboration framework for the WoT in the hopes that it becomes an enabler of controlled, audited, reliable and effective means of multi-department and cross-organisation information sharing.

In the current digital age, ubiquitous connectivity is fast becoming the norm and social media is integrated with every aspect of our daily lives. As this phenomenon of ubiquitous networking continues to evolve rapidly, an increasing number of people are becoming internet, technology and social-aware and IP-connected devices continue to surge in both demand and supply. In this world where microblogging thrives and users often use social media platforms as their source of information and updates surrounding issues of interest, including tragic situations like disasters, it is becoming imperative that the next generation of information exchange and collaboration frameworks adopt the principle of “open-data” and therefore breed an ecosystem where dissemination of data leads to empowerment and collaboration opportunities. This will help tackle some of the issues related to false speculation and untrue rumours that can circulate on the social networks within minutes of an incidence and cause panic, distress and unwarranted unrest or complications in the subsequent relief operations. Take, for example, a scenario involving a major flood in London, UK. Management of this type of disaster will not only involve the participation of and coordination between different emergency departments like the Police,

Fire Brigade Service, Ambulance Service, HM Coastguard, RAF Search and Rescue, etc. and thus the resultant collaboration amongst these heterogeneous responders but equally important will be the task of disseminating (timely and effectively) critical information to the general public, of which include affected people, people likely to be affected, relative and friends of those in distress, the general public and of course the media. Thus the problem here is not only of timely and controlled data dissemination and collaboration amongst the “active” actors/responders tackling to manage, contain and resolve the disaster(s) but there is also a problem of distributing useful information and updates to “passive” parties so as to inform the general masses with the correct and most up-to-date status information and the relevant procedures to undertake. There is also support for this claim in current literature, for example, in [1]. It is believed that in this setting, SAW can deliver the next-generation collaboration framework that can tie and link the somewhat closed and restricted information hubs like governmental bodies not only with other businesses who might need to make use of certain data, but also regular citizens who might experience a need to consume critical information in times of distress.

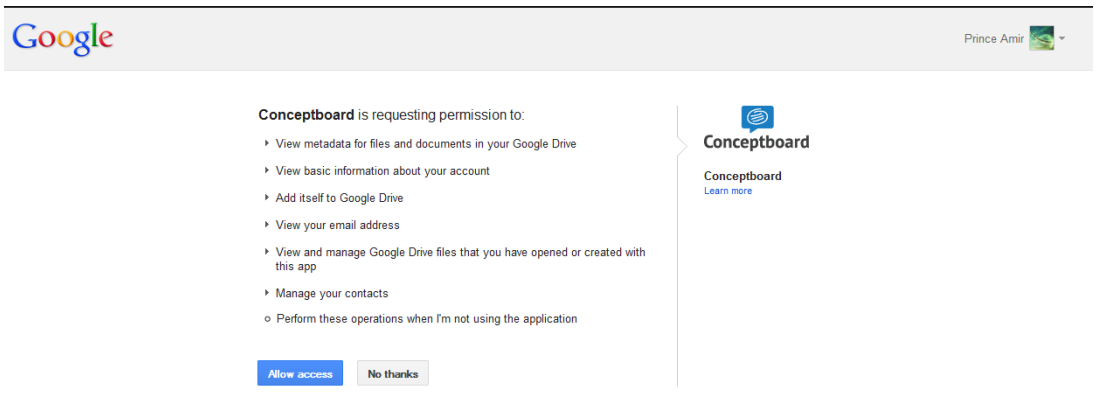


Figure 1-1: Conceptboard, a Google app, requests "permissions" before it can be used fully

While SAW has been designed as a generic and all-purpose collaboration framework, the prototype has been tailored towards a particular application domain to give the framework focus, substance and enable critical evaluation of the inner and deeper workings. We, therefore, chose to focus on the DM application domain not only because of its growing importance in the wake of increasing natural disasters, but also because of its widespread affects, the plurality of the involved actors and the heterogeneity of these very actors. This all makes for a very testing and volatile environment and a very effective means of testing and evaluating a mission-critical collaboration framework for the WoT. It is anticipated that a fully developed instance of the SAW framework will utilise cloud computing to dynamically leverage the required computing resources as per the data processing needs of the framework. However, due to shortness of time and limitation of scope, the prototype developed in this thesis is based on a simpler server-client model where the SAW framework exists on a normal PC. The utilisation of cloud computing to host the SAW framework is marked as an item for future work.

1.3 Contributed Work & Achievements

In the current literature, the existing and proposed information exchange and collaboration models suffer from one of the following deficits (these will be discussed in their respective place later on):

- Misaligned with or having no integration with social media;
- Not considering mass data gathering and analysis (i.e. not being designed for the WoT);
- Inappropriate/unsuitable data sharing mechanisms (i.e. not considering data sharing requirements beyond the scope of the

immediate framework, and thus, lack of or no support for inter-department or cross-vendor collaboration).

From the literature studied so far, the analysis for which is presented in the next section, it is concluded that there is a dire need for an all-around collaboration framework that provides a generic but extensible resource-oriented package that can capture, codify, store, process, and share not just raw data but processed information and derived knowledge. This is made possible with the power of semantics to deliver a platform-independent information exchange and collaboration framework that can be tailored for any particular domain where sensor data needs to be collected, processed and shared in a unified and standardised manner. The semantic annotation of all sensing devices and data can even enable semi-autonomacity in the system, thereby removing the need for manual processing and annotation of resources. However, this does not imply, by any means, that SAW is an all-inclusive framework, which is not the purpose of this undertaking. SAW is designed on the principle that “knowledge management is a continual process of incremental improvement and evolution – not a one-time effort” [2]. Therefore, SAW exposes an extensible resource-oriented architecture that can be easily augmented with additional functionality as and when the need arises.

SAW primarily contributes 3 main systems that help to produce an overall distributed system for the WoT domain:

1. Abstract and resource-based asset model: Enables the provisioning of multiple layers of inspection to represent assets at different levels of granularity with a clear and logical data hierarchy and generic but

extensible data templates. In current literature, this type of generic and extensible asset model which provides a low entry-barrier for potential users of the framework cannot be found. Existing solutions are either too simple, lacking semantic capability altogether, or are too complex, forcing users to adhere to strict schemas and therefore hindering acceptance;

2. Resource-based access control mechanism: An enhanced Token-Based Access Control scheme that allows distributed access to resources of any granularity and also scaling efficiently for large number of resources without projecting a noticeable impact on network performance. Existing access control mechanisms are largely role-based and therefore user-centric. However, to scale efficiently in a WoT application, resource-centric access control mechanisms are needed. SAW introduces an enhanced resource-centric access control scheme which plays nicely with the resource-based asset model while being capable of operating over resources of any granularity.
3. Service-oriented and semantic interaction model: A set of distributed resource annotation and collaboration mechanisms which enable inter-department and cross-vendor collaboration in a standard and unified manner, without forcing users to adhere to strict semantic schemas which may otherwise impact usability of the system. Existing literature is ripe with semantic efforts to define new domain-specific ontologies and interactions. However, significantly less focus has been placed on the actual semantic profiling and annotation of

resources which are to be stored in the network, and even less in reaching out to other systems and frameworks and profiling these foreign assets. SAW fills this void by enabling the capability to semi-autonomously profile and annotate resources from external networks such as Xively so that resources which are already published on the web but lack semantics can be used effectively.

1.4 Thesis Structure

The order of this thesis is as follows: This section will introduce the framework by providing a brief overview and also discuss related literature and motivation for the project. Section 2 discusses the various topics relating to collaboration models and mentions that until now a suitable solution does not exist which deals efficiently with the heterogeneity of involved actors, thus the motivation for SAW. The issue of semantic-level interoperability is also discussed and it is highlighted how integration of semantic technologies within the framework can help and aid in solving the problem of collaboration amongst a diverse array of interested parties. The current semantic efforts are highlighted and an analysis is presented on why the current efforts are not suitable for realising a semi-autonomous collaboration framework. Section 3 details the design of the SAW framework and lists all the different components that make up SAW. Section 4 then talks about the prototype implementation and lists the tools and techniques that will be used to not only implement but also test the reliability and performance of the framework. Section 5 then leads on from the framework architecture and discusses simulation models and results obtained from vigorous testing. A critical analysis of the results reveals that SAW is fit for the purpose it was designed

for. Section 6 discusses the framework validation procedure, and then section 7 mentions a real-life use case for SAW. Finally, section 8 concludes this thesis by delivering a critical evaluation of SAW and possible areas where improvements can be made.

1.5 Published Works

The following works have been published by taking material from this thesis:

- M. Amir, Y. F. Hu, P. Pillai and Y. Cheng, “Interaction Models for Profiling Assets in an Extensible and Semantic WoT Framework,” in Wireless Communication Systems (ISWCS 2013), Ilmeanu, Germany, 2013.
- M. Amir, P. Pillai and Y. Hu, “Cascading Permissions Policy Model for Token-Based Access Control in the Web of Things,” in Future Internet of Things and Cloud (FiCloud) 2014, Barcelona, 2014.
- M. Amir, P. Pillai and Y. Hu, “A Generic & Extensible Asset Model for a Semantic Collaboration Framework,” International Journal of Advanced Computer Technology (IJACT), vol. 3, no. 1, pp. 88-96, 2014.
- M. Amir, P. Pillai and Y. F. Hu, "Effective Knowledge Management using Tag-Based Semantic Annotation for Web of Things Devices," in European Conference on Knowledge Management (ECKM), Santarém, Portugal, 2014.

The following works have been submitted and are pending notification of acceptance:

- Aggregated Sensor Payload Submission Model for Token-Based Access Control in the Web of Things – Fi-Cloud 2015 Conference.

The following works are being prepared for submission to upcoming journals and conferences:

- Tag-Based Semantic Annotation Mechanism: Effects of Varying Number of Tags and Concepts to be Mapped

Chapter 2: Problem Statement & Literature Review

2.1 Collaboration Frameworks

2.1.1 Definition of Collaboration Frameworks

In the WoT, there is a strong emphasis on both the amount of data being generated and the ability to understand and derive knowledge from this data, readily, effectively and accurately. Furthermore, for the WoT to truly flourish, the data, whether its raw data coming from physical assets or derived knowledge produced through some process, needs to be exposed so that collaboration can take place. The act of collaboration with and by other actors improves the outreach and capabilities of the involved systems through enrichment of existing information and generation of further knowledge. The collection of technologies and methodologies pertaining to the enablement of the aforementioned system is termed a "collaboration framework" in this study. The purpose of a collaboration framework in the context of the WoT and as defined by this study is to: (1) Capture and represent data, (2) Generate knowledge, and (3) Share and exchange information and knowledge with external human and machine agents. Thus, a collaboration framework can be envisioned as having several components as illustrated in Figure 2-1.

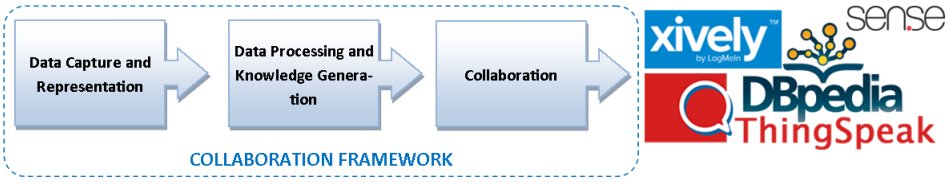


Figure 2-1: Top-level concept illustration of a collaboration framework

Data capture and representation is the first component of the framework. This is where acquisition of raw data and its modelling and representation takes place. The data will usually be modelled according to a proprietary

schema. The next step is to apply semantic contexts and business rules on the data to convert it into useful information and actionable knowledge. Finally, the processed knowledge is ready to be exposed and collaborated upon with external agents, either through an Application Programming Interface (API), or via proprietary adapters. This is illustrated further in Figure 2-2.

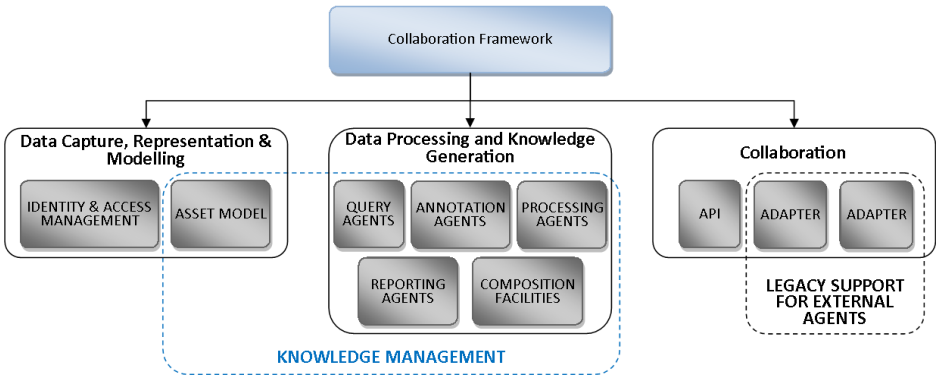


Figure 2-2: Concept architecture of a collaboration framework

2.1.1.1 Data capture, representation and modelling

Sensor data is primarily captured from local sensor networks, but it can also be fetched from repositories that expose their data through an API (e.g. Xively). The two components identified here are: (1) Asset model and (2) Identity & access management.

The asset model handles the modelling and representation of the sensing devices and data in a platform-specific manner. In other words, this means that the data pertaining to the sensing devices and their readings is stored according to a proprietary schema. The purpose of the asset model is to make this data available to the other components of the collaboration framework for further processing and enrichment. Thus, the asset model becomes a building block of a wider system known as Knowledge Management (KM), which will be discussed further in the oncoming sections.

The identity and access management component deals with the authentication and authorization of actors who want to access and interact with the data stored in the asset model. These actors can be both internal and external. Internal actors are those that reside within the organisational boundaries where the system is being operated (e.g. network administrators, instance operators). External actors are all other agents who want to interact with the system, for example, hobbyists, participating networks, autonomous agents, data mining applications and the general public at large. This idea is further illustrated in Figure 2-3. In this illustration, users of the system within organisation 1 appear as external users to the collaboration framework setup in organisation 2, and vice versa. Online repositories like Xively and DBpedia appear as external users to both systems. Collaboration takes place when the system is exposed to external entities such as other participating networks, as shown here. In this case, it is important to differentiate between internal and external users because external users are temporal whereas the internal users are more permanent. This then affects the way the system authorizes access for temporal external users. Further analysis of this problem will be provided in the section pertaining to identity and access management.

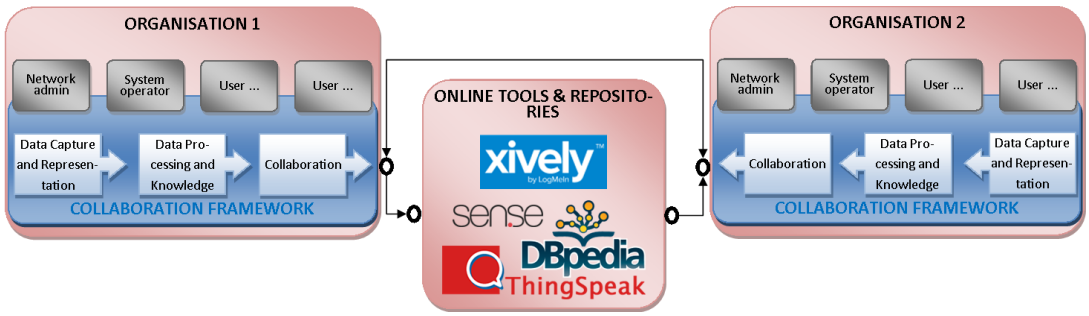


Figure 2-3: Illustration of internal and external network and users

Data may be stored in any corresponding storage medium (e.g. RDBMS (Relational Database Management System), document-based NoSQL database systems, flat-file storage systems, etc.). The stored data is codified according to a schema which is used by the internal network. The ability and ease of sharing and using the data in contexts of collaboration depends on how this data is represented when it is retrieved from storage.

2.1.1.2 Data processing and knowledge generation

Once the data has been captured from the local sensor network or imported from other repositories, this data needs to be contextualised so that it can represent some meaningful information. The process of turning raw data into useful information can take place through a variety of methods. As an example, business rules can be applied to pieces of data to generate meaningful information. For example, a numeric data value taken from a sensor can have semantic contexts applied to it so that it turns into useful information, like a radiation level or pressure value. The reading can then be given further meaning by comparing it to pre-defined thresholds (e.g. the information “radiation level is high” is generated if radiation reading is above a certain threshold). After high-level information has been generated, it might be annotated in a specific way to provide interoperability with other systems, and to enable further processing. This whole field of capturing data and then generating, annotating and making available high-level information is known as KM [3]. KM will be further expounded upon in the section pertaining to this issue.

2.1.1.3 Collaboration

Once high-level information has been generated, it can be exposed to participating networks and external actors. This act of exposing data and information leads to the enrichment of the knowledgebase for involved entities and therefore improves the capability to compose mashups and produce more meaningful reports. The process of collaboration can be enabled by developing an API and/or legacy adapters. The API can expose information that is represented in either a proprietary or an interoperable fashion. The legacy or platform-specific adapters can be used to collaborate with systems which use a proprietary schema and are therefore not interoperable.

2.1.2 Scope of Study

This study aims to develop a collaboration framework which is generic in nature so that it can support any type of sensor, and therefore any type of WoT application. However, the prototype developed, discussed and analysed in this study is catered towards one particular application domain to provide focus and effective extraction of data management needs. The study has chosen the application domain of DM as a potential scenario for evaluating the performance of the developed prototype. DM has been chosen both because of its growing impact in the world and the extreme data capture, modelling and collaboration needs inherent in this application. This does not mean, however, that the developed prototype is limited to this one application domain. As is illustrated in Figure 2-4, the application domain for the WoT is diverse, ranging from applications for smart environments to industrial control to managing logistics. The underlying functionalities within

each of these applications are similar. What differs is the actual data management needs in each scenario, but the principle of modelling data and exposing it for wider consummation is present throughout the entire application domain. Since DM touches a whole variety of applications ranging from logistics and tracking through to control and automation, it is an ideal scenario to evaluate the feasibility of a collaboration framework for the WoT.

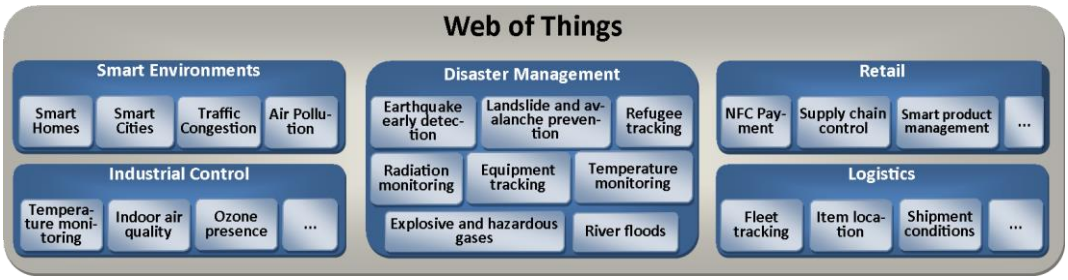


Figure 2-4: Illustration of potential application domain for the WoT

2.1.3 Challenges in Disaster Management

There are many publications available today that highlight the imminent danger from natural disasters due to their increasing frequency and level of damage. This may be due to global climate change, growing population, risky and hazardous energy extraction or simply as a result of more people populating areas of risk [4] [5]. Regardless of the actual means, these disasters cause mass catastrophes that bring with them a large number of casualties, loss of wealth and livelihood, and costly and complex search and rescue efforts.

The UN/ISDR defines a disaster as "*A serious disruption of the functioning of a community or a society involving widespread human, material, economic or environmental losses and impacts, which exceeds the ability of the affected community or society to cope using its own resources."* [6]. Whether these disasters are caused naturally or by manual conflicts and intervention, what

is important is the speedy resolution and effective management of the search and rescue and asset management process during the relief operations. There is ample research material available for disaster prevention and emergency planning, for example, [7] [8] [9] [10] [11]. The problems in this area are *hazard assessment* and consequently risk reduction where hazard assessment entails *identification*, assessment and *monitoring* of hazards. There has also been considerable research in regards to disaster inventory management and goods distribution over the past decade and to this day, for example, [12] [13] [14]. In this regard, the outlining problem is the effective *tracking* of assets alongside logistical planning while storage is a secondary concern. Aside from the problems outlined above, there is the major problem of *integration* and *collaboration* simply due to the myriad of interested and involved parties (e.g., national and international aid agencies, International and National Government Organisations (INGOs/NGOs), national emergency management and government personnel, volunteers, local businesses, etc.) each with their own organisational boundaries, fiscal constraints, working practices, technological capabilities and accessible areas [15] [16]. The differences in cultural and organisational policies as well as conflicting priorities and variations in mission goals and operating constraints further complicates the coordination and collaboration process; the result of which can be lost opportunities, ineffective relief operations, and loss of life and livelihood [17]. Thus the issue of collaboration and timely data dissemination turns into a complex procedure of “who has what”, “where is the information” and “who can we share it with”. This issue arises due to the fact that no single operational actor enjoys the full authoritative role; meaning

that there is usually no lead actor who has the authority as well as the capability and resources to monitor and coordinate the activities of the involved and/or affected parties. Thus, a top-down approach of appointing a lead actor is ineffective in these situations due to the plurality of actors involved/affected and differences in each actor's operating procedures as well as trust relationships with other INGOs, NGOs and governmental bodies [18]. On the contrary, a decentralised network might provide more glue to the myriad of actors as suggested by available evidence from academic research, for example, [17] [19]. Although the aforementioned research suggests a loosely-coupled social network, the same concepts can be applied to an online electronic network, a "network of networks" which would allow potential actors access to data of interest so long as they are authorised to consume the given data.

From the analysis above, the characteristics of a DM application can be summarised as follows:

- Non-linear demand in a largely unpredictable environment: In DM, sudden bursts in data processing requirements can arise due to a surge in acquired sensing data, or if a certain deadline needs to be met, or if another disaster occurs at the scene. This needs to account for peaks & troughs in demand can be attributed to most WoT application.
- Array of actors: DM and relief are usually carried out by a variety of governmental agencies (e.g. coast guard, ambulance and police services, fire services, etc.), NGOs and INGOs. This array of actors, by the very nature of our species, has inherent trust issues. Furthermore, varying mission goals, working practices and cultural differences can paint further

chaos and confusion in the scene and lead to a point where timely and effective collaboration becomes nearly impossible.

2.2 Data Classification & Representation

2.2.1 Data classification

In this study, data is represented into three forms [20] to identify it in terms of its granularity and usefulness in contexts of collaboration:

1. Data – This is raw data that is only understood by the internal network, and is useless in contexts of collaboration because external agents do not understand what it represents. An example of this type of data is a sensor reading, “30”. The internal network can understand this data because it conforms to some proprietary schema which is known and understood by the network. However this data may not be understood by external networks and agents if they don’t understand the proprietary schema used to represent the data.
2. Information – When a semantic concept is applied to data, it turns into useful and machine-process-able information. An example of this is when the semantic context of “temperature” is applied to a raw sensor reading of “30”, producing the information “temperature is 30”. Information can be understood by other networks as long as they understand the semantic concepts used to annotate the data.
3. Knowledge – By composing pieces of information and applying intelligent reasoning on it, high-level and rich business knowledge can be derived. An example of this is: “very cold in flooded areas of Sunbury”, derived from the information “-2 degrees Celsius” , “location is Sunbury” and “water level: overflowing”. For the most

part, knowledge can be considered a composition of various pieces of information that are then combined using some pre-defined rules and abstractions. As humans, this is what we are really interested in at the end of the day.

Figure 2-5 shows an illustration of this data classification. In the first instance, the raw data is present. In order for this data to represent meaningful information, it needs to be represented in a semantic context which adds meaning to the data. Further processing and composition can then turn many pieces of information into high-level knowledge which can be used in business-centric applications such as production of reports and delivering of informative status alerts.

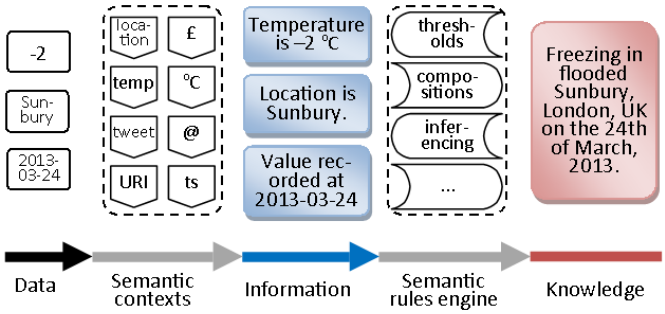


Figure 2-5: Illustration of data classification

Data may be stored in a semantic or non-semantic fashion. Traditional approaches use non-semantic storage and represent data in proprietary schemas. These types of schemas are a hindrance to cross-vendor collaboration because they do not use a standard language to represent the data and thus cannot be processed automatically by machines. However, even if the data is stored in a non-semantic fashion, it can still be represented in a semantic fashion to facilitate machine processing collaboration at a later time.

2.2.2 Sources and methods of data acquisition

It has been mentioned previously that sensor data is primarily captured from deployed sensor networks, but it can also be fetched from repositories that expose their data through an API (e.g. Xively). Most of the time, the fetched data will be non-semantic [21]. This will require an understanding of the source schema to decode the definition of the sensing devices and their associated data. However, semantic data (also known as “Structured Data” and “Linked Data”) is starting to appear on the wider web, and the most prominent repository in this regard is DBpedia [22]. There has also been a significant rise in governments exposing crucial environmental data in a semantic or semi-semantic fashion. The most recent example of this is a case where the UK Environmental Agency released structured flood data for the #*Floodhack* event organised in London on the 14th of February 2014 [23]. A great wealth of the data offered is often packaged in the form of an archive, and is not therefore a live representation of events occurring on the ground in real-time, but rather a historical account of events that have transpired.

Acquisition of data depends on the way data is published in a repository. For data that is published in a semantic fashion, semantic query languages (discussed further on below) can be used to fetch and interact with the data. This provides a universal approach to interacting with semantic data without the need to build special APIs and adapters. Non-semantic data, on the other hand, requires development of special adapters that can interface with the target repository. The lack of semantic interoperability in this case means

that non-semantic data is harder to expose and make use of, and therefore a hindrance to cross-vendor collaboration.

2.2.3 Representation of data and devices through an asset model

The asset model describes the relationship between the different levels of granularity of sensing devices. It also defines structures for storing and representing the sensor data. Essentially, it forms the foundations for the capture, storage and representation of sensing devices and their data, at a basic level and in a non-semantic way. The asset model concept is something that is largely introduced by this study, although fragments of it exist in current literature.

The asset model provides a means of modelling and representing data at a primitive level. At this stage, there are no formal semantics involved in the definition of the sensing devices and data. The provision of a non-semantic asset model may seem a frivolous task at first since the focus is on representing the data in a semantic fashion. However, this study deems it essential to provide an asset model in a collaboration framework for two reasons:

1. To provide backwards compatibility for non-semantic systems: By virtue of an asset model and an API, systems that are not semantically aligned can still access raw and unprocessed sensing device data from the collaboration framework.
2. To allow inspection of source data [24] which constitute a higher level semantic knowledge: It might be required to drill down to the different sources of data that are resulting in the composition of a piece of knowledge for debugging or performance analysis purposes.

2.2.4 Limitations of a proprietary schema-driven asset model

In order to make use of data represented in a proprietary schema-driven asset model, participating agents need to have an understanding of the proprietary schema so that the data can be captured and transformed into a suitable format. There is scope for further complication in this process since the proprietary schema can incur changes in meaning and/or structure, forcing participating agents to update their understanding of the schema to the new format. In Figure 2-6 it can be seen that each proprietary schema requires a separate mapping before the data represented through it can be understood by the participating networks and services.

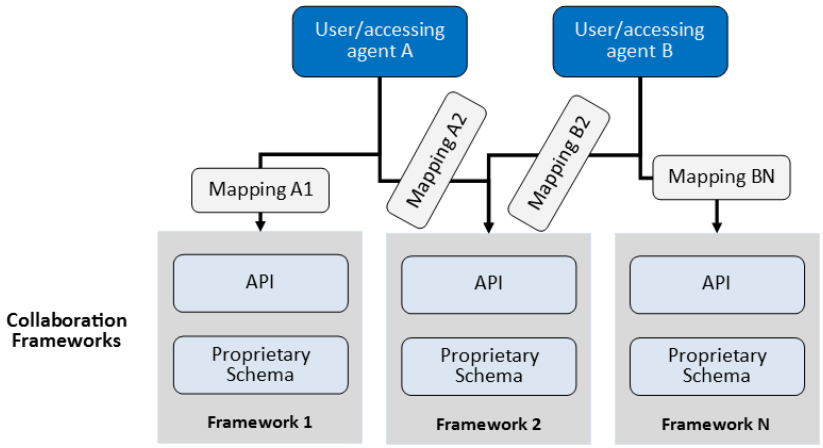


Figure 2-6: Illustration showing separate mappings needed to work with each proprietary schema

Solving this problem requires the development of a KM system that can transform data into knowledge and effectively manage this derived knowledge to facilitate collaboration [3].

2.3 Knowledge Management

2.3.1 Overview of knowledge management

KM is the functionality to capture, codify, store, process and share raw data, information and knowledge [3]. Essentially it is a study related to the generation of high-level knowledge from processed data and information. In

regards to DM, current literature uses a variety of terms to define frameworks utilised for the task of KM, which is to cater for standardised information retrieval/derivation and information/knowledge sharing/exchange:

- Information/KM Systems (IMS/KMS) [3]: KM is defined in [3] as an activity by which an organisation captures, processes and applies knowledge effectively. Such systems can be represented as "document management systems, semantic networks, object oriented and relational databases, decision support systems (DSS), expert systems and simulation tools". Examples of literature that explicitly identify themselves as a KMS-based solution include [25], [26] and [27]. Further analysis on these studies is presented in the literature review section.
- Emergency Information Systems (EIS) [28]: Defined as a system that is used by organisations to react and respond to situations of crisis and disaster, these systems are designed to: (1) Support communication during crisis response; (2) Enable data gathering and analysis; and (3) Support the decision-making process. Examples of EIS include IMASH [29], PeopleFinder [30] and Google's Person Finder application [31].
- Terms such as Crisis Response (CR), Crisis Response Information Systems (CRIS), Emergency Response Systems (ERS) and Web-based Emergency Management Information System (WEMIS) are also used in some literature.

Regardless of the actual terminology used by existing literature, the intent of these systems is to manage information/knowledge, be it related to the

actual collection, retrieval, processing and analysis of data or the consequent knowledge-derivation and sharing of that data to enable higher-level functions and business processes.

2.3.2 Schemas & mechanisms for annotating devices and data

A schema is a means of representing the definition of sensing devices and their corresponding properties, attributes and data. To enable cross-vendor collaboration, the deployed schemas need to be interoperable so that each participating network can understand and relate to data and information being exposed in the other systems. The main challenges to overcome here are: (1) Heterogeneity in the data modelling architecture and hierarchy, and (2) Heterogeneity in terminology used to store data [32]. Heterogeneity in the modelling of data refers to issues such as differences in groupings or granularity of elements. For example, in one schema, the sensor attribute “range” might be stored in the group “root -> sensor_properties”. In another schema, the same attribute may be stored in the group “root -> device -> properties”. Heterogeneity in terminology used refers to the issue of synonyms, antonyms, and the like. For example, the “battery level” attribute might be stored in different schemas in elements with varying terminology, like “battery_level”, “fuel_capacity”, etc. The issue of interoperability in the context of a schema for the WoT can be classified in two categories:

1. Syntactical interoperability.
2. Semantic interoperability.

Syntactic-level interoperability is necessary to model and represent data in a standardised way across multiple systems. It can be achieved through the use of standardised encodings and by using an interoperable mark-up format

such as XML. This facilitates interoperability in terms of terminology and mark-up. However, the actual interpreted meaning still remains an issue and can change from system to system. Maintaining a consistent meaning of definitions and data across multiple systems and platforms requires semantic-level interoperability. Semantic-level interoperability is achieved through utilisation of semantic technologies and ontologies (explained below in section 2.3.3).

2.3.2.1 Schemas and mechanisms that achieve syntactical interoperability

Descriptions of sensing devices and data need to be encoded in a certain fashion before they can be represented in a presentable fashion. Open Geospatial Consortium's (OGC) Sensor Web Enablement (SWE) [33] suite of standards is perhaps the most commonly used set of schemas to achieve a unified and standardised encoding and representation of devices and data. The suite consists of various standards, but the following are the most prominent within the suite:

- Observations & Measurements (O&M), which provides annotation mechanisms and encodings in XML for recording sensor observations and measurements.
- Sensor Model Language (SensorML), which enables modelling of sensor devices and their processing systems. It outlines mechanisms for discovering sensors, locating observations (with the capability to process low-level observations), and listing task-able properties.
- Sensor Observation Service (SOS), which carries provisions for web services to interact with sensing devices.

These standards provide XML encodings and mechanisms to model devices, their observation principles and their measurement processes, and provide a standardised representation of sensing devices and their data [34]. The schemas themselves are very comprehensive, albeit complex, and prove successful in achieving syntactic-level interoperability through deployment. However, semantic-level interoperability still remains an issue, and this is the focus of this study.

2.3.3 Overview of semantic web technologies and languages

Recent systems are increasingly relying on semantics to achieve a unified representation of data and enable collaboration. The underlying semantic technology and language is called Resource Description Framework (RDF). Other technologies then build-upon RDF to achieve a certain goal. Examples of such technologies and languages are Ontology Web Language (OWL), which is used to create ontologies, and SPARQL, the query language for RDF.

RDF is used to write semantic statements as a set of “triples”. Triples consist of a subject, an object, and a predicate relating the subject to the object [35]. An example of this is: “Sensor1 measures Temperature”, where “Sensor1” is the subject, “Temperature” is the object, and “measures” is the predicate linking Sensor1 to Temperature (Figure 2-7).

Figure 2-7: Illustration of RDF SPO (Subject-Property-Object) structure

Subjects can have many predicates, thereby linking them to many objects, which in turn can have predicates linking them to other objects. This concept is illustrated in Figure 2-8 where the “Sensor1” subject has an additional

property linking it to another object. Also, the object “Temperature” now has a property of its own linking it to another object. RDF datasets such as this are often called Graphs.

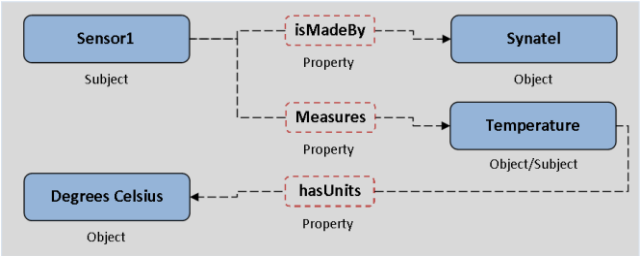


Figure 2-8: Illustration of an extended RDF SPO (Subject-Property-Object) structure, showing how objects can become subjects and vice versa

If published properly, this can contribute to the Linked Open Data (LOD) Cloud, which can be navigated and browsed like webpages [36]. One main benefit of this is that while previously proprietary data was stored and represented in rigid and vendor-specific schemas, using this language it can now be presented on the web in a standardised manner. Since this linked data is written in RDF, it can be processed by machine agents. This allows for its automatic consumption by machines, and enables autonomous Machine-to-Machine (M2M) communication and interaction.

Once the RDF annotations have been stored in some form of database backend (usually referred to as a “triple-store”), it is equally important to be able to query the triple-store in a semantic fashion. SPARQL meets this requirement by providing an SQL (Structured Query Language)-like syntax which can be used to compose semantic queries and traverse RDF triple-stores in a formal and publicly standardized format [37]. Again, by standardising the query language for RDF, it becomes possible for machines to fetch and publish semantic fashion in a (semi-)autonomous manner.

A SPARQL endpoint accepts semantic queries and returns results via HTTP. The endpoints can be:

- Generic: These will return results from any published RDF data on the web.
- Specific: These will only return results from particular online/offline RDF datasets.

The actual SPARQL query consists of the following [38]:

- Prefix declarations: These are useful to abbreviate URIs. For example, PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>. Here, “rdf:” is the abbreviated form of the full URI “http://www.w3.org/1999/02/22-rdf-syntax-ns#”. Since URIs are used in the actual query, using the abbreviated prefix is a lot easier than using the full URI.
- Graph/dataset definitions: A SPARQL query is run against a set of RDF datasets.
- Result clause: This is used to define what action to perform in the query and/or what results to return.
- Query Pattern: This is used to define the query by restricting, refining and filtering triples from the desired graph.
- Optional Query Modifiers: These perform actions on the returned results such as ordering and rearrangement.

Some of these concepts are illustrated in Figure 2-9. This figure shows an excerpt of triples from a knowledgebase. The knowledgebase is written using an ontology which is abbreviated as “saw-ont”. The SPARQL query also uses the prefix “saw-feed”, which is another abbreviation for a URI that is

used in the knowledgebase. For the sake of clarity, the prefix declarations have been omitted from the illustration. In the knowledgebase data, there is a “SensingDeviceConcept”, which has a number of derived sub-concepts: “LightSensor”, “CO2Sensor”, and so on. The “CO2Sensor” concept then has 3 named instances. The named instances are: “CO2Sensor1”, “CO2Sensor2” and “CO2Sensor3”. The query uses the “ASK” keyword which returns a Boolean (true or false). Variables are prefixed with a question mark. The query searches for a *device* belonging to a *feedConcept* which has the *rdf:type* of “SensingDeviceConcept”. The “FILTER” keyword is then used to check for a specific device instance, which in this case is “CO2Sensor2”. If this instance exists in the knowledgebase, the query will return TRUE, otherwise it will return FALSE. Different keywords can be used in the result clause to return other information or to perform update/delete tasks with the knowledgebase.

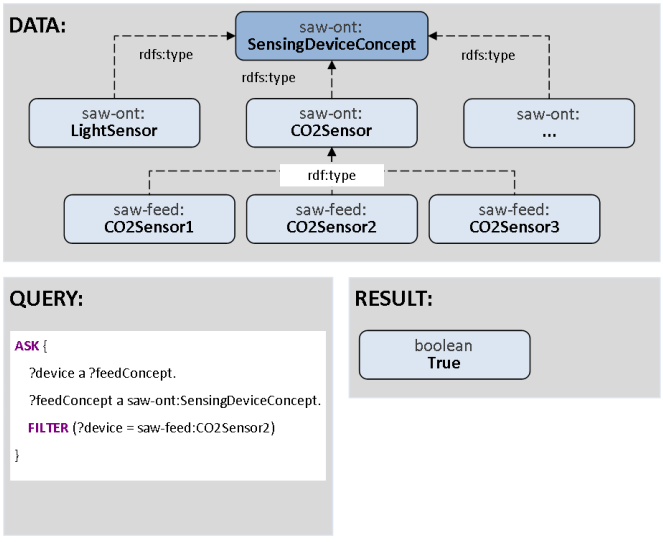


Figure 2-9: Illustration of a SPARQL query checking for existence of an instance of a sensing device

After using a machine-readable language to represent data in a standardised manner, the problem is now one of choosing the correct definitions to define properties which link subjects to objects. OWL is a semantic language which

is used to define these vocabularies and terminologies. The main artefacts in an OWL ontology are the following:

- Classes: A class is a general construct which can have members. Classes can have properties which are inherited by all individuals belonging to the class. An example of a class is “SensorDevice”. Properties attached to this class can be: “measures” (what parameter does the device measure), “manufacturer”, “isWireless”, and so on.
- Individuals: Members (or instances) of a class are called individuals. Individuals inherit properties of their parent class. Specific properties can be assigned to just the individuals as well. An example of an individual is “PositionSensor”, which belongs to the “SensorDevice” class defined above. The “PositionSensor” individual will inherit the previously defined properties for the “SensorDevice” concept (e.g. measures, manufacturer, isWireless, etc.).
- Properties: Properties have a domain (who the property applies to) and a range (what values the property can accept). Both the domain and range can be individuals which are defined in the ontology with their own properties. A property called “hasBrother” might have the domain “Person” (which resources can be mapped) and a range of “Male” (who can the resources be mapped to). Properties can be sub-properties of existing properties for better specialisation. There are two types of properties:
    - Datatype properties: These relate individuals to data types. Example of a datatype property is “sensorValue”, which might

have a domain of “PositionSensor”, and a range of another resource called “xsd:double”.

- Object properties: These relate individuals to other individuals. An example of this has been mentioned above already (“hasBrother”).

OWL makes it possible to *restrict* classes/concepts so that they have a clear and understandable meaning [39]. For example, OWL allows concepts and properties to have a data range restriction so that they can only accept certain values (e.g. the property “hasSister” having the range of “Female”), or restrict properties to only apply to specific concepts (e.g. the property “sensorValue” having the domain “SensorDevice”), or to define certain concepts as aliases of one another (e.g. “Person” defined as an alias of “Human” by using the owl:sameAs construct). With OWL, a vocabulary of well-defined and machine-readable terms can be generated, and a common understanding of these concepts can be presented to external collaborative parties. These vocabularies are referred to as ontologies, and an example of this is the Semantic Sensor Network (SSN) Ontology [40], which enables annotation of sensing devices and platforms.

The relationship between RDF, RDFS and OWL is illustrated in Figure 2-10. At the very basic end, RDF is used to write triples which relate subjects to objects via predicates. RDFS then allows for the construction of slightly more complex relationships between the subjects and objects, and most notably, it allows classes to be sub-classes of other classes. OWL then builds on top of RDFS and offers the capability to define richer and more complex relationships between classes, individuals and properties.

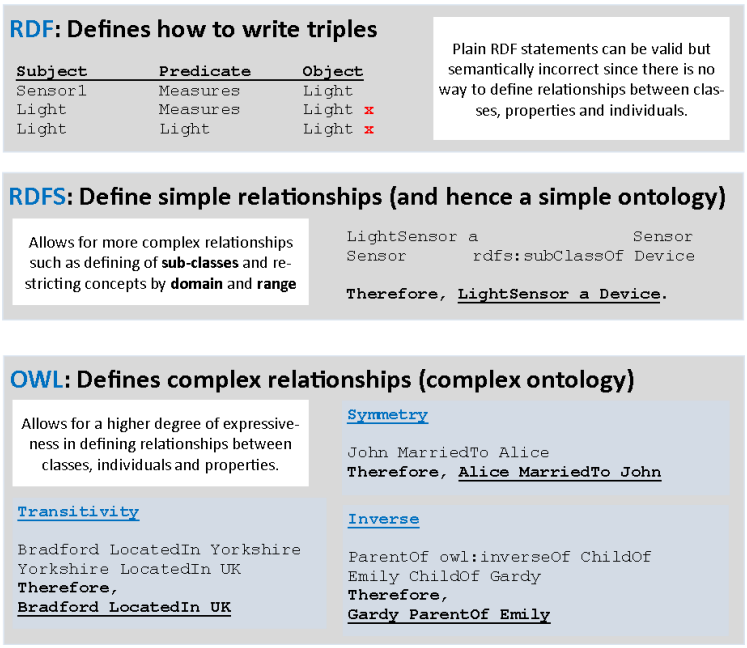


Figure 2-10: Relationship between RDF, RDFS and OWL

Ontologies, like the SSN ontology mentioned above, are typically OWL documents containing definitions of domain-specific classes and properties. The SSN ontology is related to sensing devices and sensor network deployments, so it defines classes and properties that allow sensors and their deployment environments and parameters to be described in a semantic fashion. Another popular ontology is the FOAF (Friend of a Friend) ontology which describes people and the relationships between them [41]. The illustration in Figure 2-11 shows an excerpt from the ontology (top) and a sample usage (bottom). In the ontology excerpt, the “Person” class is defined. The “Person” class is defined as a sub-class of more generalised classes/concepts from other ontologies. This increases the semantic scope of the defined concept, and enables inter-linking between existing concepts. The definition also states that the “Person” class is “disjointWith” the “Organization” and “Project” classes. This means that any member of “Person” can never be a member of the other two. For example, Bob is an

instance of “Person”. The ontology states that Bob can’t now also be a member of “Organization” or “Project”. This is useful for semantic clarity of the concepts. The second excerpt shows the definition of the “knows” property. Both the domain and range of this property are stated as “foaf:Person”. The domain implies that this property can only be applied to a member of “foaf:Person”. The range implies that the object being linked to through this property can only be a member of “foaf:Person”.

FOAF Ontology:

```
<rdfs:Class rdfs:label="Person" rdfs:comment="A person.">
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
 <rdfs:subClassOf>
  <owl:Class rdf:about="http://xmlns.com/foaf/0.1/Agent"/>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Class rdf:about="http://www.w3.org/2000/10/swap/pim/contact#Person"/>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
  <owl:Class rdf:about="http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing"/>
 </rdfs:subClassOf>
 <owl:disjointWith rdf:resource="http://xmlns.com/foaf/0.1/Organization"/>
 <owl:disjointWith rdf:resource="http://xmlns.com/foaf/0.1/Project"/>
</rdfs:Class>

<rdf:Property rdfs:label="knows" rdfs:comment="A person known by this person.">
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
 <rdfs:domain rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
 <rdfs:range rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
</rdf:Property>
```

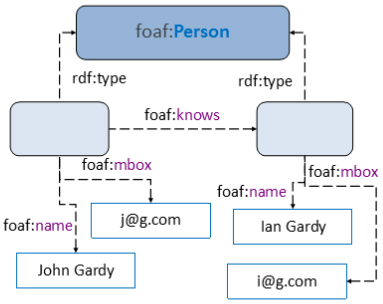


Figure 2-11: FOAF ontology illustration, showing an ontology excerpt (left) and sample usage (right)

2.3.4 Schemas that achieve semantic-level interoperability

W3C’s Semantic Sensor Network Ontology (SSN) tackles the issue of semantic interoperability by providing an ontology for describing sensors and methods [40]. The ontology itself is aligned with the Dolce Ultralite (DUL) upper ontology [42]. An upper ontology defines very general concepts that are similar across multiple application domains, and can be considered part and parcel of enabling semantic interoperability across multiple application domains. Aligning to upper ontologies helps to further formalise the semantic concepts and provision for extensions of definitions and inclusion of other ontologies [43]. SSN is capable of describing sensing devices in terms of their capabilities, observation principles, measurement processes and types

of deployments. By providing an ontology for defining sensing devices and data and by outlining mechanisms to annotate systems, devices, observable features and data, SSN achieves success in offering semantic-level interoperability [40].

SSN can either be used in a standalone fashion or in tandem with the SWE suite to annotate sensing devices and data. Unlike SWE, SSN is not XML-dependent. Furthermore, semantic annotations using SSN form the basis of Semantic Sensor Web (SSW) [34] and Linked Data principles, and present a unified representation of sensing devices, their processes, and data. In turn, the semantic technologies enable interlinked sensor data which can be published on the web.

While SSN permits semantic annotation of sensing devices and data, thus enabling M2M interaction and semantic information consumption, the mechanisms required to add and process the annotations still need to be developed separately. The annotation can be manual or (semi-)automatic. Manual annotation will require a greater level and frequency of intervention by system operators and administrators as they will have to add new concepts and correct existing mappings. This level of intervention can be reduced (therefore improving the system scalability) by introducing automatic or semi-automatic annotation mechanisms and processes. This will require a system where the users of the system can contribute to the network in a seamless fashion to expand the knowledgebase and improve the accuracy of its mappings. The system will in turn need to provide mechanisms that can recognise and correctly identify mappings against existing concepts stored in the knowledgebase.

2.3.5 Analysis of existing knowledge management systems

From the commercially or publicly available solutions, Xively stands out both for its list of features and for its public adoption. It allows granular modelling and representation of sensor devices and data in terms of DF, DS and DP. However, it uses a proprietary and rigid schema with which to define sensors and actuators, and offers no relief in terms of semantic metadata to increase the openness of the system.

Thingspeak is further restricted as it lacks the ability to flexibly model DS. At most, a user can have 8 DS (called “fields”) in a DF (called “channels”), and there is no support for managing access to the individual DS within the DF. As with Xively, there is no support for adding semantic metadata to devices and data in Thingspeak.

In literature, Murphy and Jennex highlight the importance of KM and the growing necessity of its effective application in Crisis Response [30] by analysing two *leaderless* systems developed in the wake of Hurricane Katrina which hit the US Gulf Coast in 2005, namely PeopleFinder and ShelterFinder. The study states that “*everyday citizens that would like to contribute are unable to, not only because they are not inside the physical operations*” but also because responders themselves are not able to reach out to the community to mine crucial data. Furthermore, the study brings to light the growing importance of the need for “*systems that can quickly find and display knowledge relevant to the situation in a format that facilitates the decision maker in the decision process*”.

The case study presented by Bharosa & Janssen in [44] is very comprehensive in this regard as it focuses on KM and deals with the issue of

information management adaptability, that is, the system's "*ability to rapidly change existing or create new resources in order to align the internal information demand with external information supply and events*". By using a resource-oriented approach, they investigate the problems related to the adaptability of a CRIS (Crisis Response Information System) in regards to internal data management needs as the external conditions vary in response to on-going/evolving disaster scenarios. The team made several notable observations and concluded that overall in the mock drill, the information quality was poor in regards to its relevancy (who is the information for), consistency (various interpretations), accessibility (inability to access contextual information), reliability, correctness and completeness (at the time of viewing) even with the use of CEDRIC, an advanced web-based application for collaboration. Furthermore, they alluded to the fact that existing EIS systems are very close-knit solutions and are not flexible enough to permit integration with external resources beyond the scope of the immediate framework. The study does not propose a definite system design but rather presents a set of principles which, in the minds of the authors, should be implemented by an adaptive and responsive EIS. These principles are:

1. Maintain and update team memory via a directly exploitable library/information storage system that is capable of storing information flow thus removing the need for repetitive requests and reducing the chance of presenting incomplete or outdated contextual information.

2. Dedicate specific resources for environmental scanning to ensure up-to-date situational awareness amongst all involved actors.
3. Maximise the number of alternative information sources to augment existing information base and derive more accurate knowledge about the situation. Alternative sources of information also help to remove single source dependency which can impact the decision-making process and reduce quality of available information.

SENSEI [45], a European FP7 project, provides a more comprehensive KM system. It recognises four types of base-level resources: sensors, actuators, processing components and composites of each of the preceding resources. The physical resources are represented as web resources through a Resource End Point (REP). The system model also models real-world entities centred on device interactions, for example, people, places and objects. A Resource Directory (RD) is used to store device descriptions and to provide a list of resources for which an external request meets the criteria. The RD is complemented by the Entity Directory (ED) which stores relationships between devices and the modelled entities of interest, and is used to query interaction capabilities between the two. Semantic query support is enabled over the directories so that rich and natural-language queries can be conducted on top of the stored resources and entities. However, this requires semantic annotation of the stored resources and entities, a process which is not carried out by default and is a manual, optional step in SENSEI's resource model. In all, SENSEI makes great leaps in terms of achieving device and syntactic-level interoperability, and the option to add semantic annotations can even achieve semantic-level

interoperability. However, the usage of a strict, rigid and complicated O&M schema prohibits generality and extensibility of device and data templates and it is hard to envision the technology being used by regular users outside the corporate realm.

On a slight tangent, an experiment-based study in [46] brings to light the issue of *trust* in the use of computing technologies that are made *invisible* (to hide the underlying complexities). The study evaluates that in order to build *trustable* pervasive computing systems, i.e. systems that the end-users will actually use as opposed to doubt and refute their credibility, reliability and accuracy, the system design needs to embody the following key principles:

- Flexible interaction modes to enable multiple forms of inspection and retrospection.
- Multiple levels/layers of inspection to enable system-wide examination of the *states*, *processes* and *connections* of the system.

The authors argue that while the underlying complexity can be hidden to provide more intuitive UI (user interfaces) to the end-users, there needs to be the capability to inspect the traffic flow within the framework, ideally, down to the very primitives (e.g. GPS coordinate string in a data packet). It is SAW's belief that this level of transparency can be best achieved by utilising a resource-oriented systems design approach.

A very recent experiment-based study by Caragea et al. in [47] highlights the growing importance of social media integration and alignment in DM applications and the inherent challenges imposed for machines in learning to analyse and *classify* information posted by affected people and the general public, especially short messages such as those produced via SMS and

Twitter. The findings in this study are further augmented by [48] which highlights the power and usefulness of social media as an information dissemination tool but warns that it also has the potential of being a distractive and disruptive medium if the capability to methodologically analyse, process and act upon the data is lacking or absent. [48] argues that a *representation strategy* (i.e. an ontology) needs to be formulated and/or identified for "*formulating methods for communicating and capturing crisis response data, information and knowledge from social media*". This analysis strongly suggests that a next-generation semantics-driven collaboration framework for the WoT needs to be mindful of and strive to provision the capability to integrate with social media and take steps to ensure that information retrieved through these non-official networks is used, albeit cautiously, to build a more complete model, in real-time, regarding the situation on the ground.

The virtual multi-user collaboration simulation mode envisaged in [49] also deserves a mention as it targets the issue of multi-user and cross-organisation collaboration and sets the foundations for further research into the respective areas regarding these issues. Similarly, [50] demonstrates the foundations of a similar system with a virtual environment powered by Half Life 2 and the collaboration needs met through a combination of a CMS (content management system) and a Wiki. But as is apparent, these systems have limited outreach and the fundamental design principles do not permit flexible and comprehensive cross-vendor integration let alone collaboration.

[51] analyses the performance of an Integrated Operational System (OS) built on the premises of integrating multiple sources of information arriving in

multiple formats into a single common platform. The analysed system is, however, a closed-ecosystem whereby open and 3rd party integration is not possible and only a limited number of pre-configured platforms can be utilised by the system, and even then, relatively marginally. Therefore the problem of effective and open cross-vendor collaboration remains a key research issue.

The KM solutions reviewed so far either make no mention of the underlying asset model, or treat it as an insignificant subsystem, focusing instead on high-level tasks and services. The study presenting a resource-oriented WoT architecture in [52] is different as it focuses on this very same thing: how should resources best be modelled and presented on the web. It adopts a RESTful approach and promotes a hierarchical resource architecture such that each device (DF) and its children devices (DS) can not only be browsed individually, but also contain a link back to the parent (DF or DS). For example, the URL *http://.../devices/device1/sensors/sensor1* navigates to *sensor1* which is part of the *sensors* property of *devices*. This creates an unlimited-depth and hierarchical structure that can be easily crawled and navigated. The authors recommend the use of JSON to model smart object properties and show a proprietary schema used to define properties for smart things. It is not clear whether the definition templates are extensible from the presentation, but the authors do make an effort to semantically-align their proposed solution by describing devices and data in Microformats [53], achieving some form of semantic interoperability. The annotation process, however, is manual and there is no learning system that can adapt from

previous annotations and offer some sort of semi-autonomous semantic annotation facility.

Similarly, the IoT-A reference architecture [54] also looks very promising. The rich, comprehensive and complex architecture and associated information models are still in development, and the final architecture is, as of yet, unavailable. Sensors, actuators and smart tags are abstracted as *devices*. Then physical and virtual resources are combined to form *aggregated entities*. The definitions are stored according to a rich but proprietary schema and prospects of semantic annotation are considered, but not detailed in this thesis.

DERMIS, [24], undoubtedly provides the most comprehensive set of principles and guidelines for developing an emergency response system, but focuses on developing a "*single integrated enterprise type system*" that "*spans all the functions of the emergency response from planning, through execution and recovery, to training*". However, this goes against the very premise and design principles of modern distributed systems, as is evident primarily in social media and secondly in the rapid influx and growing number of web services exposing public APIs whether it be for authentication, data access, information sharing or embedding non-native and 3rd party functionality. So while respecting the design principles and fundamental guidelines offered by this study, the single-systems approach adopted by the authors is refuted as it's incompatible with the modern decentralised and distributed nature of web services. Furthermore, whilst the availability of a single authoritative command and control centre might improve data accuracy and consistency, it hinders third party integration and thereby limits

not only the ability of regular users in accessing and making use of the system, but it also dampens the prospects of mining data from disparate sources that would effectively be shut off to the single monolithic repository represented by DERMIS.

2.3.6 Derivation of functional requirements for knowledge management

The KM system in question will be dealing with a general WoT domain. The characteristics of the DM application domain in WoT have been presented earlier. In relation to the issues highlighted, it can be said that a framework which sets out to enable data mining, processing and collaboration facilities not only in the DM scene specifically, but in the whole WoT domain in general, needs to be designed with three characteristics in mind:

1. Flexibility**:** The framework should be easy to setup, maintain and dismantle. The interaction mechanisms should enable different instances of the framework to collaborate with ease, whilst also making possible collaboration with other IoT repositories and/or other 3rd party services.
2. Generality**:** The framework should not be designed as a particular application but instead as a generic platform for storing sensing devices definitions and data. This will enable the framework to be utilised for different purposes, thereby increasing its value and configurability for a diverse range of WoT-related problems which share common traits.
3. Extensibility**:** The framework should be designed in an extensible and service-oriented fashion so that extensions can be developed and deployed with ease. This will allow the development of extra components for the system which can be leveraged by users of the system to increase

functionality in a certain realm or to augment the system with additional features.

Furthermore, the authors of [24] outline a set of conceptual design requirements from which the following are taken as being appropriate for a generic framework:

- Information source and timeliness: All assets captured by the network should be identified by their source and time of capture. Any linked or child assets should also clearly state the time of capture and the source of the information so that decisions can be made regarding the most up-to-date piece of information.
- Comprehensive system & event log: All system actions should be logged at different levels of granularity and categorised to produce a collective memory bank. This bank can be inspected at any given instance in time to build an overall picture of the state of the system regarding manipulation of all system assets (capture, publication, interactions, processing, requests, responses, etc.). A log is the "*ongoing roadmap of the emergency*" situation and thus needs to be comprehensive enough to allow for different levels of abstraction and detail so that the correct level of detail can be presented to the responders at any given time.

To develop an efficient next-generation KM system, there needs to be a balance between enforcing semantic schemas to provision semantic-enrichment of data and the capability for users and agents to use the system without being forced to comply with strict and rigid schemas. Furthermore, from the aforementioned analysis and issues raised from existing studies, it

is necessary that a suitable asset model is developed which allows high level of introspection of the underlying assets so that underlying DS can be analysed when needed [24]. Taking this into consideration, key fundamental principles for a semantically-aligned KM system can be defined as follows:

1. Hierarchical data/information model that enables high level of introspection;
2. Generic and extensible data definition templates that provision for future extensions and enable users to upload customised data;
3. Semi-autonomous semantic annotation capability that is optional and not enforced.

A hierarchical data/information model is necessary to allow introspection into the system, and to be able to compose and decompose raw data into information into knowledge. Real-world sensing and actuating devices can be generally split into two types:

i. Sensing/actuating devices, for example, a temperature sensor connected to an Arduino board;
ii. Multi-device platforms, for example, an Arduino board and a SunSPOT system; such a device may consist of several sensing devices.

In this thesis sensing/actuating devices would be modelled as datastreams (DS or streams) and multi-device platforms as datafeeds (DF or feeds). This creates a simple hierarchical model where DF contain one or more DS. The DS upload data, for example: sensor readings, to the network periodically or when a sensing event occurs. These readings will be referred to as

datapoints (DP or points). The illustration in Figure 2-12 shows the relationship between DF, DS and DP.

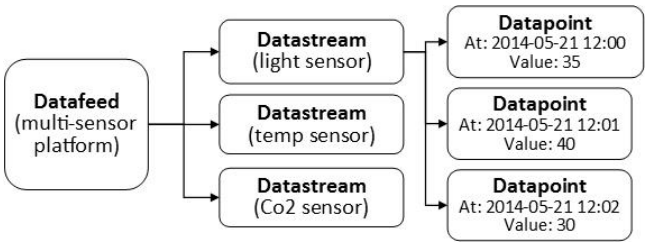


Figure 2-12: DF, DS and DP relationship diagram

A generic and extensible data definition template is essential for providing a low-entry barrier for users of the system and enabling extensions. Before a device can be used with a network, it will need to be registered. When this happens, the user will need to upload a definition by filling in some template. By providing a generic and extensible data definition template, it becomes easier and more convenient for the user to supply mandatory information and still retain the flexibility to add arbitrary data that might not be understood by the network, but can still be used by the user.

Semi-autonomous semantic annotation capability is an important aspect of a heavy-load WoT application where manuals annotations are unfeasible due to the large number of sensing devices being added to the network constantly. It has been mentioned before that a community-driven and user-oriented system is needed that can learn from the annotations submitted by the system users. Thereafter, the system can offer automated annotation facilities for new sensing devices by attempting to map them with concepts already stored in the knowledgebase. A community-driven and self-learning system such as this is essential for scaling a semantic WoT application. The actual semantic annotation process is selected to be optional. This will hopefully lead to greater adoption of the system since no one is forced to

comply with the semantic annotation process, but anyone can do so to increase the capability and usefulness of the overall platform.

To compare existing works in light of the performed analysis and deduced principles, a formal list of functional requirements for KM is presented below. Please note that FR1 is the group of functional requirements for evaluating KM systems. FR2 is the group of functional requirements for evaluating access control systems. FR2 is presented in the section pertaining to access control mechanisms further below.

FR1: Comprehensive, extensible and semantic asset model: Devices, their definitions and the data they upload to the network is referred to as “assets”. The framework should be able to represent assets as resources and users of the framework should be able to break down any composite (e.g. overlaid graphs) into its basic underlying assets (e.g. list of DS). Furthermore, the data model should be semantics-driven to enable semi-autonamicity in the system (i.e. the ability of the machine to “learn” the “meaning” of data over time).

1. General Capabilities – Asset Model:
    a. Capability to model and represent multi-sensor devices and composite platforms (e.g. an Arduino board, a SunSPOT device, or another multi-device platform) as DF, each consisting of many DS;
    b. Capability to model and represent a single sensing/actuating device that may or may not be part of a bigger platform as a DS;

c. Capability to model and represent individual sensor readings that occur at some moment in time as DP;
d. Capability to define arbitrary and additional properties when modelling devices (DF, DS and DP), easily and conveniently, through a generic, schema-less and extensible template;

2. Semantic Capabilities: Capability to represent resources in a semantic fashion to represent data in a standardised manner and to enable automatic or semi-automatic machine processing.
    a. Capability to annotate DF and DS with semantic concepts/metadata;
    b. Capability to run semantic queries against semantic metadata.

2.3.7 Comparison of existing knowledge management solutions in relation to the asset model

An analysis of KM systems was presented earlier in section 2.3.5. These KM systems existed both for DM applications as well as WoT applications in general. Table 2-1 presents a summary of all of these solutions that describe some form of asset model for modelling devices and data. It compares each of these existing solutions against the list of functional requirements presented in the previous subsection.

It is important to note that in all the presented studies, very few actually tackle the issue of unified data/knowledge dissemination. Where this issue has been given some focus, the outreach has been more or less limited to *similar frameworks/systems* (i.e. cross-instance collaboration) and no particular focus has been applied on the problem of exposing this data or knowledge to 3rd parties (i.e. cross-vendor collaboration). It is believed that a generic and extensible asset model would form the basis for standardised

sensing device definition and data representation. Since this area has not been addressed adequately in existing literature, it is one of the requirements and deliverables of the successive semantic KM framework being proposed in this thesis.

Table 2-1: Comparison of existing literature against requirements to satisfy asset model

Requirements/ Solutions	**Granular device modelling hierarchy, allowing modelling of resources as a:**		**Extensible and schema-less device and data representation templates for:**			**Semantic capabilities**	
	DF	**DS**	**DF**	**DS**	**DP**	**Annotation**	**Querying**
Commercial and publicly available solutions							
Xively [55]	✓	✓	✗	✗	✗	✗	✗
ThingSpeak [56]	✓ (as channels)	Partial (as fields, <= 8 / channel)	✗	✗	✗	✗	✗
Paraimpu [57]	✗	✓ (as sensors and actuators)	✗	✗	✗	✗	✗
KM solutions and reference implementations in literature							
Dynamic-map container terminal in [58]	No mention or indication of granular access to underlying assets or how this data is defined and represented					✗	✗
PeopleFinder and ShelterFinder in [30]	Based on manual input and web-scraping of information conforming to a strict schema (people, locations) as opposed to collecting sensor data. DM application but not based on sensor networks					✗	✗
WoT architecture in [52]	✓	✓	✗	✗	✗	✓	✗
IoT-A reference architecture [54]	✓ (as devices)	✓ (as sensors, actuators and tags)	✗	✗	✗	✗	✗
SENSEI [45]	✓ (as resource hosts)	✓ (as resources)	✗ (O&M and other suites from OGC's SWE package are used to model and store devices and data)			✓	✓

SmartSantan-der [59]	✓	✓	✗ (O&M and other suites from OGC's SWE package are used to model and store devices and data)	✗	✗

2.3.8 Comparison of existing knowledge management solutions in relation to semantic capabilities

Table 2-2 presents a summary of all KM solutions that describe some form of semantic technology for annotating sensing device definitions and data. Again, each solution is compared against the list of functional requirements presented in section 2.3.6.

It is apparent from the analysed literature that there is a clear lack of autonomous semantic annotation capabilities and mechanisms. A majority of the major vendors are opting for OGC's SWE suite of standards which provide syntactic-level of interoperability. Various ontologies are then used to enhance the metadata contained within the data stores and to annotate the sensing devices and data. This, of course, achieves some form of semantic-level interoperability and enables M2M interactions over the data. However, the issue of autonomous or semi-autonomous semantic annotation remains a pressing issue. In the WoT where hundreds or even thousands of devices are expected to appear in a short intervals, manual semantic annotation of artefacts is simply unfeasible, and automated or semi-automated annotation of resources is a key priority [20].

Kno.e.sis linked sensor data platform is a major effort in collecting weather data from weather data stations, and then encoding these in O&M. However, it is not clear whether there exist any mechanisms to automatically enhance the encoded O&M data store with semantic annotations, and if they do, then whether there are provisions to annotate a broader range of devices. On that

front, SENSEI is a broader and more comprehensive platform which provides semantic annotation capabilities as an optional enhancement to its O&M encoded resource and entity directories. However, the process for doing so is manual and there is no support for providing automatic annotations for sensing devices, entities and observations.

Table 2-2: Comparison of existing semantic annotation solutions

Requirements/ Solutions	**Annotation methodology and ontologies used**	**Further comments**
Kno.e.sis linked sensor data [60]	**Encoded using:** OGC's Observation & Measurement (O&M) standard [61]. **Ontology:** Custom-made Sensor-Observation ontology (based on O&M concepts) **Procedure:** Encode raw textual data obtained from MesoWest in O&M and then turn it into RDF statements.	**Info:** Published datasets contain description of 20K weather stations in US. Approximately 5 sensors per weather station measuring temperature, visibility, precipitation, pressure, wind speed and humidity. **Drawbacks:** Limited scope, only deals with known sensor types from weather stations. Sensor definitions are very top-level and no way to define a sensor as part of a bigger platform. From what can be observed (as the actual annotation mechanisms are not detailed) annotation requires manual adjustment and there is no autonomous feature to learn from existing annotations.
Sense2Web linked sensor data platform [62]	**Encoded using:** RDF. **Ontologies:** Custom-made local ontologies. **Procedure:** Manual annotation via web interface	**Info:** Uses DBpedia for Sensor Types **Drawbacks:** Focuses only on describing sensors on a very top-level and provides no support for describing observation and measurement principles. Manual annotation and no autonomous features to speed up annotations.
SensorMasher [63]	**Encoded using:** RDF. **Ontologies:** Custom core ontology and an extended ontology inspired by the SWEET [64] and SANY [65] ontologies. **Procedure:** Manual annotation	**Drawbacks:** Shallow device hierarchy, doesn't use the more prominent SSN ontology which provides better interoperability, no automated annotation capability.
WoT architecture in [52]	**Encoded using:** Microdata. **Ontologies:** Mixture. **Procedure:** Manual.	**Drawbacks**: Limited scope, manual annotation, very top-level and simplistic.
SENSEI [45]	**Encoded using:** O&M and RDF. **Ontologies:** Unclear **Procedure:** Manual	**Drawbacks**: Manual annotation, no learning mechanism.
SPITFIRE [66]	**Encoded using:** RDF. **Ontologies:** Custom ontology based on DULE and SSN. **Procedure:** Semi-automatic	**Info:** Semi-automatic creation of semantic sensor descriptions is achieved by comparing the sensor output of newly deployed sensors

	annotation based on sensor output	against already deployed sensors.

The WoT architecture presented in [52] is overly simplistic in terms of providing semantic annotations. It uses Microdata which is not as diverse and flexible as RDF, the official annotation language used in many of the present-day frameworks and systems for annotation. On the other hand, the Sense2Web linked sensor data platform in [62] does a much better job by using a suite of custom local ontologies and interfacing with DBpedia to inference some information (e.g. pulling sensor types). However, once again, there is no learning system or autonomous annotation capability and semantic metadata has to be entered manually.

SPITFIRE in [66] achieves some form of semi-automated capability to annotate sensors semantically by comparing the output of newly deployed sensors against those already deployed over some period of time. By comparing the time series of devices, SPITFIRE can correlate new devices against those already deployed and producing a similar time series. If multiple correlations are present, then the user has to manually select the most correct one. Over time, the system is expected to increase in accuracy as more devices are added to the network and annotated successfully. What is not clear is the success rate of this algorithm for similar devices that are deployed in completely different environments, or a great distance apart; for example two motion sensors, one deployed in a busy university laboratory and another in a fire exit of a shopping centre.

In all, there is a definite lack of attention and innovation in the field of automated annotation for sensor devices and data.

2.4 Identity & Access Management

2.4.1 Overview of IAM

IAM is the functionality to manage the visibility of assets to users of the system providing they are authorized to do so. This may be done via numerous methods and approaches (and not necessarily containing functionalities for *identity* (who are you) and *authorization* (can you do a, b and c) in each case). Typical DM and collaboration frameworks have limited collaboration facilities and therefore only focus on the internal IAM functionalities, i.e. the ability of users within the organisation to access resources, carry out tasks and view information. However, in the context of WoT access to information from external assets is also vital, the concept of IAM needs to be expanded. For example, if an autonomous agent wants to scrape and process semantic metadata from an IoT repository, what kind of access policies will it require? Will registration be a prerequisite? What about if it's a temporal interaction and registration, therefore, becomes an unnecessary and even prohibitive hurdle? The problem is no longer managing access from internal users (which has received due attention); but now it's becoming more important to consider how external collaboration agents wishing to interact with the network, can do so without necessarily having to register, but at the same time, retaining control over access privileges. This study only deals with the latter problem; that of managing access to unbounded, temporal and dynamic resources, and SAW contributes a potential solution to this problem by presenting an enhanced token-based approach for managing access rights and policies.

The purpose of access control is to limit access to privately-owned resources and assets by the owner of these resources. In this regard, a few methodologies exist:

- User/Identity-Based Access Control (UBAC/IBAC)
- Authorization-Based Access Control (ABAC)
- Role-Based Access Control (RBAC)
- Token-Based Access Control (TBAC)

2.4.1.1 UBAC/IBAC

The most basic type of access control is UBAC which forms the basis for security standards like SAML [67]. In this scheme, access to resources is controlled by the identity of a user and is therefore very problematic when it comes to public sharing of resources directly. Furthermore, since policies are tied to user accounts, if a user account is revoked or deleted, the related policies also disappear and have to be generated again for any successive users who might assume the same role. If the policies are stored in an access control list (ACL), then the ACL needs to be updated as and when users accounts are revoked or deleted which can quickly become messy in a real WoT scene [68]. It is highly unfeasible to use this scheme in the WoT due to its numerous restrictions, even though it can be considered to be the most secure out of the other schemes. It is only really suitable for applications where users and their roles within the application domain remain constant over a long period of time, for example, in a business environment.

2.4.1.2 ABAC

ABAC is very similar to UBAC in that it is still based around the existence of users but instead of using the identity of a user (i.e. *who is the user*), it

focuses on making use of the actual authorization data for the identity (i.e. *what can the user do*). In doing so, it eliminates the problems related to distributed identity management since each participating service domain has information about its own services and the relevant authorizations required to carry out those services. Then each domain only needs to be presented with the correct authorization to invoke a service, as opposed to revealing the caller identity as is the case in UBAC [69]. Whilst ABAC achieves many advantages over UBAC, it is still user-centric and therefore unsuitable for a resource-centric WoT where access to resources by anonymous agents is a key requirement.

2.4.1.3 RBAC

UBAC introduces many problems in large organisations where users are prone to changes in jobs, roles and duties. Any change in the job or role of a user would result in a complete rewrite of the policies for the user to align it with the subject's new job or role. Aside from this, two users with similar jobs and roles will have two separate access policies in an UBAC system. RBAC was introduced to offset some of these managerial disadvantages of UBAC [70], and it is based on the premise of roles which belong to users and have access policies [58]. The scheme relies on the hypothesis that roles and responsibilities largely remain constant within an organization and it's the users that change, therefore modeling access policies through user roles instead of user identities provides a more convenient and maintainable solution [71]. However, RBAC is unsuitable for modelling access control where roles are hard to define and/or unsuitable to use. Take, for example, a typical WoT scenario where hundreds of devices are being connected daily

such that multitudes of DS are being added to the network and thousands of sensor readings are being published. How can roles be defined for each user for each device for each DS in this case? It is both illogical and unfeasible to define roles in this dynamic setting, especially when the ability to control access right down to individual DS of data is needed. Furthermore, and as it has been alluded to previously, RBAC is based on roles which are tied to users and therefore promote a user-centric scheme which prohibits anonymous access of resources by non-registered agents.

Context-based Access Control (CBAC) is an extension of RBAC, since it also takes into account the context of the user when requesting access to resources (e.g., user location, device where request is made from, etc.) [72]. However, it still does not remove the user-centricity from the control mechanism.

2.4.1.4 TBAC

TBAC systems are based on the premise of reusable and reconfigurable tokens that grant access to a set or group of resources for a particular user [73]. After generation, they are transmitted to agents who need to consume a set of private resources that are normally hidden from public view and accessible only by the resource owner. Tokens can be configured to only expose the relevant resources and assets without leaking any information regarding the identity of the resource owner. This is advantageous over UBAC which requires the identity of the user to be transmitted with a request. Whereas roles in RBAC are a part of the overall organizational structure and are therefore more permanent and long-term artifacts, tokens in TBAC are much more decoupled and can be easily generated, modified and revoked

without affecting the organization structure. This provides a significant managerial advantage when tokens are used to control access to temporal assets of the network. Finally, since tokens are tied to resources as opposed to users who own those resources, this scheme provides a resource-centric access control scheme which is perfect for managing interactions with resources in an enterprise-grade WoT setting.

2.4.2 Comparison of access control mechanisms

Table 2-3 presents a summary and relative comparison of the various access control mechanisms that have been discussed so far. In the WoT context, the access control mechanism needs to be resource-centric ideally so that it is not tied down to user identities which are not significant in WoT repositories. This necessitates the capability of enabling anonymous (non-registered) agent authorisation to enable access to resources, which of course means that resources need to be shared publicly in the first place. UBAC is highly unsuitable in the temporal characteristic domain of the WoT and ABAC doesn't fare much better either as it is still user-centric. RBAC makes some leaps in masking user identities when cross-domain service requests are made but its user-centricity, again, makes it unsuitable for use on temporal resources and services. TBAC, on the other hand, provides a decoupled resource-centric mechanism of access control which is capable of scaling, efficiently, with the dynamic environment of temporal assets in the WoT. It suffers, in part, from lower security because at its core, TBAC offers a single-step authentication service (i.e. the presence of a token is sufficient to access a service). In contrast, the other schemes generally require two-step authentication which increases security.

From the analysis so far, it can be seen that a flexible and extensible access control mechanism is required to manage access to the various assets that are available in the network. It should be easy to spawn, grant and revoke access rights dynamically when the need arises (as actors emerge onto/leave the scene). It should also be easy and possible to grant access to selected resources without requiring explicit registration of the external party to the network so that inter-department and cross-vendor collaboration can be provisioned on the go. TBAC is evaluated to be the most suitable access control mechanism to achieve this task.

Table 2-3: Comparison of access control schemes in the context of temporal resources in the WoT

	UBAC	**ABAC**	**RBAC**	**TBAC**
Centricity	User	User	User	Resource
Anonymous access?	✗	✗	✗	✓
Public sharing of resources?	✗	✗	✓	✓
ID protection in cross-domain invocations?	✗	✓	✓	✓
Suitable for temporal assets?	✗	✓	✗	✓
Dynamic scaling efficiency	Very low	Low	Medium	High
Security	High	High	High	Lower

2.4.3 Derivation of functional requirements for IAM

To compare existing works in light of the performed analysis and deduced principles, a formal list of functional requirements for IAM is presented below.

FR2: Comprehensive and extensible IAM: The framework should be able to provision access to assets at any level of granularity, from the top-level DF (e.g. an Arduino board and all its related DS) right down to the low-level DP

(e.g. light sensor readings on an Arduino board), without requiring extensive policy rights management and without registration being a prerequisite for access.

1. General Capabilities [Core]:
    a. Capability to manage identities & access for internal (and trusted) users of an organisation;
    b. Capability to manage access for external/temporary (and untrusted) users of participating networks. In the case of external parties, managing identities is not as important or crucial as federating temporal access, enabling audited and controlled multi-party collaboration.
2. General Capabilities [Optional]:
    a. Capability to support federated identities if possible (i.e. authentication provided by 3rd party services like Google and Facebook).
3. Access Management Capabilities [Core]:
    a. Capability to issue & revoke access rights (also called “grants”) for creating, modifying, viewing and deleting DF;
    b. Capability to issue & revoke grants for creating, modifying, viewing and deleting DS;
    c. Capability to transitively apply grants for ease of access. For example, it should be easy to grant access to a DF and all of its DS, without having to explicitly apply this grant for each resource;

d. Capability to easily and conveniently exclude sub-resources from grants applied in (iii). For example, it should be relatively easy to exclude a single DS, or a group of them, from having the same access rights as their parent DF.

4. Access Management Capabilities [Optional]:
    a. Capability to define white and blacklists for a variety of server environment variables (e.g. IP, Referring URI, Browser Agent);
    b. Capability to control the lifespan of grants;
    c. Capability to specify the permitted and forbidden contexts for grants;
    d. Capability to mark grants as "*volatile*" (these access rights must be renewed after some set condition has been met and are designed to improve security and/or remove inactive users), and to provision for their renewal.

Comparison and analysis of existing IAM solutions

Table 2-4 presents a summary and relative comparison of the various solutions in terms of their IAM capabilities. Where "*(implied possibility)*" is mentioned, it means that the necessary functionality has not been mentioned explicitly in the corresponding study/medium but that by applying the derived principles and mechanisms, it is theoretically possible to achieve the desired outcome.

From the currently available commercial solutions, Xively offers the best access policy model for controlling access to temporal and dynamic sensor devices and data. The web service assigns each user a master API key which grants CRUD actions on all resources owned by the user. However, it

hits a roadblock when it comes to using a single key/token to control access to multiple DF and DS, as this is not possible in Xively. A single API key can only manage access to DS of a single device in Xively, and this is not ideal for the WoT domain where greater token flexibility is required.

Table 2-4: Comparison of existing solutions against requirements to satisfy IAM

Requirements/ Solutions	**Fine-grained access control for each of the following resources:**			**Anonymous but audited access to DF, DS and DP**
	DF	**DS**	**DP**	**Further comments**
Commercial and publicly available solutions				
Xively	✓	✓	✓	✓ **Drawback:** One API key can only control access to a maximum of one DF.
ThingSpeak	✓	✗	✗	✓
Paraimpu	✗	✗	✗	✗
Applied IAM solutions in literature				
ARCE in [71]	✓ *(implied possibility)*	✗	✗	✗ **Drawbacks:** Focuses on high-level organisational access as opposed to low-level and fine-grained access to resources.
CBAC in [72]	✓ *(implied possibility)*	✗	✗	✗ **Drawbacks:** Same as above except for the added convenience of being able to refine access according to defined contexts.
CapBAC in [70]	✓ *(implied possibility)*	✓ *(implied possibility)*	✓ *(implied possibility)*	✗ **Drawbacks:** Requires credentials as well as the capability token to access services, so while it seems to enable anonymous access at first, that's not the case.
ABAC for SoA in [69]	✓ *(implied possibility)*	✓ *(implied possibility)*	✓ *(implied possibility)*	✗ **Drawbacks:** No anonymous access, requires policy store.

SENSEI	✔	✔	✔	✖ **Drawbacks:** Several steps need to be performed to enable access from another domain (provide token, negotiate a security session, agree keys, setup security session, and so on.). Anonymous access is impossible.

In comparison, Thingspeak is more limited. Its model only extends access rights to DF, and there is no way to fine-grain access to the individual DS. Even then, the access model is shallow at best as it does not allow selective choosing of individual CRUD operations for the DF, and only offers two preconfigured sets: (1) Write actions (create, update, delete) and (2) Read action.

In literature, [71] presents a web-based EIS that is built upon a Role-Based Access Control model and aims to tackle the issue of managing diversity of actors within an EIS environment when accessing hypermedia (web content) and their ensuing roles and responsibilities within the whole system. It relies on the hypothesis that roles and responsibilities largely remain constant within an organisation and it's the users that change, therefore a RBAC model offers a convenient and maintainable solution that is based on *access policies* and can be modified readily as new requirements arise and is especially useful when the number of users is huge (as is common in web-based systems). Whilst the access-based model proposed by the study produces a highly transparent and secure data access system, the authors exclude the issue of enabling cooperation/collaboration (whether internal or external) within the RBAC model, thereby limiting the full potential of the system and hindering third-party integration, something which is becoming

very important in the modern world of growing publically-exposed APIs. Furthermore, the study does not deal with data/KM issues at all and therefore can only be considered a partial EIS solution at best. A similar (albeit more comprehensive) model is presented in [58] where the authors discuss a service-oriented and resource-based information modelling architecture that is capable of providing users with personalized information and/or services based on their profiles (composed of the user's role and associated tasks within the crisis response team). However, and once again, lack of focus on developing semantics and the inability to share data without requiring explicit registration with the network dampens the outreach of the proposed solution.

Still very much related to RBAC, the CBAC approach in [72] brings enhancements in terms of restrictions based on contexts. The contexts describe the situation in which the request to resources is made and control when the grants can be applied. Restrictions can be imposed in terms of the source (device type, IP, software), location, user role, security level, session restrictions, and similar parameters. The implementation of the access model is, however, quite complex and is more suited towards access control for static services as opposed to fine-grained and dynamic resources.

The CapBAC system presented in [70] is more promising as it provides better scalability and permits temporal access to services. Before a user can access a resource, he/she needs to obtain a "capability token" from a Policy Decision Point (PDP), which considers the details of the requesting user as well as the service in subject and then either grants or denies the token. The method is suitable for temporary access to services and resources and does

not force the user to register with the service provider. This approach solves issues with managing trust between heterogeneous service providers. Still, it can only be considered a semi-anonymous access solution at best since the user still has to register with the PDP and a client profile for the user needs to be maintained.

In all, it is apparent that literature is ripe with RBAC models that have been enhanced to deal with a variety of organisational access control problems. Whilst this is important and great leaps have been made in this regard, it is now becoming equally important to invest in more dynamic and token based access policies for the management and distribution of sensor devices and data in the WoT. Proof of this claim lays in the access policy mechanisms of the biggest commercial IoT repository on the market today, Xively. SAW contributes to this problem by enhancing the access policy mechanism of Xively and making it suitable for an environment where access to temporal, dynamic and volatile sensor devices and data is becoming increasingly important.

2.5 Concluding Remarks

Almost all of the solutions discussed in the literature have very weak cross-vendor collaboration models, if present at all. The main reasons for this are the following:

- The favouring of proprietary schemas to store and represent definitions and data. This data can only be understood by the internal network, and therefore hinders collaboration.
- The lack of semantics in annotating and representing data, producing many definitions and representations of the same data.

Many of the analysed works use rigid, proprietary schemas and force the users to conform to these in a strict manner. This raises the entry-barrier, makes it harder to extend the system, and prevents cross-vendor collaboration. A live example of this is the well-known internet repository called Xively (formerly Cosm and Pachube). Xively uses a rigid and proprietary schema and forces users to adhere to this when defining devices and uploading data. While it works wonderfully inside Xively, it is not so user-friendly for external networks who may want to make use of the wealth of publicly available sensor data in the internet repository. External users will need to develop special adapters to translate Xively's schema into their own, before they can process the information. If, instead, Xively had annotated and represented its data in a semantic fashion, then any external user could have understood that data by simply conforming to the set of ontologies used to annotate the dataset (semantic concepts such as annotation and ontologies are explained later on in the thesis). Furthermore, semantic annotation of data can even enable semi-autonomous machine-processing, thereby yielding even greater returns.

Analysis of the current literature in the field of DM and collaboration systems reveals the dire need for a unified and extensible collaboration model. This collaboration model needs to be flexible enough to cater for cross-vendor collaboration so that the in-house data and knowledge can be shared readily and effectively, whilst at the same time, data and knowledge from suitable 3rd party services can be easily brought in-house and exploited to create advanced mashups and intelligent services. It is paramount to keep in mind that crucial data can no longer be kept in-house and exposed to a select few

through conventional collaboration schemes; there is now an ever-increasing need to integrate with social networks and enable the capability to inspect multiple sources of information (including other (commercial) IoT service providers like Xively, Thingspeak, Paraimpu, etc.). This requires the undertaking of further research in regards to the modelling of data in such a way so as to enable this level of cross-vendor collaboration and data access. Furthermore, many systems reviewed earlier employ RBAC which, while suitable for a set of uniform organisations with similar roles and hierarchies, is insufficient for representing a generic data model which is not organisation-based. For example, if there is a need to grant access rights to an unregistered user (registration not being a pre-requisite for using the framework) to embed a graph generated by a registered user, this can't be done through an RBAC system. A more comprehensive and *decoupled* access control mechanism is required to fulfil these supposedly exceptions that are becoming the norm in the WoT realm. Furthermore, current proposed solutions lack comprehensive semantics and therefore, most of the times, act as monolithic repositories as opposed to decentralised information and collaboration hubs. Modelling data with semantic metadata to enable and promote semi-autonomacity within networks will not only increase the productivity and efficiency of a system as more tasks can be automated, but it should also foster better understanding and sharing of data amongst different vendors since common ontologies can represent data in a unified manner in most networks and thereby facilitating a greater level of collaboration. Overall, it can be concluded that existing implementations in the studied literature have the following limitations and drawbacks:

- Limited Collaboration Facilities: This is by far the biggest limitation in current systems and deserves the most attention in future research simply due to its overwhelming importance in the modern WoT. Because these systems are largely unable to obtain information from multiple sources (or make no effort to do so), they are unable to build a more relevant, accurate and consistent picture of the current environment in a given scenario. For example, in an earthquake, information from affected people in the area could prove invaluable as they can share up-to-date information regarding the situation on the ground in areas perhaps not covered by the response team (for example, due to lack of equipment, facilities and expertise). It is even possible to imagine people uploading data to IoT repositories like Xively as a disaster unfolds (e.g. real-time radiation monitoring in Japan in the wake of the nuclear catastrophe [74]). Being able to use this type of 3rd party and external information within the DM application can prove to be invaluable and it means that the responders, themselves, do not have to cover the entire affected area. Rather they can make use of existing setups and leverage 3rd party data and services to augment their understanding of the situation on the ground and increase their relief capabilities and outreach.
- Lack of Semantics: As we move towards a semantic web, i.e. a state of the web where machines can understand and derive meaning from data that is present on the web providing it is marked up and annotated in a certain way, there is a need to design DM applications and general collaboration frameworks with semantics from the

ground-up. This will primarily foster machine-learning and allow us to automate rudimentary tasks, thereby reducing the time (and complexity) to setup. For example, the first time a multi-sensor platform like a SunSPOT device is connected to the network, the user may have to define the relevant properties and capabilities of the device which are then marked with semantic metadata and stored in the repository. The next time the same or another user wants to connect a similar device to the network, the system can offer "suggestions" to the user as the device description is entered by making use of the existing semantic metadata and Linked Data principles. This leads to a higher likelihood of users specifying properties and capabilities of devices, sensors and data in a unified, standardised manner, and therefore better probability that machines can understand and act upon this data by themselves, with as little manual intervention as possible.

- Unsuitable data and access control models: When design is considered with generality in mind, it is not possible to strictly model data and access rights in an RBAC fashion simply because the idea of "organisations" clashes with the fundamental principle of generality, "decoupled systems". Instead, a more comprehensive and decoupled access control mechanism is required to satisfy (albeit ironically) the constraints imposed by generality.

Looking at the current state of the web and the growing need for open distributed systems, it is proposed that a *semantics-driven*, *service-oriented* and *resource-based asset model* would be ideal for creating a decoupled,

easily extensible, plug and play framework that can be customised for a variety of applications ranging from DM and relief work to monitoring and interacting with next-generation WoT applications (e.g. smart cities). The SAW framework is proposed as an enabler of the above vision.

Chapter 3: SAW - Semantically-enriched & semi-Autonomous Collaboration Framework for the WoT

3.1 SAW Concept

SAW is envisioned as an enabler of the next-generation cross-vendor collaboration through the development of a decoupled, semantics-enabled, service-oriented and resource-based data model and the corresponding collaboration mechanisms. It is important to point out at this stage that the focus of SAW is on developing the actual collaboration mechanisms to achieve the vision of cross-vendor collaboration which necessitates the development of the underlying data model. However, as SAW is designed to be generic in nature, there is no intention to provide all the functionalities required by a DM application, even though the problem of managing disasters effectively through cross-vendor collaboration is used as the test case and scenario for developing and evaluating the prototype. Rather, the aim is to provide the underlying functionality and the necessary mechanisms to enable the extension of SAW to any WoT-related application. Therefore, the focus of SAW is on tackling the problem of collaboration amongst vendors that ultimately do not trust each other but still want to make use of each other's data, information, knowledge and expertise, in a uniform and consistent manner.

The outlining issues faced by WoT applications and the resultant goals in achieving to remedy these issues have already been defined in the preceding chapter. Furthermore, the underlying functional requirements have also been defined and the basis laid for the foundations of the two major systems:

1. Resource-based asset model;
2. Semantics-based interaction models.

The goals mentioned previously necessitate the design of a distributed systems architecture whereby different instances of the framework can be individually maintained by different actors but at the same time, can collaborate not only amongst each other, but also make use of information and knowledge present in 3rd party (and commercial) IoT offerings, as illustrated in Figure 3-1.

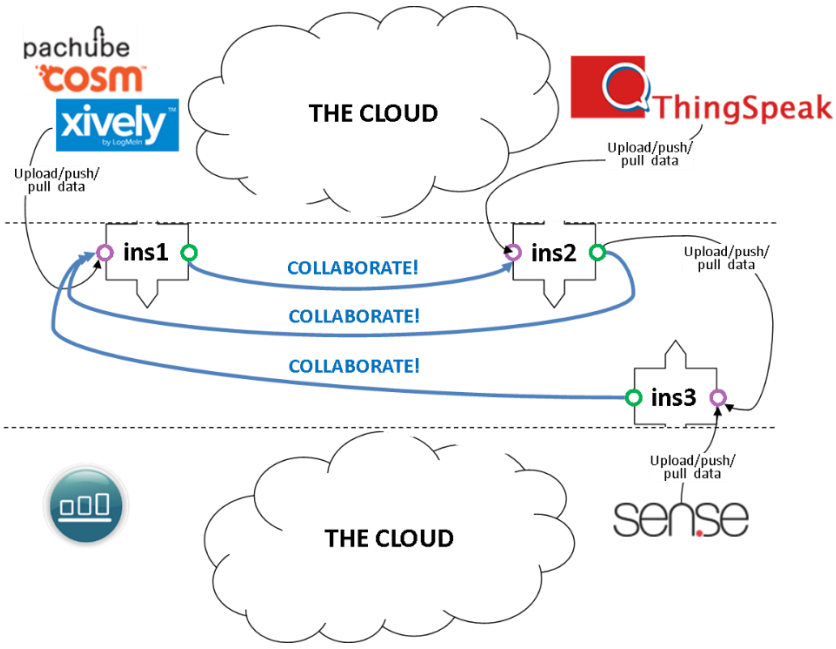


Figure 3-1: SAW: The concept of a distributed system architecture

A system like the one illustrated above is truly decentralised as no single instance has the capability to administer any of the other instances. This means that each instance can operate independently of the other instances

and can be customised for a particular application. If sometime down the line there is need for collaboration with another instance of the system (which might be customised for the same or even a different application), then this can be easily achieved and data, information and knowledge can be conveniently exchanged without relinquishing any partner's control over their private instance. This distributed approach to collaboration also means that underlying assets can be reused and removes the burden of each actor having to collect the same portion of data that may have already been collected, processed, analysed and converted into information and knowledge by another actor.

3.2 System Overview & Architecture

The overall system architecture for SAW is presented in Figure 3-2.

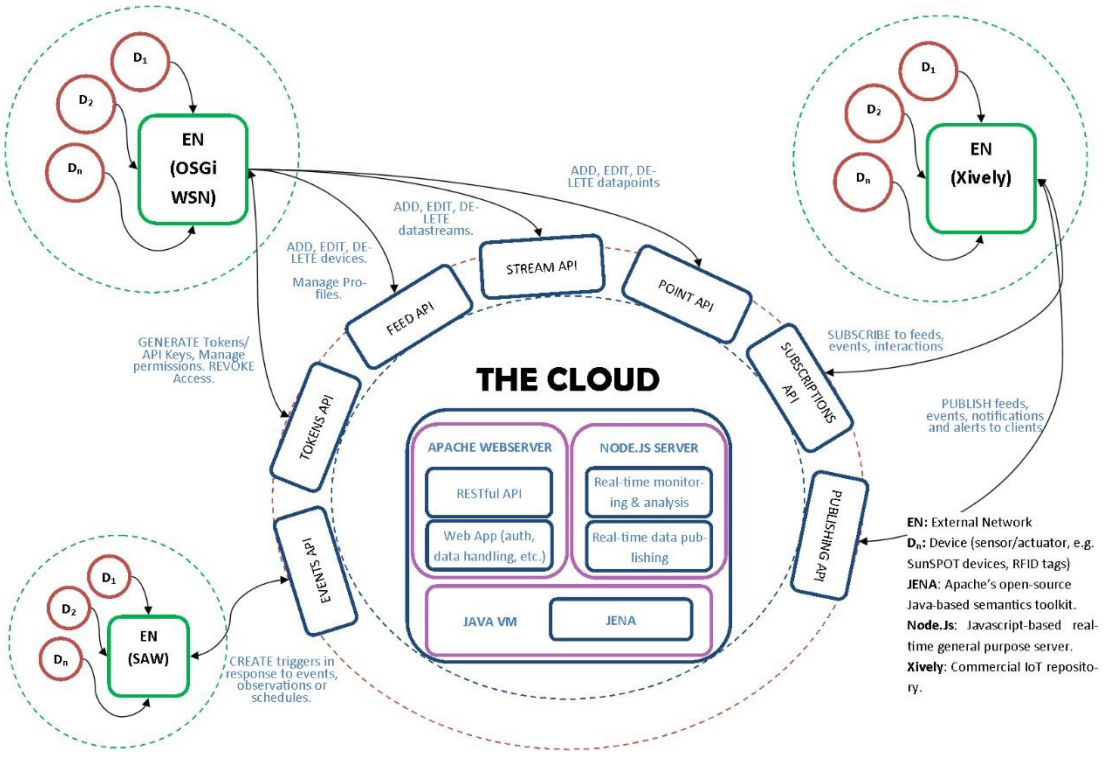


Figure 3-2: SAW system architecture

It consists of:

- External Networks Integration: These might be local Wireless Sensor Networks (WSN), internet repositories or even other instances of SAW itself;
- The SAW Network: The SAW network consists of:
    - An API that enables external networks to communicate with SAW. The API exposes where endpoints to provide different types of interactions with the sensing devices and data.
    - A web application that provides administration capabilities for the SAW network.
    - A real-time server that provides real-time statistics (e.g. number of devices, requests coming in, semantically annotated devices, etc.).
    - A semantics engine that semantically annotates resources. The semantics engine relies on ontologies to define semantic concepts that can be applied to annotated devices.

The above is a comprehensive system architecture and envisions the SAW framework being hosted in a cloud computing environment. However, due to shortness of time and limitation of scope, only the following features are designed and implemented in this thesis (and the implemented prototype):

- External Networks Integration: Integration with a local WSN only. More details about this are provided in Chapter 4: (prototype implementation);
- The SAW Network:
    - Only the following API endpoints are designed and implemented: Tokens (CPPM-TBAC – section 3.4.3), Feeds,

Streams and Points (the Asset Model – section 3.4.1). These enable basic device interactions such as registering devices and uploading sensing data. Other endpoints such as event, subscriptions and publications are not designed or implemented in this thesis.

- The web application.
- The semantics engine. The capability to annotate sensing device properties is provided. The capability to define semantic concepts for measurement data are marked as an item for future work.

In brief, this thesis is not investigating the design or usage of the real-time server, cloud computing hosting and events and publishing/subscription endpoints and these should be considered items for future work.

3.3 Design Considerations

3.3.1 Ontology Selection

SAW is a semantic solution to the generic problem of collaboration in a multi-party environment. The basis of using semantic technologies to tackle and solve this issue of cross-vendor collaboration has been established in earlier chapters and so it is evident that a comprehensive, flexible and all-encompassing ontology needs to be adopted to make possible the design of a generic and extensible collaboration platform. In this regard, the SSN ontology stands out for its adaptability and extensive nature.

SSN can be considered the semantic equivalent of the highly popular OGC SWE suite. In fact, SSN is designed as an enhancement of some of the founding standards of SWE, for example, SensorML and Observations &

Measurements (O&M). This makes the SSN ontology a highly valued semantic adaptation of the popular OGC SWE suite.

The OGC SWE standards provide description and access to data for sensing devices. However, the suite does not provide facilities for abstraction, categorization, and reasoning of devices and data. This is made possible through the semantics offered by SSN. In effect, SSN provides a domain-independent, end-to-end model for sensing applications which is ideal not only for DM applications, but also as a whole for the WoT domain wherever sensing devices and data is considered.

Another reason for the selection of SSN over other ontologies is its expressive nature, and the fact that it has been designed after reviewing many of the major existing ontologies. This has enabled SSN to learn from the mistakes and limitations of existing vocabulary systems and produce a more generic and extensible ontology.

Existing ontologies for sensing devices and data can be split into two predominant categories:

1. Sensor ontologies which focus on defining devices; and
2. Observation ontologies which focus on quantifying the actual observations and measurements processes.

Various efforts exist for each of the above two categories. CSIRO Sensor Ontology [75], OntoSensor [76], Sensor Web for Autonomous Mission Operations (SWAMO) ontology [77], Sensor Data Ontology (SDO) [78], Coastal Environmental Sensor Networks (CESN) ontology [79], WIreless Sensor Networks Ontology (WISNO) [80] and Ontonym – Sensor [81] are example sensor ontologies that have been reviewed by the SSN group as

part of their design process. Similarly, Semantic Reference Systems (SeReS) O&M [82], Stimuli-centered ontology [83], Sensei O&M [84], O&M-OWL (SemSOS) [85] and Socio-Ecological Research and Observation oNTOlogy (SERONTO) [86] are examples of observation ontologies which have also helped shape the final SSN ontology. Since a comprehensive survey and analysis of these ontologies is presented in the SSN final report [87], the same will not be provided here in favour of brevity.

SSN combines efforts of existing sensor as well as observation ontologies to produce an all-inclusive vocabulary system that can be extended to application-specific domains easily. The broadest definitions for concepts have been chosen in this ontology so that in the future, domain-specific sub-concepts can be defined and extended easily and intuitively. The ontology, therefore, allows modelling of sensing devices, their measurement capabilities, operating and survival restrictions and deployments on multi-platform systems and physical sites. The decision to align the core ontology concepts to the DOLCE-UltraLite (DUL) upper level ontology ensures future extensibility of the vocabulary definitions and usage in a broad array of applications.

The SSN ontology is designed around the Stimulus–Sensor–Observation (SSO) pattern [88] and can be viewed from four differing perspectives:

- From the sensor perspective with a focus on the sensing of the device: what is sensing, what is being sensed, and how it is being sensed.
- From an observation perspective with a focus on what is observed.

- From a system perspective with a focus on deployments and multi-sensor platforms (sensors)
- From a feature and property perspective with a focus on observable properties, what senses them and how observations are made about them.

In SAW, since the SSN ontology is used to annotate sensing devices and data, it makes to take the ontological architecture into consideration when designing the data and asset model for the framework. To do this, the SSN ontology needs to be viewed from the system perspective. In this perspective, the following core concepts are used:

- Platform;
- System; and
- Sensors and Devices.

A “System” is the overarching concept which can have many subsystems (Sensors and Devices) attached to it (i.e. sensing devices). Systems can, in turn, be mounted on platforms in deployments. A practical example of this is an Arduino Board which is a “System” (a multi-sensor platform), has subsystems (sensing devices): light and temperature sensors, has platform: laptop, and has deployment: wireless sensor network (or any other arbitrary name used to refer to the system of interconnected devices).

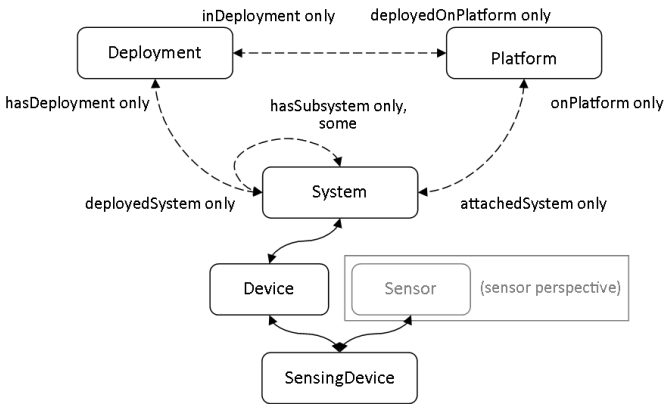


Figure 3-3: SSN System perspective showing relationship between System, Deployment, Platform and Devices

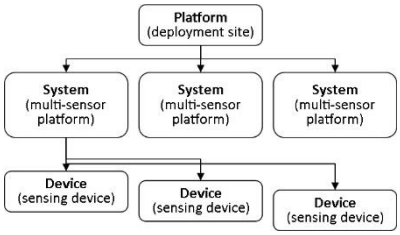


Figure 3-4: Device hierarchy of SSN ontology

Figure 3-3 shows the relationships between core concepts in this perspective. If this was to be modelled in a hierarchical data model, then it would produce a 3-tier hierarchy like the one shown in Figure 3-4 which shows a deployment platform containing many systems (multi-sensor platforms), and each system containing many devices (the actual sensing devices). This is, of course, quite extensive, and retaining all 3 tiers when mapping this semantic model to a non-semantic one might be unnecessary since the Platform (deployment site) and top-tier System (multi-sensor platform) can, essentially, be represented as one entity: a multi-sensor platform. This will achieve the required simplicity in the non-semantic model without impacting the expressivity of the semantic model.

3.3.2 Extension of the SSN Ontology

SAW's semantic annotation system, as is explained later on in this chapter, revolves around the usage of arbitrary tags specified by users of the system to identify devices. Users of the system are not semantically restricted when it comes to defining tags for their devices, so the keywords specified in the payload in the form of tags can be arbitrary strings of data. The first challenge is to represent these arbitrary tags as semantic concepts that can

be linked to sensing devices. The second challenge is to map these arbitrary tags to the sensing devices associated with their representation through the inference of the semantic concepts relating to the specified tags. The final resultant ontology is attached in the appendix.

3.3.2.1 Defining Types of Sensing Devices

The first step is to define the various types of sensors to bootstrap the system and build the initial knowledgebase. Each type of sensor will be a subclass of the *SensingDevice* concept (e.g. TemperatureSensor *rdfs:subClassOf* ssn:SensingDevice). This allows retention of the semantic definitions and restrictions applied to the *SensingDevice* concept from the SSN ontology whilst permitting instantiation of devices as instances of a particular type of sensor. A small variation of sensors has been defined in the initial ontology as shown in Figure 3-5. End-users can extend the ontology to refine definitions for individual sensors or to add further device concepts easily and in an extensible manner.

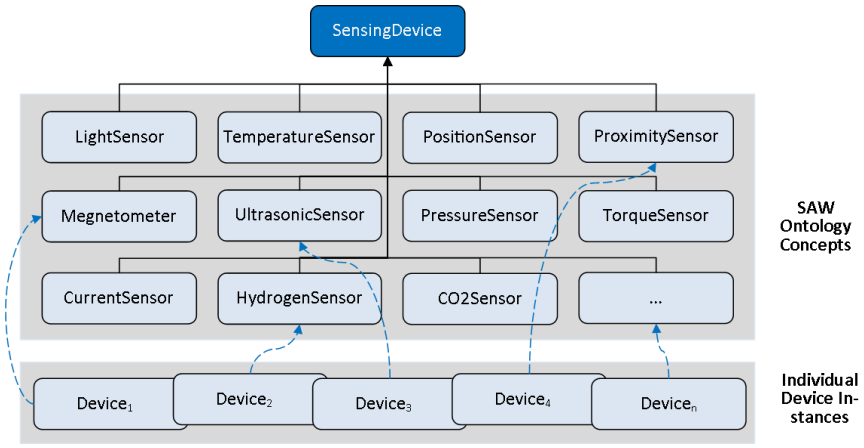


Figure 3-5: SAW sensor type concepts as a subclass of ssn:SensingDevice

If the user is registering a new type of device whose concept doesn't exist in the bootstrapped knowledgebase, then the system should facilitate the addition of the new device concept as a subclass of ssn:SensingDevice. This

will enable community-driven enrichment of the knowledgebase and extension of the system.

Furthermore, the ontology uses the *owl:sameAs* property to contribute to published linked data. Figure 3-6 shows an excerpt from the ontology where the *CO2Sensor* concept is being defined. This concept corresponds to a sensing device measuring the carbon dioxide gas. An *owl:sameAs* assertion is made on line 252 in the figure to indicate that this concept is similar to another sensing device concept which exists in DBpedia and identified by the URI: *http://dbpedia.org/resource/Carbon_dioxide_sensor*. This concept of "class equality" allows the ontology to be extended in the future and interlinked with published semantic metadata on the web to integrate with similar semantic concepts from other knowledge bases.

```
243     <owl:Class rdf:about="http://saw.local/sw/ontology#CO2Sensor">
244         <rdfs:label>Carbon Dioxide Sensor</rdfs:label>
245         <dc:title>Carbon Dioxide Sensor</dc:title>
246         <rdfs:subClassOf rdf:resource="&ssn;SensingDevice"/>
247         <rdfs:comment>A carbon dioxide sensor or CO2 sensor is an instrument for the measurement of carbon dioxide gas.
248         <rdfs:isDefinedBy>http://saw.local/sw/ontology</rdfs:isDefinedBy>
249         <dc:creator>Mohammad Amir</dc:creator>
250         <dct:created rdf:datatype="&xsd;date">2013-10-14</dct:created>
251         <dct:modified rdf:datatype="&xsd;date">2013-10-14</dct:modified>
252         <owl:sameAs rdf:resource="http://dbpedia.org/resource/Carbon_dioxide_sensor" />
253     </owl:Class>
254 </rdf:RDF>
```

Figure 3-6: Excerpt from SAW ontology showing the CO2Sensor concept

A similar approach is adopted to define concepts for multi-sensor platforms as shown in Figure 3-7. On line 34, a concept is defined for an Arduino board, a multi-sensor platform. On line 36, a primary tag for the concept is added to the definition. This process will take place when a new device is added to the network.

On line 38, an instance of this multi-sensor platform is created and on the following line a primary tag is inserted for this new instance. When searching for tags corresponding to a device, both the parent concept and the individual instances will be traversed to produce an all-inclusive list of tags.

```
#   -- Define an Arduino board concept
saw-ont:Arduino                 rdf:type                    ssn:System;
                                rdfs:comment                "Arduino Board Concept";
                                saw-ont:DevicePrimaryTag    ("arduino").
#   -- And then create an instance of it
saw-feed:Arduino-001            rdf:type                    saw-ont:Arduino;
                                saw-ont:DevicePrimaryTag    ("arduino board").
```

Figure 3-7: Excerpt from a device definition file showing description of a concept for the Arduino multi-sensor platform

3.3.2.2 URI Structure for Ontology Concepts

A logical URI structure is used to reference not only the ontological concepts defined above, but also the sensing devices and their deployment characteristics within the network. To begin, the base URI is set to: *http://saw.local/sw/* (please note that the domain name, “saw.local” is used for illustrational purposes only and resolves to a local instance of the network, and is thus not available on the web directly). Then the ontology URI is set to: *http://saw.local/sw/ontology#.* All ontological concepts derived by SAW are referenced to this URI, for example, *http://saw.local/sw/ontology#PressureSensor* for referring to the pressure sensor ontological concept, and *http://saw.local/sw/ontology#Arduino* for referring to the Arduino multi-sensor platform class. The individual instances of DF and DS are referenced to the URIs *http://saw.local/sw/feeds#* and *http://saw.local/sw/streams#* respectively. This would imply that an instance of an Arduino board labelled “Arduino-UoB-001” would be referenced to the URI *http://saw.local/sw/feeds#Arduino-UoB-001*. Similarly, a position sensor labelled “PositionSensor-UoB-001” would be referenced to the URI *http://saw.local/sw/streams#PositionSensor-UoB-001*. A coherent and comprehensive URI is essential for developing an easily traversable linked data map of the semantic information contained within the knowledgebase. The URI structure presented above achieves this aim by splitting up assets

into logical groupings and separating the ontological concepts from individual instances.

SAW also defines URIs corresponding to three other overarching concepts found in the SSN ontology: ssn:System (multi-sensor platforms), ssn:Platform (host machines on or through which the multi-sensor platforms operate, such as PCs, laptops, servers, etc.) and ssn:Deployment (characteristics of the actual deployment, for example, location information). The corresponding URIs for these three concepts are *http://saw.local/sw/deployment/system*, *http://saw.local/sw/deployment/platform* and *http://saw.local/sw/deployment* respectively. Using this URI scheme, SAW is able to easily distinguish between the different levels of granularity involved in the semantic mapping of devices and their properties in the network.

3.3.2.3 Representation of Tags as Semantic Concepts

SAW extends the SSN ontology to define concepts to represent tags. The main concept is an owl object property termed *DeviceTag*, with the sub-properties: *DevicePrimaryTag* and *DeviceSecondaryTag* as shown in Figure 3-8. The *DevicePrimaryTag* is defined as a tag that has a direct and unambiguous relation with the tagged device, for example, the tag “temp” for a sensing device that is an instance of the *TemperatureSensor* concept. *DeviceSecondaryTag*, on the other hand, is defined as a tag that has an indirect and possibly ambiguous relation with the tagged device, for example, the tag “position” for a sensing device that is an instance of the *ProximitySensor* concept, or the tag “sensor” for any device. Finally, by applying an *rdfs:domain* restriction on the *DeviceTag* concept to ssn:System,

it is ensured that the tag properties can be applied to any type of device modelled through the SSN ontology.

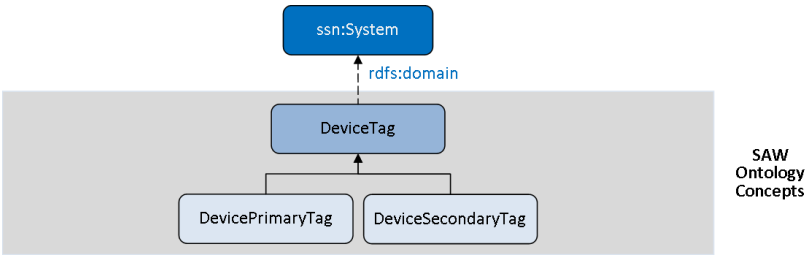


Figure 3-8: Illustration of the "DeviceTag" tag in the SAW ontology

Since the *DeviceTag* is modelled as an owl object property, it can be assigned to instances of *ssn:SensingDevice* and its sub-concepts. For example, Figure 3-9 shows an excerpt in the Turtle format that first creates an instance of the *CO2Sensor* concept, and then assigns tags to it:

```
1  saw-stream:CO2Sensor-Ard-001      rdf:type                    saw-ont:CO2Sensor,
2                                                                owl:NamedIndividual;
3                                    rdfs:comment                "Carbon Dioxide sensor
4                                                                attached to an Arduino
5                                                                board.";
6                                    saw-ont:DevicePrimaryTag    ["co2", "carbon dioxide"];
7                                    saw-ont:DeviceSecondaryTag  ["sensor", "arduino"];
```

Figure 3-9: Sample Turtle excerpt showing device instantiation and tagging

Lines 1 & 2 instantiate a sensing device called “CO2Sensor-Ard-001” as an instance of the *CO2Sensor* concept. A list of primary tags for the device are then defined on line 6. These are tags that have a direct and easily identifiable relation with the tagged device. Line 7 then defines some secondary tags which don’t have any direct association with the tagged device but might prove useful as a stepping stone. The real challenge, however, is to categorise tags defined by the user as either primary or secondary tags, and then to append these onto the actual sensor type definitions so that a degree of semi-autonamicity can be achieved in future annotations.

Putting this altogether, individual types of sensing devices can now be modelled through the corresponding *ssn:*SensingDevice sub-concepts from the SAW ontology. Each of these sub-concepts can then be enriched by specifying a list of tags (both primary and secondary), which can be used to profile new devices being added to the network according to the correct concept.

3.3.2.4 Methodology for Extracting & Classifying Tags

An initial list of primary and secondary tags can be specified for each concept in the beginning to bootstrap the system. Eventually, however, an actual mechanism will be needed to extract user defined tags for a device. This mechanism will revolve around one of the following scenarios:

- The device being modelled is a new type of sensor whose concept does not exist in the knowledgebase;
- The device being modelled has a corresponding concept in the knowledgebase.

If the device being modelled is a new concept, then the system can define the new concept and assign *non-ambiguous tags* as the primary tags for the new device. Non-ambiguous tags are those tags which don't already have an association with another sensor type concept in the knowledgebase. All other tags should be assigned as secondary tags for the new concept.

The mechanism for classifying tags of a device belonging to an existing concept is slightly more complex. First the user-defined tags will be used to search the knowledgebase for any associations to a sensing device concept. If no associations can be found, then either the user has provided an inadequate list of tags, or the device concept does not exist in the

knowledgebase. The mechanisms for dealing with this are detailed in the semantic annotation section.

If a primary association has been found (i.e. one of the provided tags matches an instance which is *DevicePrimaryTag* of a sensing device), then the device should be modelled as the corresponding sensing device concept. Any tags provided for the new device by the user registering the device should then be added as secondary tags for the corresponding sensing device concept provided that they are not already modelled as *DeviceTag* properties for the sensing device concept.

If more than one primary association has been found, then the secondary tags need to be processed to produce a similarity rank. The primary association with the most secondary tags associated to it should be chosen. If more than one primary association still remains after this process, then the user should be presented with the associations and asked for the final selection. If this is not possible (e.g. when mining data from other repositories that don't implement the feedback loop), then an association should be chosen at random.

If no primary associations are found but one or more secondary associations are found, then the secondary association with the largest number of corresponding tags should be chosen. If there is a tie, then the user should be presented with the associations and asked to make the final selection. If this is not possible, then an association should be chosen at random.

Finally, it is highly recommended to exclude from the onset or remove later on really ambiguous concepts like "sensor", "device", "sensing device", etc.

from the knowledgebase as these can lead to false positives resulting in faulty modelling of devices.

3.3.3 Database Types

Two different types of databases need to be used to store the different types of data present in SAW.

First of all, a relational database system called MySQL needs to be used to store data with a known and fixed schema. Examples of this include user details, user permissions, user group assignment, system and error logs, etc. At the same time, a non-SQL database is needed to store data that has an unknown or dynamic schema that changes depending on the object being stored. Examples of this type of data include token permissions (there can be one permission policy definition or more than 10, each with different fields, ids, keywords, etc.), and DF, DS and DP definitions (each with varying properties). MongoDB is an example of a prominent no-SQL and non-relational database system and needs to be used to store token policies and DF, DS and DP definitions and sensing data.

3.4 Proposed System Architecture

In line with the functional requirements, achieving the vision for SAW necessitates the creation of an abstract and resource-based asset model and a service-oriented and semantic interaction model. The resource-based asset model is essential for provisioning multiple layers of inspection so that assets can not only be built as mashups, but also decomposed into their fundamental origins upon inspection. The service-oriented interaction model will foster the birth of a set of decentralised and distributed collaboration mechanisms and, alongside the resource-based asset model, will lay the

foundations for an extensible collaboration model. Put together, these two approaches will enable representation of assets at different levels of granularity and expressiveness (from data to information to knowledge and the other way around), with the capability to easily expand their semantic definitions through generic and extensible templates. Secondly, the distributed nature of the interaction model will make it possible for individual instances of SAW to contain custom extensions and enable the instance administrators to augment the framework with problem-specific functionalities without affecting its ability to interact with other instances. Finally, by exposing the framework as a RESTful API, it becomes convenient and feasible for 3rd party networks to use the network's assets and vice versa.

3.4.1 Asset Model

3.4.1.1 Conceptual Architectures

Any resource that is captured, processed, derived or published is considered a network asset in SAW. The purpose of modelling assets is to allow abstraction of resources and empower the framework with the capability to define generic and extensible templates. The by-product of this process (and a much-needed functionality of the framework) is the possibility to compose assets from raw resources to form complex mashups, and then to also decompose compounded representations into their fundamental origins. The following steps are undertaken when modelling assets:

1. Define data in terms of its *expressiveness*: This is the capability of assets to hold a meaningful representation.
2. Define a data *hierarchy* so that assets can be categorised in terms of granularity.

In SAW, there exist 3 layers of data in terms of its *expressiveness* (Figure 3-10), as has been mentioned previously in section 2.4:

1. Data, which is the raw and unprocessed representation of sensor data;
2. Information, which is the processed and tagged sensor data that may or may not be semantically annotated; and
3. Knowledge, which is the derived rich set of information composed through via semantic technologies.

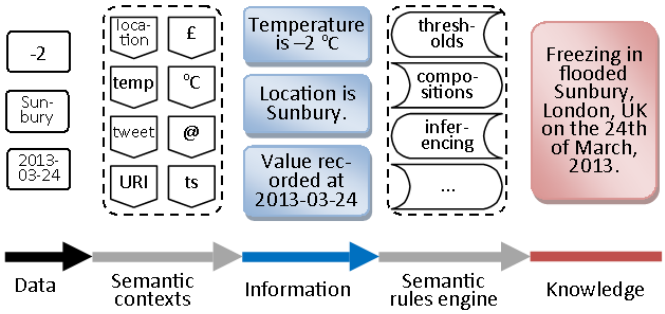


Figure 3-10: Data expressiveness in SAW

This structure makes it possible to break down high-level knowledge into the underlying information and even down to the very fundamental raw pieces of data which is useful for introspection of assets. Furthermore, the process of producing information from data and then deriving knowledge from information through the use of semantics presents a common methodology for participating networks to generate and understand network assets.

SAW provides a simple but extensible data hierarchy as illustrated in Figure 3-11 and explained previously in section 2.4. A DF implements a generic device template which can be used to model and represent any kind of physical or virtual device, for example, an Arduino board or a twitter user respectively. A DF has 1 or more DS that describe a particular sensor or actuator asset of the DF, for example, a light sensor on an Arduino board or

a twitter user's tweet DS. Finally a DS can have 0 or more DP, where each DP references a particular value at a given instance in time, for example, a time-stamped light sensor value or a particular tweet from the DS of a twitter user. It can be seen that this model is derived through the simplification of the SSN device hierarchy with DF representing the top-tier "System" and DS representing the "Device" concepts of the ontology. Aligning the non-semantic data model to the semantic device hierarchy of the SSN ontology ensures that SAW is easily able to map devices and data to the relevant concepts in the ontology, whilst retaining a sense of freedom and flexibility in the non-semantic modelling of devices and data.

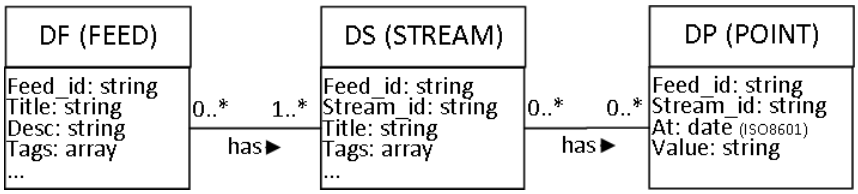


Figure 3-11: Data hierarchy

DF, DS and DP are described initially using a set of generic templates called the Generic Device Definition (GDD) templates which are domain-independent and can be customised to fit into any application domain. Once these initial payloads have been submitted to register the devices, further semantic templates referred to as Semantic Annotation (SA) templates can be used to add well-defined and cross-network semantic metadata to the device definitions. SA templates are discussed later on in this chapter.

The GDD templates constitute the initial generic templates which are mainly used to register the devices to the network. The purpose of these payloads is to provide an intuitive and flexible template which can be used to define and register devices at a very basic and non-semantic level. GDD templates are therefore:

1. Extensible, so that more fields can be added as and when needed; and
2. Transport-independent so that they can be represented in any data transport technology (XML, JSON, etc.). In SAW, the JSON format is used by default as it is more lightweight and widespread in the RESTful web than other representations.

3.4.1.2 Generic Device Definition Templates

When DF (whether physical or virtual) are connected to the network, they need to be registered to the network before they can be deployed and interacted with. To do this, the user specifies a set of mandatory properties for the device being registered, but is free to add other arbitrary definitions which are not understood but can be supported by the network, but can still be used to interact with the device. These templates are described below.

3.4.1.2.1 DF Template

The DF template is used to register a multi-sensor device platform, such as a SunSPOT device or an Arduino board. Figure 3-12 shows a DF template with only the three network-defined fields in the template.

- Title: The name or identifier of the device which can easily distinguish and identify the device. This doesn't have to be a unique identifier.
- Visibility: Either *public* or *private*. Public devices can be viewed without the use of a token (see the section on CPPM-TBAC) whereas private devices can only be viewed if the requester produces a valid token. In the absence of this definition, the visibility of the device is set to *public*.

- Tags: A comma-separated list of tags that identify some property of the device. These are not required, but highly recommended since they form the basis of the semantic annotation interactions (detailed later on in this chapter).

```
{
    title: "Arduino 001",
    visibility: "public",
    tags: [ ]
}
```

Figure 3-12: GDD template for a DF with only the network-defined fields

```
{
   title: "Arduino 001",
   visibility: "public",
 - tags: [
       "Arduino Board",
       "Light Sensor",
       "Temperature Sensor",
       "Accelerometer"
   ],
   state: "active",
 - properties: {
       property1: "string1",
       property2: "string2",
       property4: "integer2"
   },
 - location: {
       latitude: "53.792235",
       location: "University of Bradford, Chesham Building"
   }
}
```

Figure 3-13: GDD template for a DF with additional arbitrary definitions

The template presented above is, of course, very rudimentary. In effect, that is the entire purpose of GDD templates: that they are generic and simple. This is feasible because the actual semantics are added to the devices later on via the SA templates, so there is no need for complicating the baseline device registration templates. But this does not mean that these templates cannot be expressive or extensible. Figure 3-13 shows a slightly more expressive registration payload where the user has specified additional arbitrary fields to enhance the description of the multi-sensor platform. Whilst the additional fields hold no semantic value as far as the SAW platform is concerned, they can nonetheless be treated as such with extensions to the system. Therefore it can be seen that SAW offers unparalleled functionality and freedom by offering generic and extensible device registration templates which promote usage of the platform by lowering the entry barrier and catering for extensibility within the framework.

Once the user fills the above template and sends it to the network for processing, the device registration process can take place and upon successful registration, the user will receive a unique device ID (labelled "*feed_id*") that forms the resultant URI which can be used to browse to and interact with the device. This is described in greater detail in the section pertaining to the implementation of the framework.

3.4.1.2.2 DS Template

The DS template is almost exactly like the DF template, with the exception of one additional mandatory field: *feed_id* (Figure 3-14). This new mandatory field (if the DF ID is not specified in the URI) specifies which DF the new DS is being added to, since each DS must have a parent DF.

```
{
    feed_id: "PARENT_FEED_ID",
    title: "Light Sensor",
    visibility: "public",
    tags: [ ]
}
```

Figure 3-14: GDD template for a DS with only the network-defined fields

Similarly, users can extend the core GDD template for DS by specifying additional arbitrary fields and properties for their devices, just like they are able to when registering DF.

The visibility of a DS might be restricted depending on the visibility of its parent DF. If the parent DF is a *public* device, then the child DS can take either the *public* or the *private* visibility. If, however, the parent DF is a *private* device, then the child DS **must** specify a *private* visibility scope. This means that sensing devices attached to private multi-sensor platforms are always private and require a token with the necessary grants before they can be browsed or interacted with. If the user has a multi-sensor platform with an array of sensing devices but only wants to make a few of them open for

public dissemination, then this can be easily accomplished by setting the visibility of the corresponding DS to *public*, and leaving the rest with the *private* visibility scope.

3.4.1.2.3 DP Template

After the two-tier devices have been defined using the DF and DS templates, only the problem of uploading the actual sensor data remains. This is done through the simplistic DP template as shown in Figure 3-15.

```
{
    feed_id: "PARENT_FEED_ID",
    stream_id: "PARENT_STREAM_ID",
    at: "2013-06-29T15:24:54+00:00",
    value: "Temperature was very hot"
}
```

Figure 3-15: GDD template for DP with only the network-defined fields

In a DP template, the only network-defined fields are:

- DF and DS ID: All DP belong to a DS which, in turn, belongs to a DF. These two fields are not mandatory if the respective IDs are present in the request URI, e.g.:

```
POST
http://saw.local/api/v1/feeds/FEED_ID/streams/STREAM_ID/points
```

- At: A date-time stamp (formatted according to the ISO 8601 date and time standard) which specifies when the observation took place;
- Value: A string representing the observed value at the time the observation took place. This can be any type of string or integer.

Again, the DP template can be extended even though there might be very limited reasoning for doing so since the actual device definitions are already stored in the DP and DS templates. For the sake of extensibility, however, this functionality has been maintained as with the DF and DS templates.

If numeric values are specified for a reading, then it is easier for the framework to chart these values in a time series on a graph. The same may not be possible for textual and descriptive sensor readings such as “hot”, “cold”, etc. However, the functionality can certainly be developed within the framework should the need arise due to the extensible nature of the templates and the service-oriented architecture of the framework.

3.4.2 RESTful Resource Exposition

3.4.2.1 Overview of REST and URIs

To enable web-based interaction based on resources, the framework needs to extend an API based on the RESTful architecture of the web. This architecture is chosen because the vision of WoT hinges on turning everyday connected things into web-based resources that can be browsed and interacted with much like we browse and interact with webpages today. To enable this vision over the HTTP protocol of the internet, the RESTful architecture needs to be adopted [89].

REST, which stands for Representational State Transfer, provides a resource-based, web-oriented architecture for achieving interoperability and decoupling of distributed applications on the web [90]. It is more lightweight, widespread, and simpler than the more verbose and complex WS-* (Web Services) suite (based on SOAP). A RESTful architecture leverages all the inherent power and features of HTTP to deliver services to the accessing agent (e.g. a web browser) by modelling objects and services to be interacted with as resources that can be browsed, navigated, linked and bookmarked.

The RESTful architecture relies on Uniform Resource Identifiers (URIs), which are more commonly known as Uniform Resource Locators (URLs) on the web. The origins of the URIs began as document identifiers on the web which pointed to a document's location on the network. This definition was quickly changed as it became apparent that URIs did not always point to documents, and could essentially be used to refer to any type of artefacts presented on the web. URIs, thus, were redefined as endpoints that lead to resources [90]. Now that resources can be identified through URIs on the web, the only remaining problem is now one of interaction with these resources. REST solves this problem through the following four well-defined HTTP verbs:

- GET: Request/browse to a resource identified by the URI;
- POST: Create a new resource according to the presented URI and the attached payload;
- POST: Update an existing resource according to the attached payload and identified by the URI;
- DELETE: Delete the resource identified by the URI.

Through the above four HTTP verbs, it is possible to browse and interact with a web-based resource in any manner possible.

3.4.2.2 RESTful API for SAW's Asset Model

The resource-based asset model in SAW is exposed through a RESTful API, just like the growing number of Web 2.0 applications that use the same principle to expose their services for mass consumption. Basic CRUD (Create, Read, Update, Delete) operations on resources are enabled through the use of the corresponding HTTP verbs: POST for creating, GET for

reading, PUT for updating, and DELETE for deleting resources. In SAW's own terminology, POST, PUT and DELETE actions are collectively referred to as *modify* actions, whilst GET is referred to as a *read* action.

All actions require a token with the necessary grants before the action can be carried out. This requirement is imposed regardless of whether the subject resource has been set as *private* or *public* visibility. *Read* actions on *public* visibility resources are the only exception to this norm as in this case, no token is required to carry out the action. The workings of access control are detailed later on in this section, so only the process required to specify tokens in the request is detailed here.

Tokens can be specified with the request in one of two ways: (1) Through the HTTP Headers and (2) As a URL query parameter. The former method is preferred and leads to better URIs. With the former method, the token must be specified in the HTTP Headers. This can be done by specifying a new header field/key called "X-ApiToken", and then specifying the token as a value of this new header field like so: X-ApiToken: 5195feafe. If this is not possible, then the second method can be used. In this method, the token needs to be appended as a query string to the end of the URI with the query parameter: "token". An all-inclusive URI with an appended token would look something like this: *http://saw.local/api/v1/feeds?token=5195feafe*. Both methods achieve the same result, but the former is preferred since it does not clutter the URI.

Similarly, the client must also remember to specify a "Content-Type" HTTP Header. This basically informs the server of the client's wishes in retrieving the response in a certain fashion (e.g. JSON document, XML document,

HTML document, etc.). By default, the server returns all responses in JSON format.

DF, DS and DP are manipulated as resources through the API by virtue of the four HTTP verbs. Each resource has a clearly identifiable and traversable URI as is detailed below. Please note that the domain name, “saw.local” is used for illustrational purposes only and resolves to a local instance of the network, and is thus not available on the web directly. Similarly, the trailing “api/v1” is used by the framework to access the API endpoints corresponding to the version specified, which is “v1” in this case. Thanks to the service-oriented architecture of SAW, the framework could be designed so that it can easily accommodate updates and extensions. The versioning of the API helps ensure that clients can keep on using an older version of the API when (and if) updates are made to the programming interface.

3.4.2.2.1 DF Endpoints

To create a new DF (i.e. device registration); a POST request needs to be submitted to the POST URI shown in Table 3-1 alongside the necessary payload. A sample is presented in Figure 3-16 to illustrate the process.

The *visibility* property is required (*public* or *private*) alongside the *title*, as has been mentioned before. Everything else is optional, but it is highly recommended to set the DF description for ease of identification, and specify some tags to enable the semantic annotation of the device.

If the DF is successfully created, then the response will contain a HTTP Header called “location” which contains the URI for accessing the newly created DF.

Table 3-1: Resource endpoints for DF

Action	URI (http://saw.local/api/v1 has been omitted for the sake of readability)
POST	/feeds
GET	▪ View a single resource: /feeds/FEED_ID ▪ View a list of resources accessible by the specified token: /feeds
	▪ Sample: Fetch the DF with ID 51c: /feeds/51c
PUT	/feeds/FEED_ID
	▪ Sample: Update the DF with ID 51c: /feeds/51c
DELETE	/feeds/FEED_ID
	▪ Sample: Delete the DF with ID 51c: /feeds/51c

Method: POST, URI: /feeds

```
{
"title":        "New Device",
"desc":         "My New Device Description",
"visibility":   "public",
"tags":         ["tag1", "tag2", "tag3"]
}
```

Figure 3-16: Sample payload for creating a new DF

The process of updating the DF is similar to that of creating it. The only difference is that the HTTP verb used now is the PUT verb and the ID of the DF is required in the URI. The payload, illustrated in Figure 3-17, should only contain values for fields that are being updated.

Method: PUT, URI: /feeds/FEED_ID

```
{
"desc":         "My New Device Description - UPDATED",
"visibility":   "private",
"tags":         ["newTag1", "newTag2"]
}
```

Figure 3-17: Sample payload for updating an existing DF

Deleting a DF resource is far simpler. It only requires a request to be made with the DELETE verb and the ID of the DF to be deleted appended to the URI, as follows:

Method: DELETE, URI: /feeds/FEED_ID

Fetching an already existing resource is also quite simple as it requires no payload to be specified with the request. The requester should make the request using the GET verb and append the ID of the DF of interest to the URI, as shown below:

Method: `GET`, URI: `/feeds/FEED_ID`

This will return a response like the one shown in Figure 3-18:

Figure 3-18: Sample response when fetching a DF

It is even possible to fetch a list of feeds that are accessible by the specified token by trimming the DF ID from the URI like so:

Method: `GET`, URI: `/feeds`

This will return a list of DF as shown in Figure 3-19. The list contains unique IDs of the DF that are accessible by the specified token. In turn, these IDs can be appended to the request URI to fetch more details about that DF.

Figure 3-19: List of DF that are viewable by the specified token

3.4.2.2.2 DS Endpoints

DS endpoints branch off from DP endpoints since DS belong to DF. This produces a resource architecture that is logical, consistent, linked and traversable.

Creating, updating, deleting and fetching DS resources is just like manipulating DF resources with the exception that the API endpoints are different. The only other difference is that for POST and PUT requests, the parent DF ID needs to be specified in the payload if it is not present in the URI. If the URI structure being used is similar to the one presented in Table 3-2 where DS are branched off of DF, then specifying the DF ID in the device registration and update payloads is not necessary as it's already present in the URI. If however, an instance of the framework decides to use a more concise URI format which excludes the DF ID from the URI (e.g. PUT /streams/STREAM_ID), then the DF ID will need to be specified explicitly in the payload of POST and PUT requests.

When a new DS is created successfully, a HTTP Header called "location" will be returned with the response to identify the unique ID of the newly created resource and the full URI which can be used to interact with the new DS. Similar to viewing DF, a list of DS can be fetched by trimming the DS ID from the GET URI. The ID of each DS can then be taken and appended to the URI request to browse the individual DS resources.

Table 3-2: Resource endpoints for DS

Action	URI (http://saw.local/api/v1 has been omitted for the sake of readability)
POST	/feeds/PARENT_FEED_ID/streams
GET	▪ View a single resource: /feeds/ PARENT_FEED _ID/streams/STREAM_ID ▪ View a list of resources accessible by the specified token:

	/feeds/PARENT_FEED_ID/streams
	▪ Sample: Fetch the DS with ID 51d, belonging to DF with ID 51c: /feeds/51c/streams/51d
PUT	/feeds/ PARENT_FEED _ID/streams/STREAM_ID
	▪ Sample: Update the DS with ID 51d, belonging to DF with ID 51c: /feeds/51c/streams/51d
DELETE	/feeds/ PARENT_FEED _ID/streams/STREAM_ID
	▪ Sample: Delete the DS with ID 51d, belonging to DF with ID 51c: / feeds/51c/streams/51d

3.4.2.2.3 DP Endpoints

DP endpoints branch off from DS endpoints since DP belong to DS. Again, this architecture is adopted to produce a logical, consistent, linked and traversable resource tree.

Creating, updating, deleting and fetching DP resources is just like manipulating DF and DS resources with the exception that the API endpoints are different. The only other difference is that for POST and PUT requests, the parent DF and DS IDs need to be specified in the payload if they are not present in the URI. If the URI structure being used is similar to the one presented in Table 3-3 where DP are branched off of DF and DS, then specifying the DF and DS IDs in the DP creation and update payloads is not necessary as they're already present in the URI. If however, an instance of the framework decides to use a more concise URI format which excludes the DF and DS IDs from the URI (e.g. POST /points), then the DF and DS IDs need to be specified explicitly in the payload of POST and PUT requests.

Table 3-3: Resource endpoints for DP

Action	URI (http://saw.local/api/v1 has been omitted for the sake of readability)
POST	/feeds/PARENT_FEED_ID/streams/PARENT_STREAM_ID/points
GET	▪ View a single resource: /feeds/ PARENT_FEED _ID/streams/PARENT_STREAM_ID/points/POINT_ID

	▪ View a list of resources accessible by the specified token: /feeds/PARENT_FEED_ID/streams/ PARENT_STREAM_ID/points
	▪ Sample: Fetch the DP with ID 51e, belonging to the DS with ID 51d, belonging to DF with ID 51c: /feeds/51c/streams/51d/points/51e
PUT	/feeds/ PARENT_FEED _ID/streams/ PARENT_STREAM_ID/points/POINT_ID
	▪ Sample: Update the DP with ID 51e, belonging to the DS with ID 51d, belonging to DF with ID 51c: /feeds/51c/streams/51d/points/51e
DELETE	/feeds/ PARENT_FEED _ID/streams/ PARENT_STREAM_ID/points/POINT_ID
	▪ Sample: Delete the DP with ID 51e, belonging to the DS with ID 51d, belonging to DF with ID 51c: /feeds/51c/streams/51d/points/51e

3.4.3 Enhanced Token-Based Access Control Mechanism

The constant fluctuations and rapid variations in data present in the WoT makes traditional access control mechanisms such as User-Based, Authorization-Based and Role-Based Access Control (UBAC, ABAC and RBAC respectively) highly unsuitable for the task at hand, which is to flexibly manage access to (often times, dynamically generated) data at varying levels of granularity. In a WoT setting, it cannot be assumed that the users of the system are known, i.e. access by anonymous data access points, which may be users or other machine endpoints, needs to be catered for and provisioned within the system [91]. Token-Based Access Control (TBAC) mechanisms cater for this need of secure and anonymous data access through interrogation of RESTful resource endpoints, but just on their own, do not contain the flexibility to refine access to fine levels of granularity without resorting to mass-generation of tokens, which is not manageable in a realistic WoT application. This study introduces a Cascading Permissions Policy Model (CPPM) for the TBAC system (henceforth referred to as CPPM-TBAC) such that access control policies can be extended to not only allow finer control over granularity of visible resources, but also contain context-

specific parameters that can further refine access based on the request origin context.

3.4.3.1 Catering for Unbounded, Temporal and Dynamic Resources

Repositories in the WoT have very different characteristics than traditional data stores. In the WoT context, consideration has to be given to the potential of handling an unbounded number of devices (sensors, actuators, and virtual entities), services (composition, processing, transformation, etc.) and interactions (capture, publication, querying, etc.) [92]. Furthermore, the resources themselves are much more temporal and short-lived which gives birth to dynamic and unpredictable application scenarios and interaction patterns [93]. In short, the following characteristics of cloud-based WoT repositories can be concluded:

- Unbounded: New devices, services or interactions can be introduced at any time. For example, new devices may be introduced as more equipment becomes available at a disaster scene.
- Temporal: Resources are generally short-lived and undergo various changes in their properties and definitions. For example, legacy or faulty devices will be replaced with newer or more capable platforms over time. Also, the repositories may only store a certain amount of historical data and any data outside this boundary will become unavailable.
- Dynamic: Resources, their properties and definitions can change dynamically in response to events. For example, a monitoring event in a refugee camp may cause several devices in the near vicinity to activate automatically.

Furthermore, for the WoT to truly flourish and be deployed in a useful context, accessing resources should be easy, intuitive and hassle-free. Take the example of a disaster event like the likelihood of a major flood along the River Thames, London. Whilst governmental bodies will employ the necessary measures to monitor this type of event and to keep track of developments (e.g. water level across areas of high risk), keeping this data confined and restricted internally will hinder public use of this critical information, which might prove fruitful if the power of crowdsourcing can be leveraged appropriately and responsibly. If the information was instead exposed to the general public in a controlled manner, hobbyists and enthusiasts could easily conjure intelligent agents that monitor key events and push alerts or compose mashups to not only aid in the awareness of the disaster situation, but to also prepare a response in a timely manner. Even more-so, the publicly exposed data might be used for other purposes, for example to monitor environmental changes in neighbouring areas or for composing other useful mashups. But this can only really become possible (both in terms of exposing data as public resources and consuming the resources by the general public) if the mechanisms behind doing so are intuitive, flexible and speedy. If the governmental body has to setup a horde of accounts and roles and if the public agents have to register accounts to publish or use this data, then the likelihood of its adoption and the usefulness of its exposure will quickly deteriorate due to the expensive investment in time. Instead, if all this access control information could be stored in a few well designed tokens, and then these tokens distributed to those with a need to consume the data without requiring them to register an account, then it

can be seen that the effort is more likely to be rewarded with higher adoption and consumption. It is with this reasoning that this study has opted to develop an enhanced model of the token-based access control mechanism to control and audit access to temporal and dynamic resources within the framework.

3.4.3.2 How does the TBAC scheme work?

TBAC systems are based on the premise of reusable and reconfigurable tokens that grant access to a set or group of resources for a particular user [73]. After generation, they are transmitted to agents who need to consume a set of private resources that are normally hidden from public view and accessible only by the resource owner. Tokens can be configured to only expose the relevant resources and assets without leaking any information regarding the identity of the resource owner. This is advantageous over UBAC which requires the identity of the user to be transmitted with a request. Whereas roles in RBAC are a part of the overall organizational structure and are therefore more permanent and long-term artifacts, tokens in TBAC are much more decoupled and can be easily generated, modified and revoked without affecting the organizational structure. This provides a significant managerial advantage when tokens are used to control access to temporal assets of the network. Finally, since tokens are tied to resources as opposed to users who own those resources, this scheme provides a resource-centric access control scheme which is perfect for managing interactions with resources in an enterprise-grade WoT setting.

3.4.3.3 CPPM-TBAC

TBAC, as opposed to UBAC and RBAC, provides a decoupled resource-centric mechanism of access control which is capable of scaling efficiently with the dynamic environment of temporal assets in the WoT. It suffers, in part, from lower security because at its core, TBAC offers a single-step authentication service (i.e. the presence of a token is sufficient to access a service). In contrast, the other schemes generally require two-step authentication which increases security. This study is not addressing the security concerns of TBAC, so the problem then is one of modeling the necessary policy systems that allow tokens to be used efficiently in the face of big data in the WoT, otherwise the advantages gained through resource-centricity will quickly be lost against the volume of tokens needed to model access controls for volatile and highly unpredictable temporal assets. Current literature in TBAC models is scarce to say the least and it is hard to find any relevant publications which discuss TBAC in a WoT setting, let alone any enhancements on top of the scheme. This study proposes the CPPM as an effective and comprehensive modeling scheme that enables tokens to be used in a large WoT setting without incurring costs in terms of generating large volume of tokens, extensive maintenance and un-intuitive usage thereof.

The CPPM-TBAC works over the asset model for SAW which represents resources at different levels of granularity and expressiveness, as has been presented earlier. By utilizing a RESTful API, resources are first exposed as web-accessible URIs which can be interacted with using the 4 common HTTP verbs: POST for creating, GET for querying, PUT for updating and

DELETE for removing resources. CPPM-TBAC controls access to resources in this asset model at the various levels of granularity; starting from the most verbose, expressive and comprehensive DF right down to the least expressive and cardinal DP.

A set of tokens are generated automatically for each DF to represent a common set of read and write permissions and further tokens can be generated by users for refining access to DF and DS.

Tokens effectively enable the modelling of multi-faceted and cascading sets of permissions for accessing resources on the network. In SAW's implementation of TBAC (the CPPM), the 1st step is to define two top-level visibility controls for resources:

1. Public access: These resources can be searched and viewed by everyone and do not require a token.
2. Private access: These resources can only be accessed if a token with the necessary permissions is used. Child resources of a private visibility resource are always private.

The 2nd step is then to categorise actions as either:

- Read actions: Identified by the GET HTTP verb, these actions view resource information. A public resource can be read freely whereas a token with the necessary permissions will be required for reading a private resource.
- Modify/write actions: Any action that uses the remaining HTTP verbs (PUT, POST, and DELETE) has the potential to modify resources on the network. Regardless of the visibility of a resource, a token with the necessary permissions is required to carry out these actions.

The general process for creating tokens is shown in Figure 3-20. In the beginning there is the option of restricting the token scope to particular DF for a given user (and subsequently, selected DS with those DF). In the next step, actions that are permitted on the selected resources can be chosen and finally, due to the extensible nature of SAW's architecture, additional restrictions can be defined to refine the scope of the token even further by adding context-specific constraints (e.g. location) or usage limits (e.g. max requests per defined threshold). Furthermore, each token can have multiple sets of permissions in a cascading fashion thanks to CPPM which enables more fine-grained access control for network resources. Finally, the tokens can be used to audit resource access as each request is logged. This TBAC model presents a comprehensive and extensible access control mechanism for a WoT network's temporal resource-based asset model and allows users to easily provision and audit access to private resources.

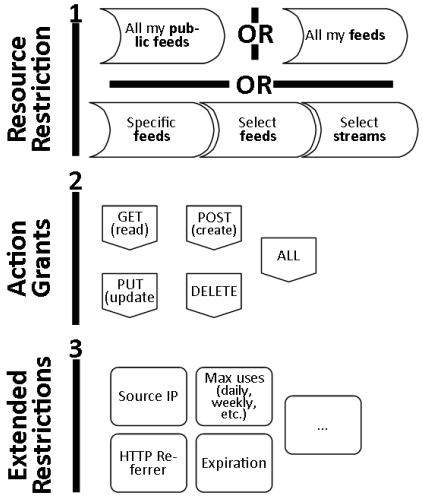


Figure 3-20: CPPM-TBAC model showing token construction process

CPPM defines two upper-level scopes when forming the tokens: (1) *Global scope* and (2) *Local scope*. The global scope can contain the basic grants (CRUD operations, i.e. create, read, update and delete) and the extended

access restrictions whereas the local scope can only specify the basic grants, but can do so for any group of resources. Permissions defined in the global scope cascade to all public and private resources of the resource owner. The local scope can then be used to refine these permissions further if needed, or to remove certain resources from the permission set altogether, as shown in Figure 3-21. The eventual applied grants are calculated according to the following methodology:

1. If global grants are present and local grants are absent then apply the global grants on all public and private resources for the resource owner.
2. If local grants are present and global grants are absent then apply the local grants on the specified resources for the resource owner.
3. If both global and local grants are present, then do the following:
    a. Apply the global grants on all public and private resources of the resource owner;
    b. For the DF and DS specified in local grants:
        i. Keep the global grants which have not been specified in the local scope.
        ii. Apply the local grants which have not been specified in the global scope.
        iii. Overwrite the global grants which exist in the local scope with the local scope grants.

This methodology is only applied on the basic grants and not on the extended access restrictions which are always defined in the global scope and cannot be overwritten locally. This is an area where the CPPM can be

improved in future iterations.

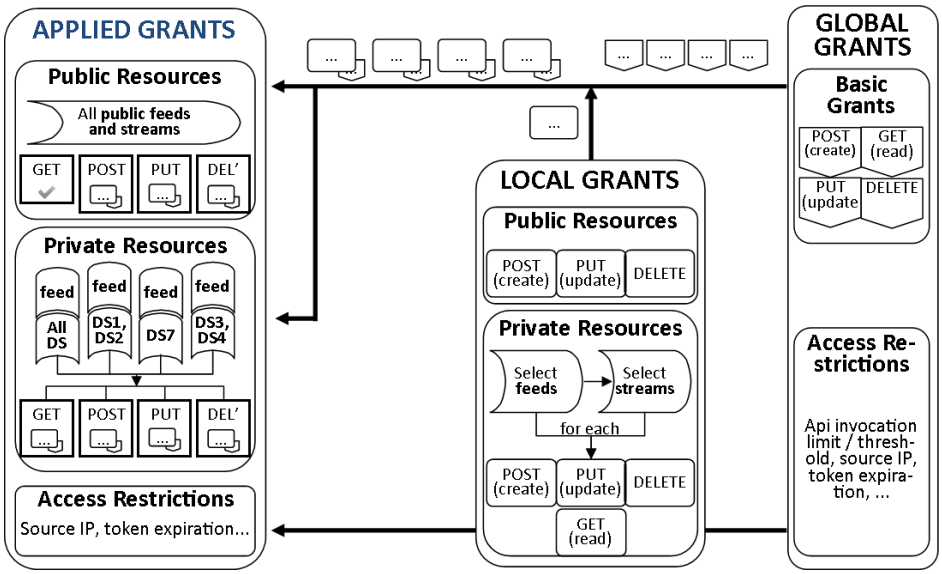


Figure 3-21: Pictorial illustration of the CPPM Algorithm

In the global scope, the basic grants consist of the CRUD operations and any or all of these can be defined with a value of 1 (grant) or 0 (restrict). CPPM employs the *least access* methodology so that the absence of a grant is equal to its restriction. Usually, it is discouraged to define global grants because they apply to all the resources of the resource owner and if the relevant local scopes are missing, they can result in the unwanted exposure of sensitive resources or the unintentional cloaking of others.

The local scope is used to refine access restrictions to resources on a finer level. Here, it is possible to specify grants and restrictions for a group of resources based on their visibility (e.g. “public/private” for the respective public/private-visibility resources, or “all” for all resources). Further extensions to the CPPM may permit other types of resource groupings as well in the future. The local scope also makes it possible to define access controls for specific resources denoted by a resource ID, which will be a DF ID at the topmost level. Going even further, the CPPM adheres to the asset

model presented earlier and allows refining of access down to the individual DS, again, either by their visibility group keyword or by specific DS IDs. This cascading permissions style allows CPPM to easily create tokens with any level of access control for any type of resource in the asset model.

It should also be noted that the GET (read) grant is not specified for public resources in the local grant scope. This is because public resources do not require a token to be queried and read, so the GET grant is meaningless in this context since it will always resolve to 1 (grant).

3.4.3.4 CPPM-TBAC Deployment in Disaster Management: Example Scenario

CPPM-TBAC can be used to model access policies for sensing devices and data in a DM situation in cases of pre, intra and post disaster. The scenario below considers CPPM-TBAC usage pre and during the disaster.

Take, for example, a flood occurring in a location known for flooding and therefore having existing sensing infrastructure to monitor the appropriate environmental variables. Pre-disaster, CPPM-TBAC can be used to manage access policies of the sensing infrastructure for known parties (e.g. governmental organisations). This can include controlling which aspects of each device can be managed by whom, and who can see the sensing data that is being collected. These access policies will typically be long-term and not change as frequently. As the disaster is unfolding, CPPM-TBAC can similarly be used to expand the range of access policies and create temporal access tokens to give new actors on the ground the relevant access to help and facilitate them in their disaster management and relief work. This can include giving first responders and relief agencies limited and short-term access to consume sensing devices data on the fly and revoking this access

as soon as their work is finished. It can also include opening up the sensing devices data to the larger public so that hobbyists can conjure up their own data-driven applications to monitor the scene on the ground and provide their communities with tailored updates. It can be seen that CPPM-TBAC can be used for a wide range of access control activities to effectively manage distribution of access to sensing devices in what is likely to be a chaotic DM environment.

3.4.4 Interaction Models

Composition of *interactions* makes possible semi-automatic processing, enrichment and publication of network assets to other agents for consumption, whether internally or externally. Generally speaking, interactions in the WoT domain come in one of the following flavours:

- Eventing systems that publish information in response to events based on pre-defined triggers and/or time-based schedules.
- PubSub systems that enable subscription to assets and the publishing of these assets thereof according to some predefined rules and/or criteria to designated agents for consumption.
- Profiling systems that work to achieve an enrichment and semantic betterment of data.

To narrow the scope of possible work and finish it within the allocated timeframe, this study only focuses on the latter interaction, that of profiling network assets in order to provide semantic metadata that can be used to enable cross-vendor, multi-party collaboration and achieve semantic interoperability in the WoT. The following section details the novel

mechanisms developed in SAW to semantically annotate sensing devices and data using the SSN ontology.

3.4.5 Semantic Annotation

The process of profiling DF and their respective DS involves the semantic annotation of the device properties, attributes, characteristics and related metadata. This is done by applying semantic concepts to the device definitions and data and then storing these concepts in the form of semantic metadata which can be shared and distributed to achieve an interoperable representation of data and devices across the multi-party collaboration framework. Over time, the framework will become more capable of automatically annotating devices semantically when they are connected to the network as more and more semantic annotations are added to the framework. This will make it easier for the system to recognise common devices and offer suggestions during the profiling phase, thereby quickening the annotation process and improving its accuracy.

Assets in the SAW framework are annotated using the SSN and SAW ontologies as has been illustrated before. The maximum benefits are reaped when users are highly expressive and precise during the semantic annotation process. Whilst this is ideal, it is not practical to assume that regular non-tech users will be capable of effectively annotating their resources, which means that direct serving of semantic templates to users is not the best solution. To account for this practical limitation, SAW offers a set of supplementary approaches that should help annotate resources even when the user cannot do so directly and on the fly.

When a DF or DS is registered to the network, it may be profiled in one of two ways: (1) Directly, and (2) Indirectly.

3.4.5.1 Direct Semantic Annotation via Tags

The direct annotation method requires explicit input from the user and it is illustrated in Figure 3-22. The direct method relies on the actual user to power and drive the semantic annotation process, and its accuracy and effectiveness depends on the richness of the semantic knowledgebase. As more devices are registered and annotated, the accuracy and efficiency of the system will increase, ultimately enabling semi-automatic semantic annotation of assets.

When a new device (DF or DS) is being registered with the network, a set of tags will need to be provided to begin the semantic profiling mechanism. The mechanism takes into account the three possible variations in this case:

1. Scenario 1: The provided tags map to one or more semantic concepts and the user has selected one of the provided concepts for mapping of the device. In this case, the device is mapped to the selected concept.
2. Scenario 2: The provided tags map to one or more semantic concepts but the user is not happy with the provided results and opts to create a new concept. In this case, a new concept is created for the device and the device is then mapped to this newly created concept.
3. Scenario 3: The provided tags do not map to any known semantic concept in the knowledge base. In this case, the user is given the option of creating a new semantic concept for the device and then

map the device to this new concept. Otherwise, the whole process repeats again until the user makes a choice/selection.

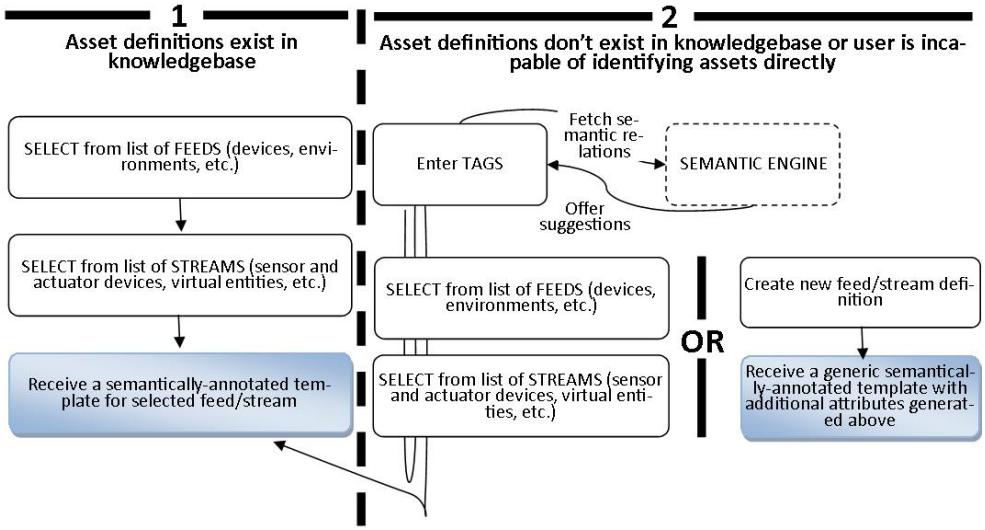


Figure 3-22: Asset profiling process illustration

Each scenario is described step by step in more detail below in Table 3-4.

In the first two scenarios, the limiting factor is the user's ability to identify the corresponding asset. In the last scenario, the limiting factor is the lack of a comprehensive knowledgebase from which to fetch the relevant semantic concepts to build the semantic template for annotation.

In the first scenario, the user is registering a device for which semantic definitions already exist in the knowledgebase. If the user is using the web interface, as is the case in this scenario, then the user can easily select the corresponding device from a drop-down suggestion list and receive the corresponding semantic template for annotation. In most cases, since the definitions for the device already exist in the knowledgebase, the user will only need to modify a small number of concepts specific to his/her device (e.g. location, observation interval, host platform, etc.) as most static properties will already be stored for that device.

Table 3-4: Possible scenarios in the semantic annotation process for profiling DF and DS

Scenario 1: Asset definitions exist in knowledgebase AND user is able to select the corresponding definitions (e.g. when using the web interface).	
Step 1:	Register DF.
Step 2:	Select the corresponding asset from the knowledgebase for that DF.
Step 3:	Fill in/modify (if required) the received semantic template for the DF and upload to server.
----------	Repeat these steps for registering DS.
Scenario 2: Asset definitions exist in knowledgebase BUT user is unable to select the corresponding definitions (e.g. when using the RESTful API).	
Step 1:	Register DF.
Step 2:	Update the DF with a list of tags identifying the device.
Step 3:	Select from suggestions provided in the response those tags which most closely match the device being registered. Repeat steps 2 to 3 based on personal discretion and then upload the final list of tags to the server.
Step 4:	The server will return a semantic template with each response in step 3. Fill in/modify (if required) the received semantic template for the DF and upload to server.
----------	Repeat these steps for registering DS.
Scenario 3: Asset definitions don't exist in knowledgebase OR user is unable to select the corresponding definitions (e.g. by not specifying the correct tags).	
Step 1:	Register DF.
Step 2:	Update the DF with a list of tags identifying the device. If no tags are provided, the system will still proceed to the next step as is the case in this scenario.
Step 4:	The server will return a generic semantic template if the list of tags submitted is either empty, or does not match any stored semantic concepts. Fill in the received semantic template as much as possible for the DF and upload to server.
----------	Repeat these steps for registering DS.

In the second scenario, the user is registering a device for which the semantic definitions already exist in the knowledgebase, but the user cannot directly select the corresponding device because of interface limitations (e.g. when the user is interacting with the network through the RESTful API). In this case, the user will need to send a list of tags identifying the device in a separate payload after the device has been registered. Once the server receives these tags, it tries to retrieve matching concepts from the

knowledgebase, the process for which is detailed further on in this subsection. At this juncture, the server generates a semantic template with semantic concepts most closely matching the provided tags, and then sends this semantic template along with a short list of tag suggestions to the user. The user, in turn, can either fill in the received semantic template if it closely matches the device in question, or can provide further tags including those provided by the server to refine the process further until he/she is happy with the received semantic template. The template can then be filled in and sent to the server.

The third scenario is similar to the first scenario. In this scenario, the user is registering a device for which one of the following is true:

a. The definition for that device does not exist fully in the knowledgebase;
b. The user is not providing a list of tags, or the list of tags provided is incomprehensive.

In this case, the server will return a generic semantic template for the user to be completed in the response payload.

To sum up, when profiling DF or DS, the users will be able to enter arbitrary tags to describe their assets. These tags are not semantically-restricted and can be anything the user wants them to be, for example, “light sensor”, “Oracle”, “Arduino”. The system might try to infer semantic meaning from the tags as they are entered by the user and offer further suggestions for tags. Furthermore, as tags are entered by the user and their semantic relation is recognised by the system, SAW will be able to provide tailored DF and DS templates to the users dynamically, thereby improving the accuracy of the

semantic annotation process. For example, the user may enter the tags: “light sensor” and “SunSPOT”. From this information, the system may be able to work out that the “SunSPOT” is a multi-sensor device and “light sensor” is possibly a sensor object that measures photons. Equipped with this information, the system will be able to offer the user a tailored DF template for a SunSPOT device and a DS template for a generic light sensor. The actual tag-based semantic annotation mechanism is illustrated in Figure 3-23. In the first step, DF (multi-sensor platforms) and DS (sensing devices) are registered to the network with an arbitrary payload where only the asset name, visibility and associated tags are required. Once this payload has been submitted (labels 1.0 and 1.1), a query builder is used to parse the specified tags and generate synonyms, possibly using open tools like WordNet [94].

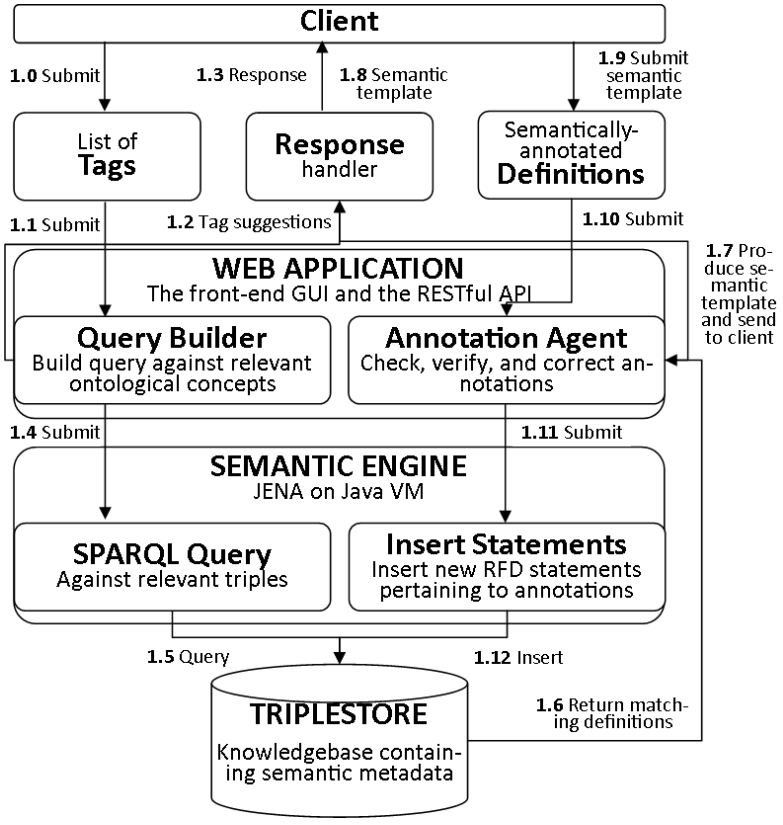


Figure 3-23: Asset profiling scheme showing how tags are used to derive semantic definitions

SAW tries to generate an exhaustive list as false positives are not a major issue since the aim is to give the end user a comprehensive list of corresponding asset definitions from which the correct or most relevant artefacts can be selected. After building the augmented list, the query builder calls the SPARQL query agent which runs a semantic query against the knowledgebase to find semantic concepts relating to the specified keywords. Since the internal knowledgebase will be limited in the beginning, manual configuration may be required to bootstrap the system. As more and more semantic annotations are added to the framework, however, it will become easier for the system to recognise common assets and offer suggestions in the feedback loop (labels 1.2 and 1.3) during the profiling phase, providing the client supports the feedback mechanism.

In the second step, matching semantic definitions are returned to an annotation agent where a semantic template is generated and sent to the client for annotation (labels 1.6 to 1.8). If the client does not support a feedback mechanism, then the system will self-annotate the template based on the available information, as might be the case when mining data from external repositories like Xively and ThingSpeak.

Finally, the client submits the annotated semantic template to the system and the annotation agent forwards the response to the semantic engine where semantic metadata in the form of RDF statements (or triples) are inserted into the knowledgebase.

The above procedure can result in one of 3 cases, as described previously:

1. The client is using the web GUI and can identify the relevant assets directly. In this case, there is no need to provide tags.

2. The client cannot directly select the relevant assets or the client is communicating via the API. In this case, the client provides tags to describe the assets.
3. The system is unable to retrieve the relevant asset definitions from the knowledgebase. In this case, the system returns a generic semantic template leading to the creation of a new semantic concept.

3.4.5.1.1 Mapping Devices to Existing Concepts – Process Explanation

When a new device (DF or DS) is registered with the network, the requester has the option of specifying device-specific tags to aid the network in semantically annotating the device. When these tags are submitted to the SAW framework, the system tries to fetch the corresponding semantic associations for these tags from the RDF triple store (openly available database for storing semantic data). The URI for submitting these tags is: *http://localhost:8111/fetch-associations/*. The requester must then append a comma-delimited list of tags at the end of the URI (i.e. the query string), for example, *http://localhost:8111/fetch-associations/tag1, tag2, tag3*. Each tag in this list is then processed by the system to try and fetch the corresponding semantic concepts from the knowledge base. Figure 3-24 shows the web form where tags are entered at the bottom. After clicking on the “Fetch Associations” button, the “Matched Concepts” section at the top is populated with a list of corresponding concepts.

Each semantic concept has an initial list of primary and secondary tags, as has been explained before in Chapters 2 and 3. When a new device is registered to the network and a list of tags is provided, then each tag from this list is processed by the system to search for any matching concepts. The

device tags are queried against the primary and secondary tags of all concepts to derive mappings. Each mapping is then classified as a primary or secondary mapping depending on whether the device tag mapped to a primary or secondary tag of the concept. If the same concepts are retrieved multiple times through different device tags, then the weight of those concepts are increased to highlight their raised rank and matching correlation. Mappings through primary tags contribute a weight of 2 whilst mappings through secondary tags contribute an additional weight of 1.

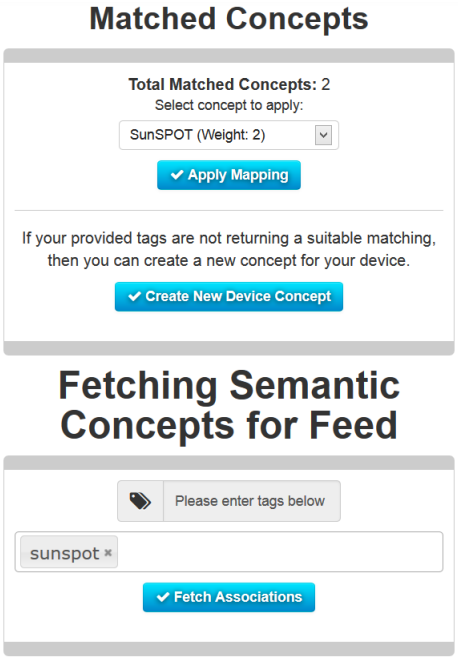


Figure 3-24: Semantic profiling screen, showing the tag mapping facility at the bottom and the selectable matching concepts at the top

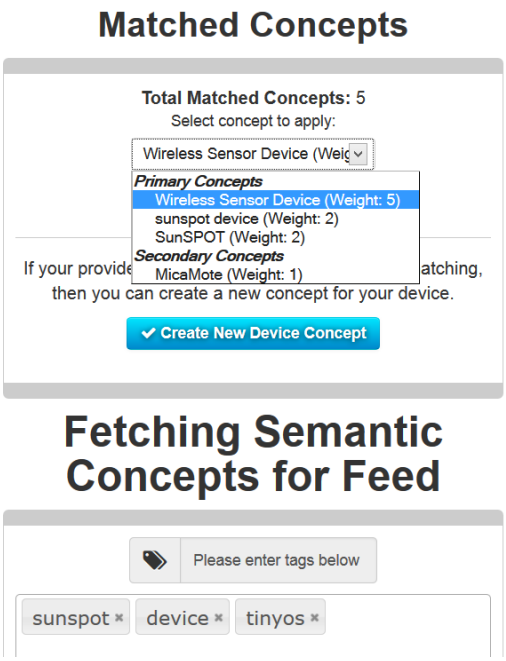


Figure 3-25: Sample annotation process showing a list of primary and secondary concepts and their respective weights

Figure 3-25 (above) shows a sample annotation where the submission of tags for a newly registered device has resulted in the system returning a set of primary and secondary concepts back to the user, each with a corresponding weight. This list may very well contain concepts that are similar or even duplicates of each other simply due to the community-driven

nature of the system whereby users are able to create new concepts on the fly.

If the user is able to interact with the SAW network, as is the case here, then he/she can select an appropriate semantic concept from the provided list and click on the “Apply Mapping” button shown in Figure 3-24. Doing so will create a new mapping for the device to the selected concept, and generate a new semantic ID for the device as well (e.g. *SunSPOT-1*). The semantic ID of the device forms a linked data URI which can be used to interact with the device in a semantic fashion. The final traversable URI of the device may look something like this: *http://saw.local/sw/feeds#SunSPOT-1* for DF or *http://saw.local/sw/streams#CO2Sensor-251* for DS. When the mapping is applied, the provided tags are classified as primary or secondary tags by matching them against the primary and secondary tags of the semantic concept the device is mapped against. A slight limitation of the SAW network here is that when mapping to existing semantic concepts, there is no way for new devices to add new primary tags to the mapped concept. The new device can only contribute secondary tags with the current mechanism. A possible solution to this would be to maintain a persistence of weights when devices are mapped to concepts, and if any secondary tags reach a certain defined threshold, then they can be promoted to primary tags. This takes care of case 1 where the provided tags map to one or more semantic concepts and the user has selected one of the provided concepts for mapping of the device.

3.4.5.1.2 Creating New Semantic Concepts for Devices

Both cases 2 and 3 presented earlier lead to the creation of a new concept. This is done by clicking on the “Create New Concept” button in Figure 3-24. When a new concept is created, a new concept ID is generated and this publishes the new concept as a linked data concept, e.g. *http://saw.local/sw/ontology#MicaMote*, where *MicaMote* is the new concept ID. Once the new concept has been created, the device is mapped to it and a unique semantic ID for the device is generated as explained before. In this case, all the device tags are classified as primary tags for the new concept.

When devices are mapped to semantic concepts, the knowledge base is enriched further and the system is able to infer and aggregate the primary and secondary tags of the devices that are mapped to the concepts to the primary and secondary tags of the concepts themselves. This results in continuous enhancement of the knowledge base and better accuracy of mapping when new devices are registered to the network. However, at the same time, a systematic and regular review of the knowledge base will be required to clean up and remove ambiguous tags such as “device”, “wireless sensor” and alike to improve the annotation accuracy and decrease the chance of generating false positives during the semantic profiling phase. At this moment in time, SAW does not implement any such semi-automatic review mechanisms and this has to be done manually. A potential area of further research is to use established resources like WordNet [95] and ResearchCyc [96] to help in the automation of removing ambiguous or erroneous tags and enhancing the existing knowledge base by enriching tags with synonyms and semantically-similar concepts.

3.4.5.1.3 Challenges in Creating a New Semantic Concept

In the case of a new device being registered to the network for which no relevant semantic concept exists in the knowledgebase, the system creates a new concept for this device. The challenges in this task are the following:

- Generating a unique concept ID so that the concept can be represented in a unique fashion without conflicting with existing concepts. For example, if the concept “saw-ont:Arduino” exists already, the new concept cannot also be called “saw-ont:Arduino” since that ID already exists.
- Aligning the new concept with existing concepts that may be similar, or at times, aliases. The biggest challenge here is the removal of duplicates when a concept has syntactical differences in representation and the list of tags attributed to it.

Generation of a unique concept ID is easily resolved by the system. The system first searches the triple store for existing concepts with the same ID as the new concept that is to be inserted. If a match is not found, then the new concept can be added with the ID originally provided. If, however, a match exists, this implies that the system needs to generate a new unique ID for the new concept. The system is provided with a list of pre-composed methods by which it can achieve this, for example, by adding a date-time value at the end of the concept ID (e.g. “saw-ont:Arduino-2013-11-14”), amongst other ways. Eventually, the system will be able to derive a unique concept ID that can be used for the new concept, and upon insertion, it will return this unique concept ID back to the requester so that the URI for the new concept is known.

The problem of semantic aligning of semantic concepts and removal of duplicates is more challenging and involves manual intervention. Due to the intrinsic community-driven nature of SAW, the likelihood of generating duplicate concepts remains very high. The scope of the current study does not permit extensive research into this area, but one way in which this problem can be effectively tackled is by using the *owl:sameAs* property to link concepts together as aliases of one another. Again, in the current setup this has to be done manually but future enhancements of the framework can investigate more effective and semi-autonomous approaches in this regard.

3.4.5.2 Indirect, Community-Based Semantic Annotation

By utilising a community-driven contribution system, SAW envisions a comprehensive and peer-to-peer community tagging system that is built and driven by members of the system. While this feature is discussed in this study, limitations in time and resources, unfortunately, did not permit its implementation in the first prototype of SAW. It is hoped that this feature will be incorporated in a later iteration of the framework.

SAW is intrinsically a community-oriented solution where the focus is on collective knowledge and collaboration. By employing a community-driven contribution system, SAW can make it even easier and viable for users to flag incorrect annotations and contribute relevant tags and semantic annotations for DF and DS. This system essentially consists of the following:

a. Ability to flag incorrect annotations. For example, if a SunSPOT device (a multi-sensor platform which is a DF) has been annotated as a temperature sensor (a sensor object which is a DS belonging to a DF).

b. Ability to contribute relevant semantic annotations. For example, a SunSPOT device may only be annotated by its owner as a “sensor platform”. Other community members with more technical knowledge might add further annotations like “has sensor”, “has analogue inputs”, “has digital outputs”, etc. These public annotations are added to the relevant DF and DS by default, but the owners of the respective assets can flag incorrect public annotations for review by the instance administrators who will have the power to remove irrelevant annotations.

As mentioned previously, limitations in time and resources does not permit further study and analysis of this community-driven approach to semantic tagging but the concept was introduced briefly nonetheless to highlight possible areas of further work and improvements to the SAW framework.

Chapter 4: Implementation of the SAW Prototype

4.1 The Cloud-Based SAW Framework

Taking the notions of a *distributed collaboration framework* and *semantics data modelling* forward, the top-level concept architecture for SAW is derived as illustrated in Figure 4-1. An instance of SAW exists in the cloud where the supporting computing resources (e.g. CPUs, RAM, storage and bandwidth) can be easily scaled up and down depending on the demand of the network. The system itself consists of 3 distinct components:

1. Semantics Engine: Enabled by the open-source JENA implementation by Apache and running on a Java Virtual Machine (VM), the semantics engine deals with the semantic annotation of resources on the network as well as semantic reasoning and querying of assets. The semantics engine is exposed to the web through a Tomcat servlet provided by Apache. The actual semantic metadata browser is called a Fuseki server.
2. Webserver: The front-end web application is hosted on an Apache webserver and exposes the underlying functionalities through a RESTful API. Amongst other things, the front-end application (i.e. the web application or the website), deals with the following:
    a. Implementing (and exposing) the underlying semantic engine to the web for applications such as semantic querying of network assets;
    b. Providing a UI for the web application users;
    c. Providing a web-based administration client for the instance administrators;
    d. Exposing the framework functionalities through a RESTful API.

3. Real-time server: Powered by Node.js, this acts much like the webserver above but has a few additional key functionalities that enable and make possible real-time monitoring and analysis of the network as well as real-time capture and publication of data, information and knowledge.

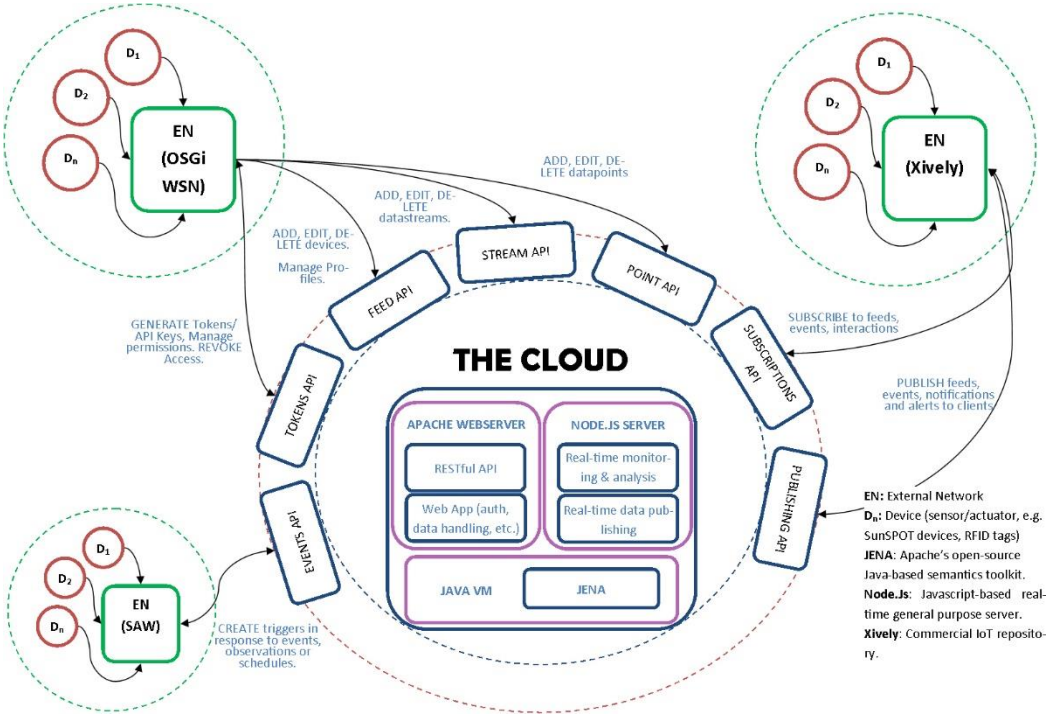


Figure 4-1: SAW - The concept of an extensible system that exposes underlying functionality through open APIs

The semantics-based modelling of assets and the distributed SoA-based design of the system enables SAW to easily communicate with and collaborate amongst not only other instances of itself, but also other commercial and public IoT solutions like Xively and Thingspeak (with the help of adapters). By extensively focusing on the problem of collaboration and tasking itself with the design and creation of a decentralised, RESTful and semantics-enabled system, SAW has the potential to offer and enable plug-and-play collaboration amongst WoT applications.

4.2 The OSGi-Based Wireless Sensor Network

Whilst developing the SAW framework prototype, it became apparent that a local WSN would also need to be built to test and evaluate SAW's asset model, the effect of introducing CPPM-TBAC into the equation, and to measure the accuracy and effectiveness of the semantic annotation process. The WSN, at the same, needed to be able to host, cope with and interact amongst various heterogeneous sensor platforms and devices, so an interoperable solution was required to build the test bed.

Eminent issues relating to device heterogeneity, vendor lock-in and platform dependencies can be resolved by using an OSGi (Open Service Gateway initiative) framework as the software fabric for IoT deployment [97], and in our case, the local WSN. The OSGi standard is essentially a service-oriented component model and enables high modularity and portability of the codebase and improved resource utilisation [98]. Managed software components deployed in the OSGi platform are called "bundles" which can be installed, updated, or removed on the fly without disrupting the operation of the host device. These bundles can also dynamically discover and interact with other OSGi bundles/services, thereby breeding an ecosystem of modular, independent and self-contained functionalities that can be adopted and extended with ease.

Executing on a networked device such as gateway, OSGi service platform is capable of managing the life cycle of the software components in the system. The provided management functions allow the dynamic installation, update, and removal of the software components without disrupting the operation of related devices. Software components in OSGi can dynamically lookup and

use other components, and even integrate with other OSGi-based components into an application or library.

The OSGi-based Sensor Gateway Node (OSGi-SGN) developed in this study interacts with a WSN and the cloud-based SAW framework. The Equinox implementation has been chosen as it is the most common and established implementation and therefore conveys high code reusability and extensibility value. Other well-known implementations include Apache Felix and Makewave Knopflerfish.

The architecture of OSGi-SGN is shown in Figure 4-2. The first three layers of this architecture (Hardware, Operating System and Java VM) are the underlying parts of this gateway and must primarily meet minimum requirements for running OSGi. The fourth layer consists of the OSGi framework and contains several components, each of which is an OSGi bundle. These bundles can communicate with one another based on the service-oriented approach and depending on the task specification provided during runtime.

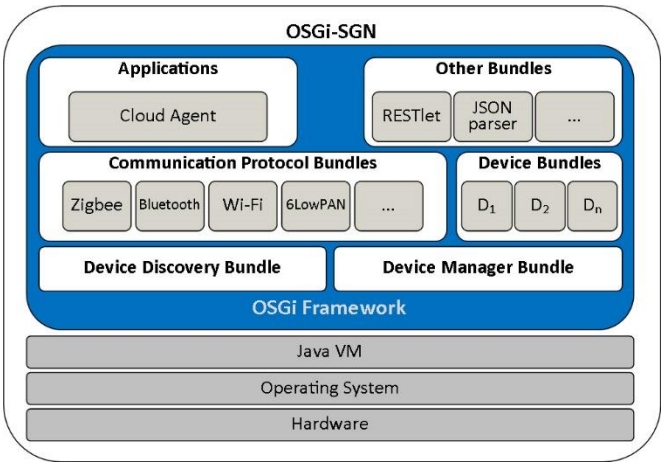


Figure 4-2: OSGi-SGN architecture

The OSGi-SGN consists of the following bundles:

1. Device Discovery Bundle: Discover new devices which have recently been added to the network, collect information about these devices (device type, communication protocol, address and etc.) and store it for future use.
2. Device Manager Bundle: Since there are a lot of possible combinations of devices and communication protocols, there is a need for a bundle which will provide a unified and abstract interface for communication between these devices and the gateway. Device Manager Bundle is responsible for direct communication with all the devices and it can control the devices, monitor their status and enable cooperation with other components (such as to receive service requests, report device status, etc.).
3. Device Bundles: In this architecture, the devices are divided into two main categories: sensors and actuators. Actuators are “active” and can be controlled to serve users, whereas the sensors are “passive” and can only be used to collect data. Sensing devices may also be considered as “semi-passive” since some devices allow tweaking of parameters and observational properties.
4. Communication Protocol Bundles: Different devices might have different means of communication with the gateway. In order to be able to get data from all these devices, the gateway has to support at least the most common communication protocols such as Wi-Fi, Zigbee and Bluetooth. More bundles can be added later on to increase the range of communication protocols available for interaction with devices.

5. Other Bundles: Additional bundles built for data processing and transformation requirements within the network and for interacting with the SAW framework.

With this architecture, it becomes possible for the OSGi-SGN to integrate a wealth of heterogeneous devices and act as a test bed for the evaluation and analysis of the SAW framework.

Chapter 5: Simulation of Framework and Results

5.1 Overview

The SAW framework consists of several elements that are essential for the correct working of the system but these may affect the overall efficiency and system performance. This section presents a performance evaluation of these elements. Namely, the elements to be evaluated are the following:

- CPPM-TBAC Mechanism;
- Semantic Annotation Mechanism.

5.1.1 Simulation Setup

The simulation setup is the same for the performance evaluation of both elements and is shown in Figure 5-1.

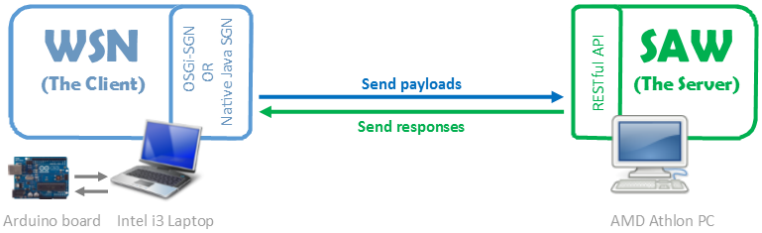


Figure 5-1: SAW simulation setup

The simulation setup consists of a client acting as the Wireless Sensor Network (WSN), and a server running the SAW framework. The client is an Intel i3 2.4GHz laptop with 8GB of memory running Microsoft Windows 8 and the server an AMD Athlon processor PC. An IP-enabled Arduino-based multi-sensor platform with two sensors is attached to the client. For all scenarios except one, the client acts as the OSGi-based Sensor Gateway Node (OSGi-SGN). For one scenario, the client acts as a Native Java-SGN (OSGi framework is not used). This is explained further in the appropriate sections below.

The OSGi/Native Java-SGN communicates with the SAW network through a RESTful API. The function of the client is to collect sensor data and readings from the attached multi-sensor platform, and then package these as payloads that can be sent to the server for processing. The function of the server is to receive the submitted payloads, process them, and then return a response to the client.

All requests considered in the simulation scenarios below originate from the SGN when it wants to register a new DF/DS, update an existing DF/DS, or upload new sensor readings/DP to the SAW network. The server then processes these requests and returns a response to the client.

5.1.2 Definition of Simulation Scenarios

5.1.2.1 CPPM-TBAC Mechanism

For the CPPM-TBAC mechanism, the parameters to be measured and evaluated are the *response time* and the *added delay*.

The response time is defined as the time taken for a response to be received from the server after a request has been submitted by the client. In each scenario, this parameter is measured both when CPPM-TBAC is turned on (the *higher response time*) and also when it is turned off (the *lower response time*). The difference between the response time when CPPM-TBAC is turned off and when it is turned on is referred to as the added delay.

$$Added\ delay = Higher\ response\ time - Lower\ response\ time$$

The added delay can also be turned into a percentage by dividing the higher response time with the lower response time and taking away 1 and this is referred to as the *percentage added delay*.

$$Percentage\ added\ delay = \left(\left(\frac{Higher\ response\ time}{Lower\ response\ time}\right) - 1\right) \times 100$$

The response time is measured primarily to calculate, compare and analyse the added delay and percentage added delay parameters. The added delay and percentage added delay parameters are being calculated to compare the difference in response times when the CPPM-TBAC mechanism is turned on in order to determine its effect on the SAW network's ability to scale and handle large amounts of data.

Section 5.2 presents a simulation where the performance of the OSGi-SGN is compared to a Native Java-SGN to analyse the various tradeoffs for using OSGi as the software fabric for the SGN.

Sections 5.3 and 5.4 then analyse the CPPM-TBAC mechanism performance with non-aggregated and aggregated payloads respectively. Non-aggregated payloads refer to the scenarios where sensing device definitions or data are submitted from the client to the server one by one. On the other hand, aggregated payloads refer to the scenarios where several payloads are combined by the client to form one aggregated payload. This aggregated payload is then submitted by the client to the server in one go. In section 5.3, the response times are also measured and compared for different payload sizes in order to determine if this has any adverse effect on the SAW network.

Finally, section 5.5 summarises the results of the CPPM-TBAC performance evaluation and presents some final analysis on the proposed mechanism.

5.1.2.2 Semantic Annotation Mechanism

For the semantic annotation mechanism, simulations are carried out to test the suitability and effectiveness of the semantic profiling mechanisms when annotating a set of sensing devices semantically. The objective is to determine the accuracy of the semantic annotation process and the ability of the system to learn from these annotations and augment the internal knowledge base.

The semantic profiling of network resources involves the semantic mapping of DF and DS to their corresponding semantic concepts in the semantic data store. A semantic concept is a class in an ontology that represents an idea, feature/property, or object. Examples of semantic concepts include an RDF class to represent a temperature sensor, or an RDF class to represent a multi-sensor platform. The process of semantic annotation transforms the schema-oriented and restricted network resources into schema-less and open assets that can be browsed, navigated and interacted with by external agents (both human and machines).

The simulations consist of a list of 50,100 and 500 devices (depending on the scenario set) with a pre-configured list of tags for each device. The devices map to 10, 20 or 50 possible concepts, again depending on the individual scenario set configuration. The basic 10 concepts used in the simulations are as follows: Arduino, SunSPOT, MicaMote, TelosMote, EpicMote, WaspMote, MicrochipPIC, DragonBall, AtmelAVR and RFID USB Reader. Each concept has a list of 2, 5, 10 or 50 possible devices depending on the scenario set and these are named with the name of the concept and a

number 1-n, for example: *SunSPOT-5*. The concepts to devices ratio is worked out by dividing the number of devices with the number of concepts.

$$Concepts\ to\ devices\ ratio = \frac{Number\ of\ devices}{Number\ of\ concepts}$$

For a simulation scenario set consisting of 50 concepts and 10 devices, the concepts to devices ratio will be expressed as 1:5.

Each concept has a list of tags and these tags are submitted with the devices during the profiling phase in a random manner. The list of tags for each concept is provided in

Table 5-1 below.

Table 5-1: List of tags being used in each of the simulation scenarios for each semantic concept.

Semantic Concept	**List of Tags**
Arduino	Arduino, Arduino Board, Arduino Shield, Uno, Leonardo, Due, Micro, Lillypad, Nano, Fio
SunSPOT	SunSPOT, Sun, SPOT, Oracle, Rev8
MicaMote	MicaMote, Mote, ATmega, TinyOS
TelosMote	TelosMote, Mote, TelosB, UC Berkeley, Willow, Crossbow
EpicMote	EpicMote, Mote, UC Berkeley, Breakout, Devboard, Irene Base, RUC Mote, HydroWatch, Quanto, Lynx, OpenMote, Nova, Texas Instruments
WaspMote	WaspMote, Mote, Libelium, Zigbee, Wi-Fi, RFID, Bluetooth
MicrochipPIC	PIC, Microchip, Microcontroller, PIC16, PIC17, PIC18, PIC24, PIC32
DragonBall	DragonBall, MC68328, Motorola, Freescale Semiconductor, DragonBall EZ, MC68EZ328, DragonBall VZ, MC68VZ328, DragonBall MX, i.MX, MC9328MX, MCIMX
AtmelAVR	AtmelAVR, Microcontroller, tinyAVR, megaAVR, XMEGA,

	FPSLIC, RISC, Raven Wireless Kit
RFID USB Reader	RFID USB Reader, Sparkfun, RFID, RFID Tag, RFID Label, RFID Button, ID-3LA, ID-12LA, ID-20LA

In simulation scenarios where there is a need to create more concepts than the 10 shown above, the above 10 concepts are replicated one by one until the number of concepts reaches the required number. The replicated concepts and their tags have unique numbers appended to ensure that all concepts are unique.

In each simulation scenario set, the following scenarios are simulated:

1. All devices are submitted in a random fashion and provided with one random tag from the corresponding concepts.
2. All devices submitted in a random fashion and provided with two random tags from the corresponding concepts.
3. All devices submitted in a random fashion and provided with three random tags from the corresponding concepts.

In the semantic annotation process, various parameters need to be measured and analysed to evaluate the performance of the framework.

In each simulation, the following parameters are recorded:

1. The total number of concepts generated;
2. The total number of concepts which are duplicates. This will reveal how big of a problem the duplicate generation of concepts is in the framework.
3. The total number of concepts which have 2/5/10 devices mapped to them (depends on the concepts to devices ratio in the particular

scenario). This will help to determine if the system is effectively mapping devices to the corresponding concepts or not.

4. The total number of concepts which have less than 2/5/10 devices mapped to them. This stats will help to explain trends observed in the other stats (e.g. highlighting concepts with only 1/x mapped devices or 2/x mapped devices, etc., where x represents the ideal number of devices that should have been mapped to the concept (see parameter 3)).
5. The total number of concepts which have more than 2/5/10 devices mapped to them. This will help determine the rate of generating false positives.
6. The average number of concepts which are returned by the system when the tags are submitted. This is calculated by dividing the total number of returned concepts for all of the cases by the total number of cases, where number of cases is number of devices being mapped in the simulation scenario;
    - Of these, the percentage which are primary concepts and the percentage which are secondary concepts.

$$Average\ number\ of\ concepts = \frac{Total\ concepts\ returned\ for\ all\ cases}{Total\ cases}$$

7. The number of cases where one concept has a bigger weight than the rest of the returned concepts. This helps to measure the ability of the system to differentiate between different concepts in terms of their mapping suitability and relevancy to the device that is being mapped.
8. The number of cases where all returned concepts have equal weights. This is the inverse of parameter 7.

5.2 Performance of OSGi-SGN vs Native Java-SGN

Both the OSGi-SGN and the Native Java-SGN use similar setups. The only difference is that the OSGi-SGN setup has an OSGi application communicating with the Arduino board and the SAW network on the client and the Native Java-SGN implementation has a Java Web Service application on Tomcat 7 (open-source Java HTTP web server environment) instead.

This simulation is carried out for two different operations: (i) Registering a new device (DF) with 2 sensors (DS), and (ii) Updating definitions of existing DS.

Amongst the many functions of the SGN, one key function is to check whether the source device is a new device in the network. If it is, then the SGN shall register it with the SAW network by sending an initial payload describing the device (the *DF*), which is the Arduino board in this case. Upon successful registration of the DF, the gateway also registers two *DS*, one for each sensor on the Arduino board. Once the DF and DS are registered, the gateway keeps on submitting sensor readings every 20 seconds, thereby simulating a typical sensor device in a volatile application scenario like DM where devices frequently (and dynamically) incur changes in their status and properties. This is the first operation and the response times are only collected for the process of registering the DF and 2 DS belonging to it.

In the second operation, the definition of an existing DS is updated. In both these operations, the simulation scenario does not consider the semantic annotation of the registered DF and DS as this will be analysed separately in the following sections. The results of the simulation are shown in Figure 5-2.

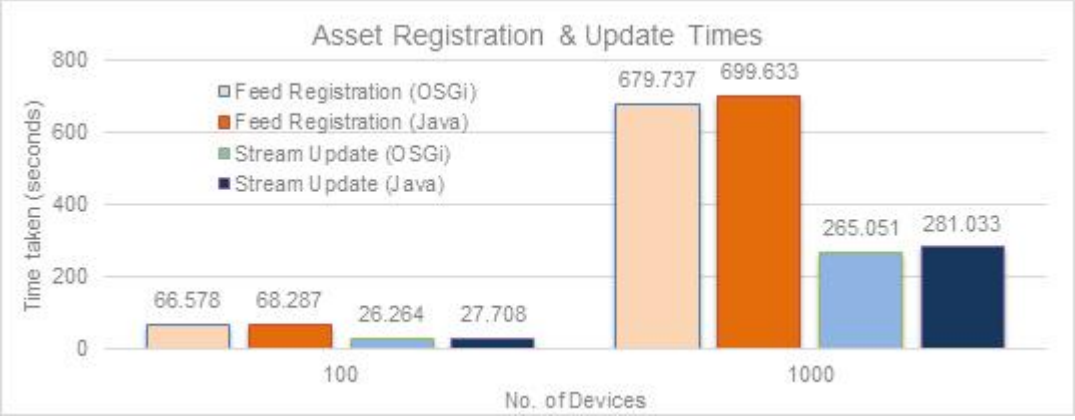


Figure 5-2: Comparison of DF/DS registration and DS update times from OSGi and Native Java-SGN

Table 5-2 shows the average response times (and the percentage added delays in brackets) for registering and updating definitions for 1,000 assets through the OSGi and the Native Java-SGN. The OSGi-SGN implementation fares marginally better than the Native Java-SGN in performing similar requests. While the performance of OSGi in this scenario is only slightly better, the real benefits are gained in the actual codebase in terms of code reusability, modularity and interoperability.

Table 5-2: DF/DS registration and DS update times for 1,000 DF/DS

	OSGi-SGN	**Native Java-SGN**
Registration time for 1K DF/DS	680s	699s (2.9% slower)
Definition update time for 1K DS	265s	281s (6% slower)

The percentage added delays in the table above are represented in the graph in Figure 5-3. It can be seen that Native Java-SGN requests take slightly longer than OSGi requests, and that the added delay increases with increasing number of devices.

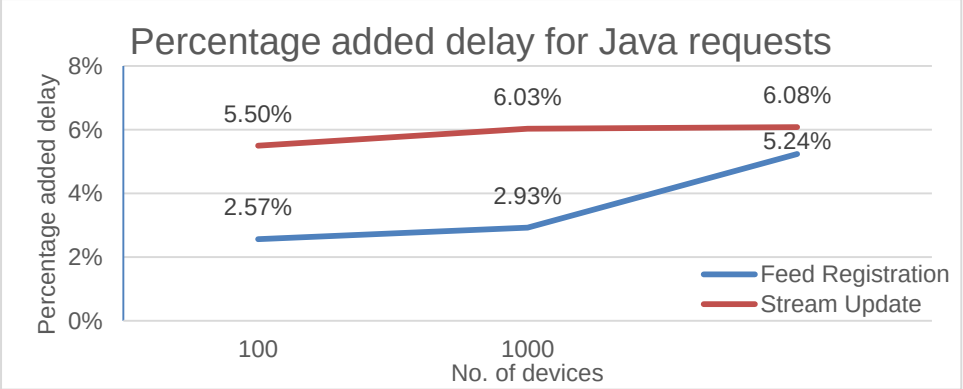


Figure 5-3: Percentage added delay for Native Java-SGN request when compared to OSGi requests

5.3 Effect of CPPM-TBAC on Response Time (Non-Aggregated Payloads)

The serial sensor payload submission procedure (non-aggregated payloads) is shown in Figure 5-4. It shows multiple devices being connected to the client, each sending sensor readings either periodically or when stimulated. The purpose of the client is to construct payloads for each device interaction. The payloads are constructed in a way such that they can be processed by the SAW network (if they are being submitted to the server) or the connected devices (if they are being submitted to the devices). Multiple devices can connect to the client at the same time.

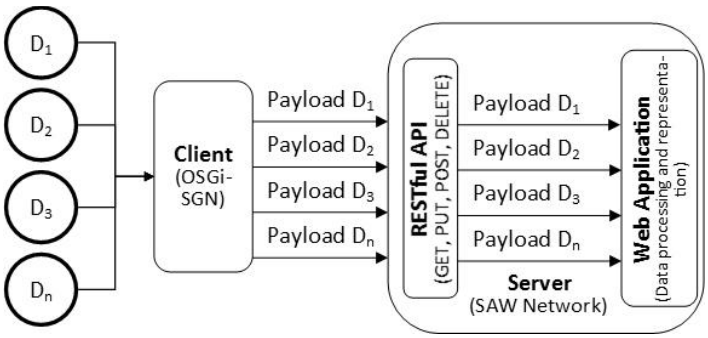


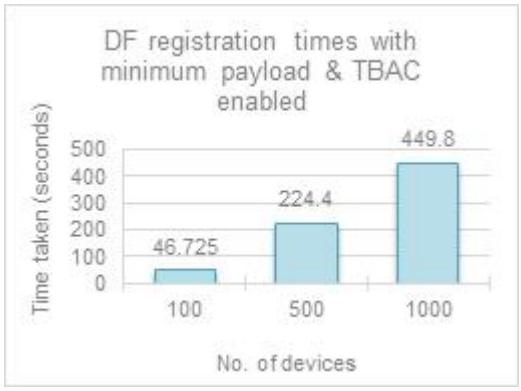
Figure 5-4: CPPM-TBAC serial payload submission procedure

Each payload is processed and transmitted to the SAW API sequentially by the client. For example, the client will submit the payload D1 to the SAW API, and then wait for a response. When it has received a response, it will send the next payload.

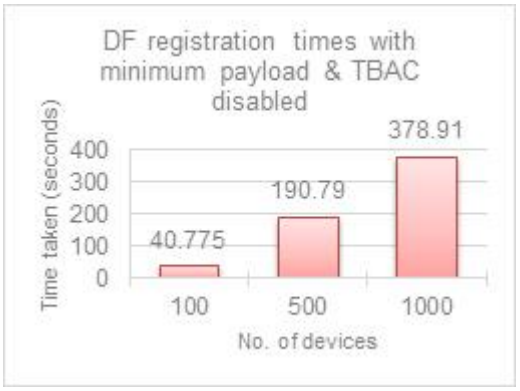
Consequently, the API receives and processes each payload in isolation of the other payloads. This means that the server needs to initialise a new processing action and a database connection for each payload it receives under this methodology. So for example, if *n* number of payloads are submitted in this manner and assuming that each payload uses the same access token, instead of the server having to check the access token only once, it will have to check it n times because each payload is captured and processed in isolation.

5.3.1 Response Times for DF Registrations with Payloads of Varying Sizes

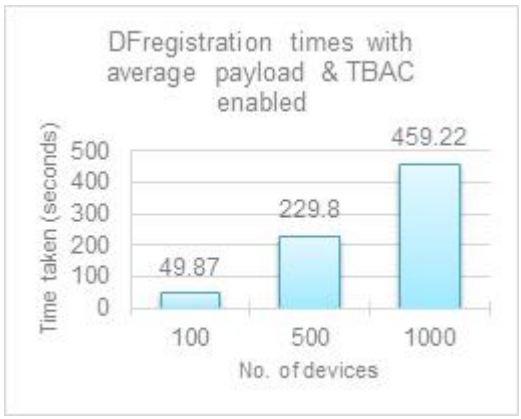
The response times are measured for the OSGi-SGN and presented below.



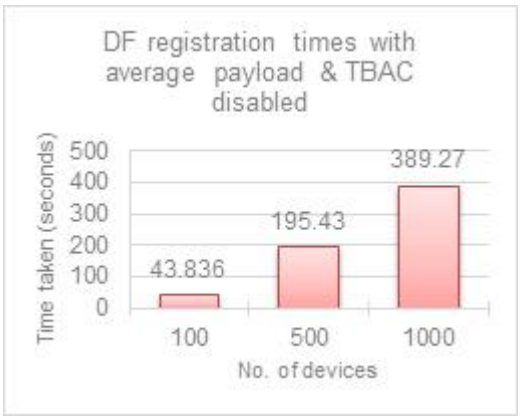
(a)



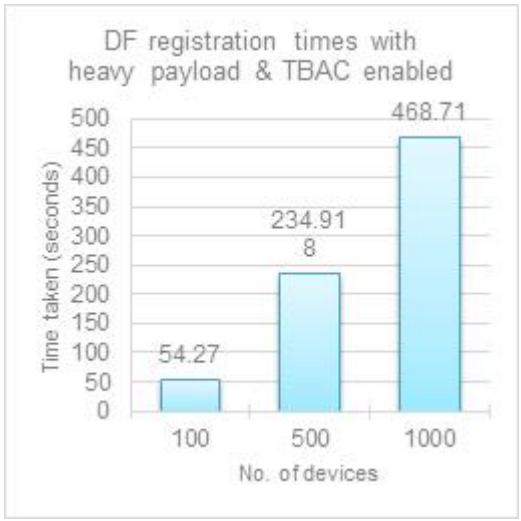
(b)



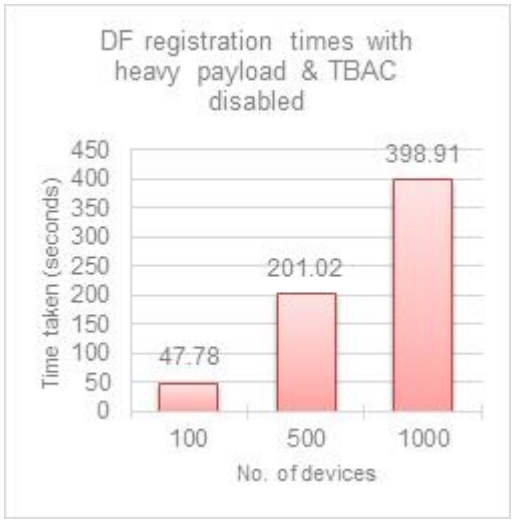
(c)



(d)



(e)



(f)

Figure 5-5: DF registration times, in seconds, for minimum, average and heavy payloads and with TBAC enabled and disabled

Figure 5-5 (a) and Figure 5-5 (b) show the minimum payload. Figure 5-5 (c) and Figure 5-5 (d) show the average payload. Figure 5-5 (e) and Figure 5-5 (f) show the heavy payload.

Figure 5-6, Figure 5-7 and Figure 5-8 show the minimum, average and heavy DF registration payloads respectively.

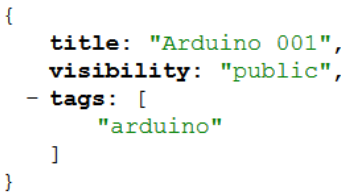


Figure 5-6: Minimal DF registration payload

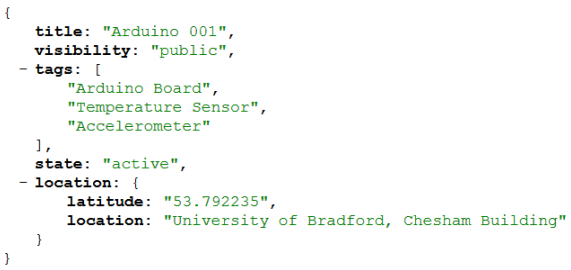


Figure 5-7: An average DF registration payload

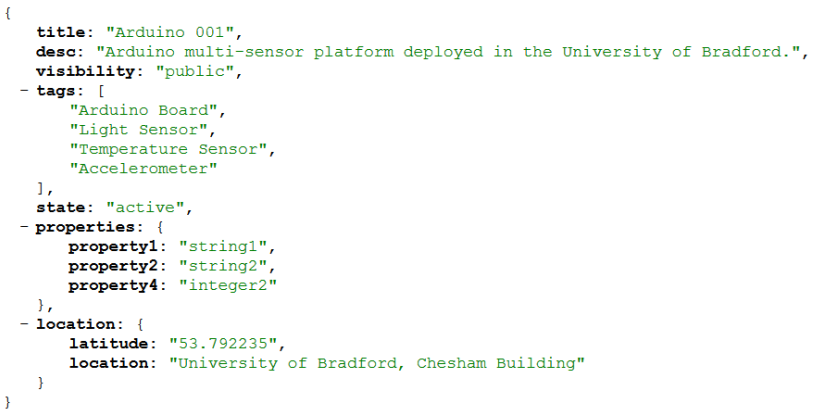


Figure 5-8: A verbose DF registration payload

The minimal payload only contains essential fields as shown in Figure 5-6. Essential fields are the minimum set of fields that SAW expects the payload to contain. The size of this payload is a mere 84 bytes. The average payload in Figure 5-7 contains some optional fields alongside the essential fields shown in the minimal payload. The size of this payload is around 290 bytes. The heavy payload in Figure 5-8 is even more verbose and is around 520 bytes.

The comparison of the device registration times with and without the proposed TBAC mechanism, for the minimum, average and heavy payloads, are presented in Table 5-3, Table 5-4 and Table 5-5 respectively.

For the minimum payload, it can be seen that registration of 100 DF takes around 40 seconds when TBAC is disabled, which is increased to 46 seconds when TBAC is enabled, resulting in a percentage added delay of 14.6%. On the higher scale when registering 1,000 DF, it takes nearly 6 minutes and 19 seconds with TBAC disabled and 7 minutes and 30 seconds with TBAC enabled. This translates to a percentage added delay of 18.7% which is only marginally higher than the increased delay for 100 devices.

Table 5-3: Comparison of DF registration times with minimum payload with TBAC on/off

Number of DF registered	With TBAC disabled	With TBAC enabled
100	40.8 seconds	46.7 seconds (14.6% slower)
500	190.8 seconds	224.4 seconds (17.6% slower)
1,000	378.9 seconds	449.8 seconds (18.7% slower)

Table 5-4: Comparison of DF registration times with average payload with TBAC on/off

Number of DF registered	With TBAC disabled	With TBAC enabled
100	43.8 seconds	49.8 seconds (13.8% slower)
500	195.4 seconds	229.8 seconds (17.6% slower)
1,000	389.3 seconds	459.2 seconds (18% slower)

Table 5-5: Comparison of DF registration times with heavy payload with TBAC on/off

Number of DF registered	With TBAC disabled	With TBAC enabled
100	47.8 seconds	54.3 seconds(13.6% slower)
500	201 seconds	234.9 seconds(16.9% slower)
1,000	398.9 seconds	468.7 seconds(17.5% slower)

For the average payload, the comparisons are similar. Registration of 100 DF takes around 44 seconds when TBAC is disabled. This is increased to 50 seconds when TBAC is enabled, resulting in a percentage added delay of

13.76%. For 1,000 devices, it takes nearly 6 minutes and 29 seconds with TBAC disabled and 8 minutes and 39 seconds with TBAC enabled. This translates to a percentage added delay of 17.97% which, again, is only marginally higher than the increased delay for 100 DF.

Similarly, for the heavy payload, registration of 100 devices takes around 48 seconds when TBAC is disabled. This is increased to 54 seconds when TBAC is enabled, resulting in a percentage added delay of 13.6%. Following the same trend, for 1,000 devices it takes nearly 7 minutes and 39 seconds with TBAC disabled and 8 minutes and 49 seconds with TBAC enabled. This translates to a percentage added delay of 17.5%. Once again, this is only a slight increase over the percentage added delay for 100 devices.

It can be seen from the presented information and statistics that TBAC introduces a noticeable added delay when registering DF. This is the trade-off that is incurred in order to get new security features. The added delay when using TBAC is most significant with a small number of devices, and is comparatively less with a very large number of devices. Hence it can be said that for a large scale cloud-based networks, the proposed TBAC would be highly suitable.

It can also be observed that the added delay increases as the payload size increases. These increases can be seen in Table 5-6, which displays a comparison between the average and heavy payloads with TBAC on and off. For example, the heavy payload takes 47.8 seconds to register 100 devices without TBAC. The same configuration with the average payload which takes 43.8 seconds. Therefore, the percentage added delay when registering 100 DF with TBAC off is 9.1% for the heavy payload when compared to the

average payload. This is the same as saying that registering 100 DF with TBAC off is 9.1% slower for the heavy payload when compared to the same configuration using the average payload.

These statistics indicate that the payload size can have a small influence on the response times of DF registrations. However, it can be clearly seen that the percentage added delays tend to decrease as the number of DF registrations increase, and the decrease is more significant when TBAC is turned on. This, once again, demonstrates that the proposed CPPM-TBAC mechanism is suitable for large scale cloud-based networks as the percentage added delay is minimal when dealing with a large number of devices.

Table 5-6: Comparison of DF registration times for different payload sizes

Number of DF registered	Average payload		Heavy payload	
	TBAC off	TBAC on	TBAC off	TBAC on
100	43.8s	49.8s	47.8s (9.1% slower)	54.2s (8.8% slower)
500	195.4s	229.8s	201s (2.9% slower)	234.9s (2.2% slower)
1,000	389.2s	459.2s	398.9s (2.5% slower)	468.7s (2.1% slower)

As to the reason for why TBAC introduces a noticeable delay on the response times, this is due to the extra processing required at the server-side for processing the token permissions and resource details. Mostly this involves extra queries to the database system for fetching the token permissions and also the details of the resources (DF, DS and DP) to be acted upon (create, read, update, delete). A potential area for future development can be the optimisation of the query generation and execution process when determining and processing access rights for resources (e.g.

not reusing already existing data, not caching frequently-used query calls, etc.) to reduce the response times when TBAC is turned on.

5.3.2 Response Times for Uploading DP

When devices (DF and their corresponding DS) have been registered to the network, sensor data can begin to upload to the network. This sensor data is referred to as DP. A DP payload is simple and usually only contains two fields: a field to specify the time the measurement was taken and another to specify the measurement value. The type of measurement (e.g. temperature, pressure, etc.) and other similar properties should already be defined for the DS submitting the DP so there is no need to replicate this information at the DP payload. The typical size of this payload in the SAW network is around 60-80 bytes and examples of it have been illustrated in Chapter 3. The following experiment tests the response times for sending these DP from the client to the server. The payload used for this experiment is shown in Figure 5-9.

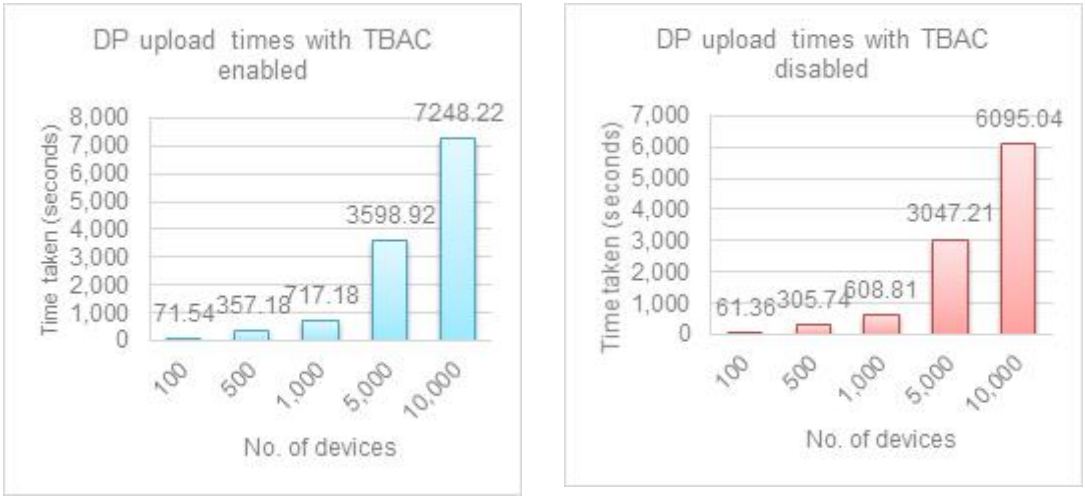

```
{
    at: "2013-06-29T15:24:54+00:00",
    value: "31.567"
}
```


Figure 5-9: DP payload showing the date of measurement and the sensor reading at that time

The response times for submitting these DP payloads are displayed in Figure 5-10 (a) and Figure 5-10 (b).

The response times for uploading DP are higher compared to registering new DF because uploading DP requires loading the resource models of the parent DF and DS as well. This is why it becomes necessary to fetch the parent resources as well to satisfy the extensive set of permissions that can be modelled in the CPPM-TBAC tokens. For example, a token can provide permissions to create new DP. In CPPM-TBAC, permissions of DS cascade

down to DP. In this sort of token, the permissions will be specified for the DF or DS. Therefore, in order to determine if the provided token can be used to act on the DP being submitted, the system will have to fetch the parent DS and DF for the DP and check the permissions provided by the token against these resources. View section 3.4.3 for more details on this procedure. Obviously this incurs additional costs in terms of database processing. Uploading 100 DP takes around 61 seconds when TBAC is disabled. This is increased to 72 seconds when TBAC is enabled, resulting in a percentage added delay of 16.6%. On the higher scale when uploading 1,000 DP, it takes around 10 minutes and 9 seconds with TBAC disabled and 11 minutes and 57 seconds with TBAC enabled. This translates to a percentage added delay of 17.8% which is a marginal increase of 1.2% from the percentage added delay for 100 DP. The full set of comparisons are available in Table 5-7.



(a) (b)

Figure 5-10: Time taken to upload DP with TBAC enabled and disabled

Table 5-7: Comparison of DP upload times with TBAC on/off

Number of DP uploaded	With TBAC disabled	With TBAC enabled
100	61.4 seconds	71.5 seconds (16.6% slower)

500	305.7 seconds	357.2 seconds (16.8% slower)
1,000	608.8 seconds	717.2 seconds (17.8% slower)

5.4 Effect of CPPM-TBAC on Response Time (Aggregated Payloads)

The results presented for the asset model in the previous sections have used serial requests from the client to the server. Under this technique, requests are submitted one after another with no query optimisation. The large response time occurs due to the instantiation of the DB for each and every request. For example, uploading 1,000 DP requires sending each and every single individual payload by itself to the server. Once the server responds, the next payload is sent, and so on.

A more efficient method of uploading numerous payloads is to first aggregate them and then upload them to the server in a single request. There are many aggregation techniques available and it is obvious that aggregated payloads will have better performance in terms of response times because the number of requests from the client to the server will be reduced and more processing will be undertaken in each request. The aim in this thesis is to analyse and compare the scale of differences in response times between serial requests and lumped-sum requests in order to prepare for the selection of an efficient technique.

The aggregated payload procedure is illustrated in Figure 5-11. As with the serial sensor payload submission procedure, multiple devices can be connected to the client, each sending sensor readings either periodically or when stimulated. This procedure is quite similar to the previous procedure but varies in two major aspects:

1. At the client end: The client has to decide how many payloads to combine and how to package this combination as a new aggregated

payload. It should be kept in mind that the current iteration of SAW only allows usage of a single access token for each request (whether it's a single payload or an aggregated payload). Thus, the client has to ensure that it only aggregates payloads for DF, DS and DP that can be processed by the network with the supplied token. Since this intelligence is currently not available in the client node, for simulation purposes the payloads for aggregation are manually generated depending on the supplied token to ensure that the request is valid;

2. At the server API end: The server API has to be able to recognise an aggregated payload submission and then extract the individual payloads for processing. As mentioned in the previous point, the server expects a single access token with each request. This access token is used to check the associated grants stored in the database to determine whether the client's request can be fulfilled.

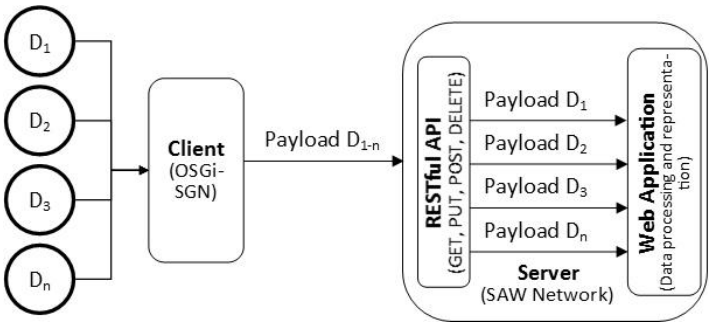


Figure 5-11: CPPM-TBAC aggregated payload submission procedure

There are obvious disadvantages to this, however, and some of these are as follows:

- There needs to be extra capability and intelligence at the client side in order to select suitable payloads for aggregation, and to also determine the optimum number of payloads for aggregation.

- The client or the server will also need extra functionality and intelligence to handle cases where aggregated payloads are lost during transmission so that they can be re-transmitted if possible.
- There can be significant added delay in the propagation of a large payload through the network due to its size.
- The server needs to carry out extra processing to extract individual payloads from the aggregated payload and then process these individually.

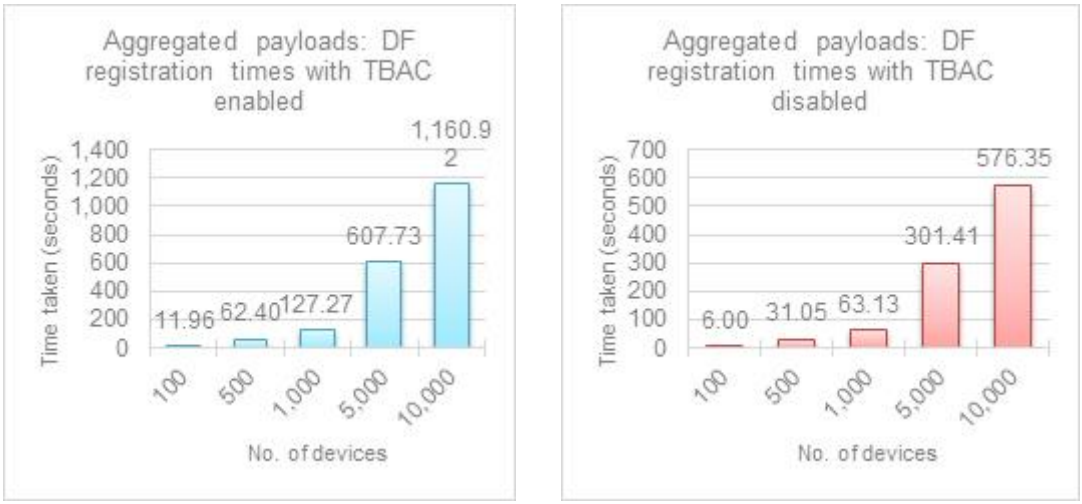
But the biggest disadvantage by far is that this scheme will only work in the SAW network if all the individual payloads in the aggregated payload can be processed using the same single token used by the client. This restriction applies because the current implementation of SAW can only accept a single token per request from the client. This restriction has significant implications on the aggregated payload submission scheme because it can be expected that aggregated payloads will contain multiple assets (DF, DS and DP), belonging to multiple users, requiring multiple tokens. Therefore, it must be kept in mind that while this scheme is being presented here as a really efficient alternative to the serial payload submission procedure, it cannot realistically be implemented with the current deployment of the SAW network. However, it is definitely something that is an area of further research and development, and hence the decision to include it in the thesis.

In this scenario, the server would handle all the processing in one request. This means that the connection to the database (both MySQL and MongoDB) will only be initialised once.

This simulation was run for DF registrations ranging from 100 devices to 1,000 devices. The results of the simulation are presented in Table 5-8 and plotted in Figure 5-12 (a) and Figure 5-12 (b).

Table 5-8: DF registration times for the aggregated payload submission procedure.

Number of DF registered	With TBAC disabled	With TBAC enabled
100	6 seconds	11.958 seconds(99.2% slower)
500	31 seconds	62.4 seconds(100.9% slower)
1,000	63.1 seconds	127.3 seconds(101.6% slower)



(a) (b)

Figure 5-12: DF registration time with TBAC enabled and disabled for the aggregated payload submission procedure

Two things can be noted with these results:

1. The response times are significantly faster in this scenario. There is an improvement of almost 700% when TBAC is off (Figure 5-13 (b)) and nearly 400% when TBAC is on (Figure 5-13 (a)).
2. The added delay when TBAC is on is almost double.

In regards to the first point, it can be see here that aggregating payloads to reduce the number of requests made to the server greatly improves the response time. This is mainly due to the reduction in the number of database

initialisations that need to be done, as this is the most costly operation on the server. Reducing the number of database initialisations leads to a great improvement in response times because the server can do more work with each database connection.

The improvements in response times with TBAC enabled are shown in Figure 5-13 (a) and the improvements with TBAC disabled are shown in Figure 5-13 (b).

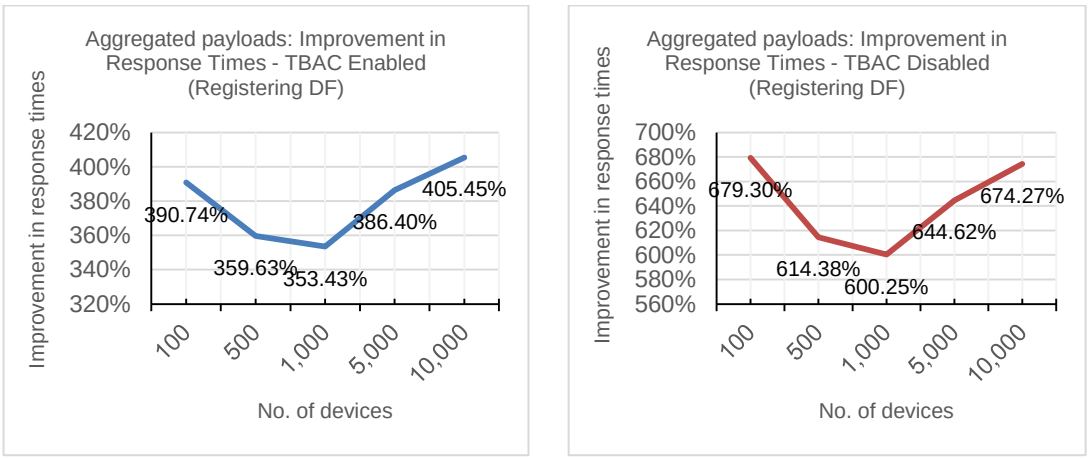


Figure 5-13: Improvement in DF registration time with TBAC enabled and disabled for the aggregated payload submission procedure

It is seen from Table 5-8 that in this scenario, the response times doubles when TBAC is enabled. In comparison, the added delay in response times seen in the serial requests scenarios was in the region of 15-30%. However, the increase of response times to just over 100% when TBAC is enabled in the aggregated payload submission procedure can be easily explained.

When TBAC is enabled, the number of queries to the database increase significantly due to checking of permission policies for the supplied token. However, the added delay due to this process is relatively small compared to the time taken to initialise and close down the database, and is thus quite

largely masked in the overall response time for serial payload submission procedure scenarios. For the aggregated payload submission procedure scenarios, however, this delay is more noticeable because the database is not being initialised or closed down as the payloads are being processed. So in the aggregated payload submission scenarios, the actual added delay for using TBAC is being seen.

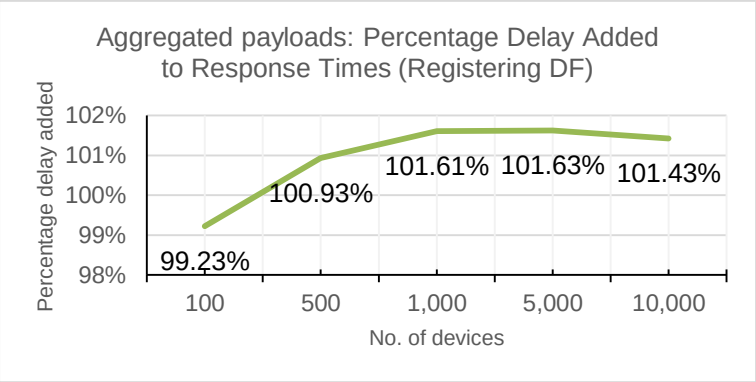


Figure 5-14: Percentage delay added on DF registration times when using the CPPM-TBAC scheme for the aggregated payload submission procedure

More importantly, it is important to note that once again, the percentage added delay remains relatively uniform as we increase the number of devices being registered from 100 devices to 1,000 devices. The plot for the added delay for the aggregated payload submission procedure can be seen in Figure 5-14. The percentage added delay only increases by a mere 2.4% as the number of devices increases by 10 times from 100 devices.

5.5 CPPM-TBAC Analysis

5.5.1 TBAC Scaling Efficiency

It is clear that any access control mechanism will undoubtedly introduce some level of delay to the network due to additional processing and authentication checks, so the actual delay is not the focal point of concern in this regard. The real problem that needs to be addressed in a dynamic

environment like the WoT is the issue of delay caused as the number of devices increases for very large networks. In this regard, it is important for an access control scheme to maintain a relatively uniform percentage added delay across an increasing number of devices.

In Figure 5-15 (a) and Figure 5-15 (b) it can be seen that the proposed CPPM-TBAC scheme scales very well with both an increase in the number of DF being registered and the increase in number of DP being uploaded to the SAW network.

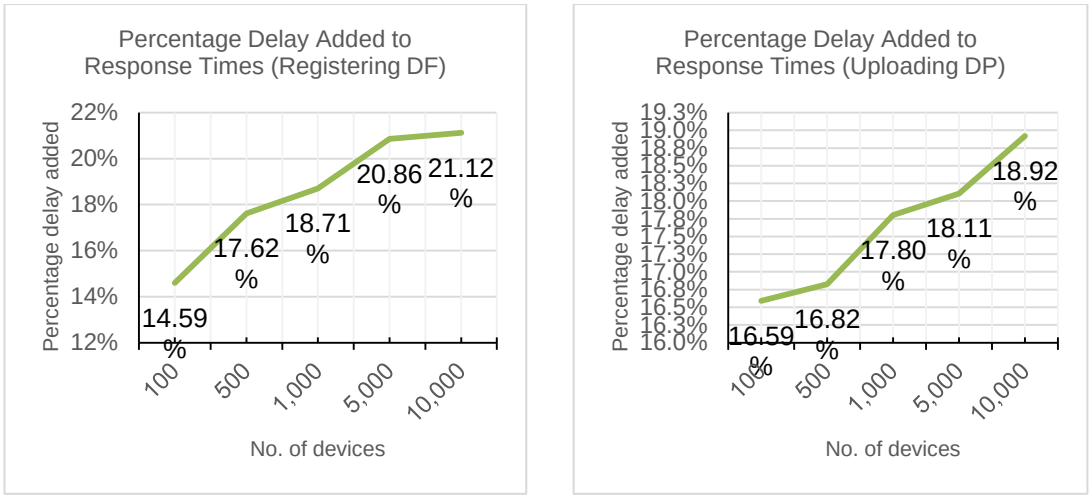


Figure 5-15: Percentage delay added on DF registration and DP upload response times when using the CPPM-TBAC scheme

The added delay with TBAC on in response times when registering 100 DF is 14.59%. For 1,000 devices, this percentage added delay increases by a mere 4.12% to 18.71%, even though the number of DF being registered increases by 10 times.

Similarly, the added delay with TBAC on in response times when uploading 100 DP is 16.59%. For 1,000 DP, this percentage added delay increases by just 1.21% to 17.8%, even though the number of DP being uploaded increases by 10 times. It can be seen here that the percentage added delay

for uploading DP to the SAW network remains relatively uniform as the number of DP increases from a modest 100 DP to a much more resource-intensive 1,000 DP. This uniform increase in percentage added delay is paramount for achieving scalability in a dynamic, temporal and high-load environment.

5.5.2 Tokens as a means of Dynamic Access Control

CPPM-TBAC tokens are designed in such a fashion so as to facilitate the dynamic generation of access rights for numerous resources of a user by using a small number of tokens. Each token, if needed, can model access rights for all resources of a user all the way from coarser DF right down to the finer DP. However, this may lead to degraded performance due to the intensive processing of a large number of access rights each and every time a resource is accessed via the token.

Each token has the capability to specify a set of global permissions for a specific user. The token can then specify access rights for DF either according to their visibility grouping (public or private), or according to specific DF IDs. The token can then specify access rights for the DS of each DF, again, either by specifying grants according to the visibility grouping of the DS or, for even finer access, according to the DS IDs. If access rights are specified for a specific DF or DS, then those access rights can easily be removed in the future without affecting the token in regards to access for other resources it has modelled. This effectively allows dynamic granting and revoking of access rights for each token.

The same is true for when an existing token needs to be updated for additional resource grants of new or existing resources. This can be useful

for upgrading an agent's access to new resources or subsets of existing resources without issuing a new token, thereby improving usability and manageability of tokens. This technique can also be used to degrade an agent's access to resources, again, without necessitating the issuing of a new token if that is what is desired in the specific application scenario. However, this may not be the best option because using the same token over long periods of times can pose a security risk if someone manages to illegally obtain the token. A token with a shorter life period is considered to be more secure.

Network administrators can therefore dynamically assign and revoke grants for each and every single token for any level of granularity by either using visibility level groupings for coarser control or specific DF and DS IDs for fine-grained access management. Each token can be enhanced or reduced whenever the need arises.

5.5.3 Improving Security

Tokens can be set to expire in the near future (temporal tokens) to force agents to request new tokens for continued access to network resources. This technique can be used to improve the security of the TBAC system and mitigate security threats related to compromised tokens (e.g. if a hacker manages to get access to a legitimate token).

If temporal tokens are not used and access still needs to be revoked for an agent, then the token in question can simply be deleted. When the agent tries to access the network resources using the deleted token, then the request will not be honoured. It goes without saying that this method should be used with care due to its vague nature.

Network administrators can also setup automated token renewal capability within the system to automatically expire and renew tokens for agents. This, again, can help mitigate threats from compromised tokens.

There are also various other issues pertaining to the security implications of using CPPM-TBAC that have not been discussed in the thesis due to limitation in time and scope. Some of the more prominent issues are listed below with brief commentaries that are not aimed at solving the problem of security but rather highlighting it as an area of future work:

- Token generation and propagation mechanisms/procedures: Currently all tokens are generated manually with a set of defined permissions for existing users and resources. They are then manually sent to clients who need to make use of them. There is no automated mechanism or procedure in place to propagate the generated tokens to clients in a secure fashion. Possible solutions to this problem can be implementation of cryptographic key generation and exchange mechanisms like Diffie–Hellman key exchange (D–H) and RSA [99]. By using a shared secret, tokens can be securely transmitted to clients in an automated fashion.
- Automatic token expiration: While network administrators have some control over the life period of generated tokens to control when they expire, there is no automated mechanism of identifying compromised tokens and automatically expiring them to prevent further security breaches. Various methods and parameters can be employed to identify compromised tokens (e.g. abnormal differences in source IP/requester location).

All of these issues can be considered potential areas of future work that can be investigated further to improve the security of the system.

5.5.4 Automated Access Grants Using Visibility Groups

It has been mentioned earlier that tokens can specify access rights according to visibility groups of DF and DS. This has the added benefit of automating the granting of access for new resources that are added to the network, if this is desired.

For example, if a token grants read and update access for public DF and all its public DS, then it will automatically grant access to new public DF and their public DS. If some of the new resources need to be excluded from this rule, then either their visibility group can be changed or specific access rights for them added to the token by specifying the DF and DS IDs. The automated granting of access leads to reduced maintenance and easier usage and deployment of the tokens, whilst the ability to override grants for specific resources helps to overcome the drawback in manageability of the TBAC system.

5.6 Semantic Profiling Analysis

One possible application of the SAW framework is in DM applications where a multitude of sensing devices will be collecting and uploading large amounts of sensor data for immediate processing. In this type of application, the system needs to be able to readily and effectively identify devices accurately and profile their characteristics in a semantic fashion so as to enable machine-initiated interaction with the sensing devices and the collected data. The simulation scenarios presented here take these considerations into account. The simulations intends to profile and measure the ability of the

framework to map devices to the correct corresponding concepts. When tags are submitted, more than one matching concept can be returned by the system. This, of course, depends on the existing knowledge base and the list of tags provided to the framework. The complete list of experiments carried out are detailed in Table 5-9 and justified below the table.

The actual semantic profiling mechanisms have been detailed earlier in Chapters 2 and 3. Simulation scenarios are presented in this section to test the suitability and effectiveness of the semantic profiling mechanisms when annotating a set of sensing platforms semantically.

Table 5-9: List of semantic annotation experiments

Experiment No	Section	No. of Concepts	No. of Devices	Concepts to Devices Ratio
1	5.6.1	10	50	1:5
2	5.6.2	20	100	1:5
3	5.6.3	50	100	1:2
4	5.6.4	10	100	1:10
5	5.6.5	10	500	1:50

Sections 5.6.1 and 0 present simulation scenario sets for a concepts to devices ratio of 1:5. The purpose of experiment 2 is to compare the effects of increasing the number of concepts and devices whilst keeping the concepts to devices ratio the same as experiment 1. Section 5.6.3 presents a simulation scenario set for a hypothetical and improbable concepts to devices ratio of 1:2. The purpose is not to analyse real-world performance as this ratio is highly improbably, but rather to identify trends between the different concepts to devices ratios and provide a basis for further comparison. For the same purpose, sections 5.6.4 and 0 present simulation scenario sets for concepts to devices ratios of 1:10 and 1:50 respectively.

Finally, section 5.6.6 compares and analyses the results collected for the various simulations.

5.6.1 Semantic Profiling Simulation Scenario Set 1: 10 Concepts and 50 Devices

This simulation uses the baseline setup so that the number of devices is 50 and the number of concepts 10. Each concept has 5 possible devices it can map to. This produces a concepts to devices ratio of 1:5.

In an ideal case, the following values will be expected for each measured parameter:

1. Total concepts generated: 10.
2. Duplicate concepts: 0. This implies 100% mapping accuracy.
3. Total concepts with 5 mapped devices: 10
4. Total concepts with less than 5 mapped devices: 0
5. Total concepts with more than 5 mapped devices: 0
6. Average concepts returned by the system when tags are submitted: Ideally this needs to be above 1 to indicate that the system is returning at least 1 valid concept for each mapping.
7. Percentage of primary concepts returned by the system when tags are submitted: No ideal value.
8. Percentage of secondary concepts returned by the system when tags are submitted: No ideal value.
9. Total cases where one concept has a bigger weight than the rest of the returned concepts: 100% of all cases.
10. Total cases where all returned concepts had equal weight: 0% of all cases.

5.6.1.1 Simulation 1: All 50 devices submitted in a random fashion and provided with one random tag from the corresponding concepts

In this scenario a single tag is being used to register the device and to map it with a semantic concept. As expected, this gives rise to a large number of semantic concepts being generated as the single tag proves insufficient in mapping new devices to existing concepts. The results in Table 5-10 show that the system generates around 73% duplicate concepts in this scenario.

Table 5-10: Results for simulation scenario set 1: Simulation 1 (1 tag)

Statistic	Result	Comments
Total concepts generated	38	
Duplicate concepts	28	73.7% of concepts generated are duplicates
Total concepts with 5 mapped devices	0	
Total concepts with less than 5 mapped devices	38	29 concepts with 1 mapped device; 6 concepts with 2 mapped devices; 3 concepts with 3 mapped devices;
Total concepts with more than 5 mapped devices	0	
Average concepts returned by the system when tags are submitted	0.26	
Percentage of primary concepts returned by the system when tags are submitted	100%	
Percentage of secondary concepts returned by the system when tags are submitted	0%	
Total cases where one concept has a bigger weight than the rest of the returned concepts	13	100% of total cases
Total cases where all returned concepts had equal weight	0	0% of total cases

Another expected outcome is the lack of concepts having all their devices mapped to them successfully. Results show that no concepts were able to achieve this in the given scenario. In fact, 29 concepts, which accounts for 76% of the total generated concepts in this scenario, were only mapped to 1

device. Only 3 concepts, i.e. around 8% of the total generated, were successful in mapping at least 3 devices successfully.

Furthermore, the average number of concepts returned when tags were being submitted to the system is well below 1, which is the reason for the large number of duplicates seen in this scenario. In fact, only 26% of the profiling attempts resulted in the system returning one or more matched concepts, and in all these cases, no secondary mappings were produced. This is due to the small number of tags which are being stored in the knowledge base. If, on the other hand, the knowledge base had more data to work with resulting in it becoming richer with each mapping, then the results would be significantly different as is seen in the following two simulation scenarios where more tags are used and therefore more knowledge added to the repository.

To conclude, submitting a small number of tags leads to a slower enriching of the knowledge base, less useful results, higher chance of generating duplicate semantic concepts, but almost 100% chance of obtaining a clearly distinguished semantic concept which has a higher weight than the rest of the returned concepts, suggesting very strong likelihood of a positive match.

5.6.1.2 Simulation 2: All 50 devices submitted in a random fashion and provided with two random tag from the corresponding concepts

In this scenario two tags are being used to register the device and to map it with a semantic concept. The results show that the system generates around 44% duplicate concepts in this scenario, which is an improvement of 33.3% from scenario 1. The results, as presented in Table 5-11, also show an improvement of 11.1% in successfully mapping 5 devices to the corresponding concept compared to scenario 1, where no concept was

mapped to all of its 5 devices. A negative outcome of the addition of an extra tag in the profiling phase can be observed in the form of false positives where 2 concepts ended up with having more than 5 devices mapped to them. This is a decline in performance of 11.1% compared to the same statistic in scenario 1.

Table 5-11: Results for simulation scenario set 1: Simulation 2 (2 tags)

Statistic	Result	Comments
Total concepts generated	18	
Duplicate concepts	8	44.4% of concepts generated are duplicates. Improvement of **33.3%** from scenario 1.
Total concepts with 5 mapped devices	2	11.1% of total generated concepts. Improvement of **11.1%** from scenario 1.
Total concepts with less than 5 mapped devices	14	77.8% of total generated concepts. 5 concepts with 1 mapped device; 5 concepts with 2 mapped devices; 2 concepts with 3 mapped devices; 2 concepts with 4 mapped devices;
Total concepts with more than 5 mapped devices	2	11.1% of total generated concepts. Decline of **11.1%** from scenario 1.
Average concepts returned by the system when tags are submitted	1.04	Improvement of **400%** from scenario 1.
Percentage of primary concepts returned by the system when tags are submitted	78.8%	
Percentage of secondary concepts returned by the system when tags are submitted	21.2%	
Total cases where one concept has a bigger weight than the rest of the returned concepts	27	90% of all cases; Decline of **10%** from scenario 1.
Total cases where all returned concepts had equal weight	3	10% of all cases; Decline of **10%** from scenario 1.

Another obvious enhancement is the 4 times increase in the average number of concepts returned by the system when two tags are used to map each device compared to the same statistics observed in scenario 1. This enhancement results from the better and speedier enrichment of the

knowledge base from using two tags instead of one. The results also show that due to the enrichment of the knowledge base, secondary mappings are also more likely to be returned with each request. Finally, it can be observed that the percentage of cases where all returned concepts had the same weight (and therefore introducing ambiguity for autonomous profiling) increased by 10% compared to scenario 1, suggesting that more tags potentially lead to bigger ambiguity. This is not a big issue for human clients but can become problematic for machine agents where a decision has to be made regarding the concept to map the device to according to the weights of the returned concepts.

To conclude, obvious improvements can be observed in reduction of duplicate concept generation when using two tags instead of one. The system also produces more accurate mappings for the devices but introduces a risk of generating false positives, and at the same time, ambiguous results.

5.6.1.3 Simulation 3: All 50 devices submitted in a random fashion and provided with three random tag from the corresponding concepts

In this scenario three tags are being used to register the device and to map it with a semantic concept. The results show that the system generates around 23% duplicate concepts in this scenario, which is an improvement of 21.3% from scenario 2 and a huge improvement of 54.6% from scenario 1. Table 5-12 also shows that there is an improvement of 19.7% in successfully mapping 5 devices to the corresponding concept compared to scenario 2, which translates to a respectable increase of 30.8% compared to scenario 1.

The results show significant improvements in the average number of concepts returned by the system when mapping devices with an increase of 23% from simulation scenario 2. This means that human clients are given more options when mapping their devices, resulting in higher likelihood of the correct corresponding semantic concept being selected in the end.

Table 5-12: Results for simulation scenario set 1: Simulation 3 (3 tags)

Statistic	Result	Comments
Total concepts generated	13	
Duplicate concepts	3	23.1% of concepts generated are duplicates. Improvement of **21.3%** from scenario 2. Improvement of **54.6%** from scenario 1.
Total concepts with 5 mapped devices	4	30.8% of total generated concepts. Improvement of **19.7%** from scenario 2. Improvement of **30.8%** from scenario 1.
Total concepts with less than 5 mapped devices	7	3 concepts with 1 mapped device; 1 concepts with 2 mapped devices; 1 concepts with 3 mapped devices; 2 concepts with 4 mapped devices;
Total concepts with more than 5 mapped devices	2	7.7% of total generated concepts. 1 concept with 9 mapped devices; Decline of **4.27%** from scenario 2; Decline of **15.38%** from scenario 1.
Average concepts returned by the system when tags are submitted	1.28	Up **23%** from scenario 2. Up **492%** from scenario 1.
Percentage of primary concepts returned by the system when tags are submitted	75%	
Percentage of secondary concepts returned by the system when tags are submitted	25%	
Total cases where one concept has a bigger weight than the rest of the returned concepts	33	89.2% of total cases. Decline of **0.8%** from scenario 2. Decline of **10.8%** from scenario 1.
Total cases where all returned concepts had equal weight	4	10.8% of total cases. Decline of **0.8%** from scenario 2. Decline of **10.8%** from scenario 1.

Finally, and as expected, using three tags instead of two or one tags has resulted in increased likelihood of returning more than one concept with the

same weight, thereby increasing the ambiguity of the returned concepts and making it especially difficult for machine agents to choose the correct corresponding concept. The decrease, however, is less than 1% compared to scenario 2, so it is not a huge change but does, nonetheless, point towards a correlation where increasing the number of tags during the profiling phase leads to the generation of more ambiguous returned concepts.

5.6.1.4 Comparison of Using 1, 2 and 3 Tags

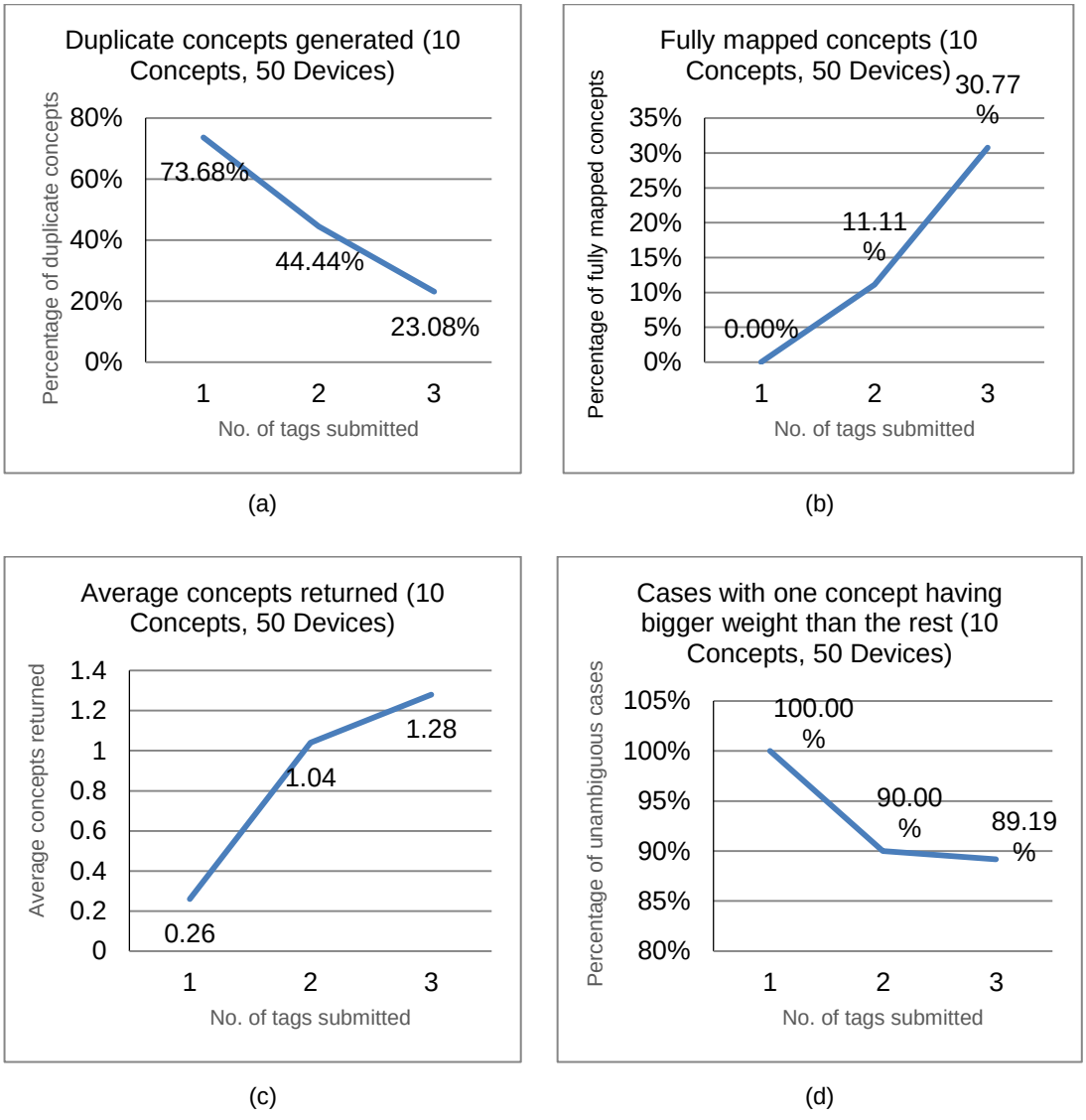


Figure 5-16: Simulation scenario set 1 results showing duplicate concept generation, fully mapped

As the number of tags is increased in each of the simulation scenarios above, the following observations can be made:

1. The generation of duplicate concepts is reduced (Figure 5-16 (a)) significantly. There is an improvement of 54.6% when 3 tags are used as opposed to 1, meaning that generation of duplicate concepts is halved by increasing the number of tags from 1 to 3. It can also be observed that the improvement in reduction of duplicate concepts is only 21.3% when using 3 tags compared to 2, indicating that the trend is approaching a saturation point. Increasing the number of tags even further may help to identify the rough saturation point for this trend.
2. The number of fully mapped concepts increase (Figure 5-16 (b)). This is due to various reasons but primarily because of the reduction in duplicate concepts being generated, which in turn is because more concepts are being returned during each mapping (point 3 below). This shows that increasing the number of tags leads to better accuracy of the system in mapping devices to their corresponding concepts.
3. The average returned concepts increase (Figure 5-16 (c)). As explained before, this is due to better and speedier enrichment of the knowledgebase which means that more concepts are turned for each device mapping, thereby resulting in a higher likelihood of an accurate mapping and less chance of generating duplicate concepts.
4. The ambiguity increases (number of cases where one concept has a bigger weight than the rest decrease) (Figure 5-16 (d)). The increase

in percentage for this stat is quite small but it is there nonetheless and points towards a trend where increasing the number of tags leads to more difficulty for autonomous agents to identify the correct returned semantic concept to map the device to.

5.6.2 Semantic Profiling Simulation Scenario Set 2: 20 Concepts and 100 Devices

Table 5-13: Results for simulation scenario set 2

Statistic	100 devices mapped with 1 tag (S1)	100 devices mapped with 2 tags (S2)	100 devices mapped with 3 tags (S3)
Total concepts generated	72	38	27
Duplicate concepts	52	18 -24.9% from S1	7 -46.3% from S1 -21.4% from S2
Total concepts with 5 mapped devices	0	5 +13.2% from S1	8 +29.6% from S1 +16.5% from S2
Total concepts with less than 5 mapped devices	72	29	15
Total concepts with more than 5 mapped devices	0	4 -10.5% from S1	4 -14.8% from S1 -4.3% from S2
Average concepts returned by system when tags are submitted	0.33	0.89 +273% from S1	1.16 +356.9% from S1 +130.3% from S2
Percentage of primary concepts returned by system when tags are submitted	100%	88.8%	80%
Percentage of secondary concepts returned by system when tags are submitted	0%	11.2%	20%
Total cases where one concept has a bigger weight than the rest of the returned concepts	27	57 -6.1% from S1	33 -6.6% from S1 -0.5% from S2
Total cases where all returned concepts had equal weight	2	9 +6.6% from S1	4 +6.7% from S1 +0.1% from S2

This simulation uses the same setup as before but instead the number of devices has been increased to 100 and the number of concepts to 20. In effect, both the number of concepts and the number of devices have been

doubled. As before, each concept still has 5 possible devices it can map to. So the ratio of concepts to devices still remains at 1:5. The results for the simulation are shown in Table 5-13.

It can be seen here that the results for this simulation are comparable to the simulation scenario set 1 where 10 concepts were being mapped to 50 devices. This was expected as the ratio of concepts to devices was the same in both cases.

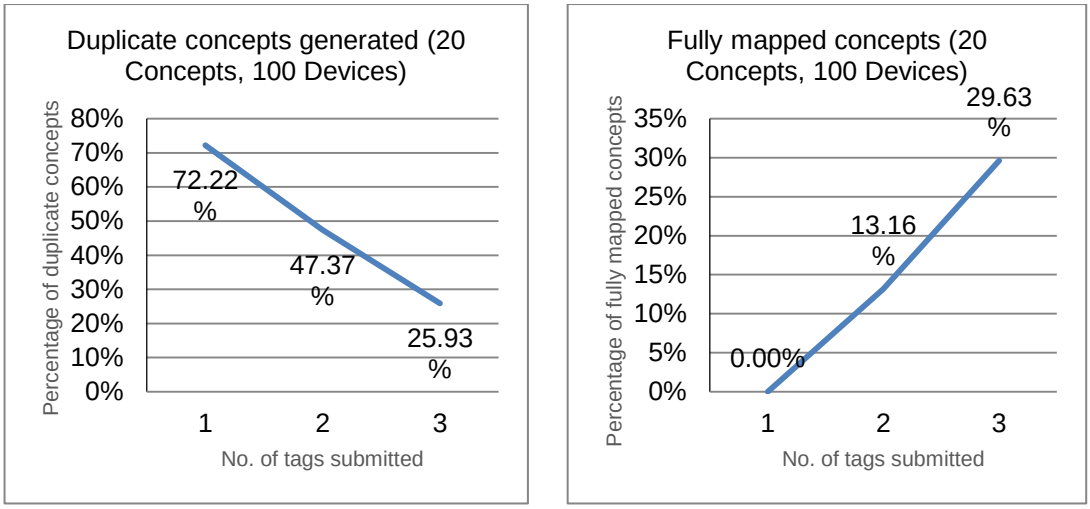
There is a similar drop in the percentage of duplicate concepts generated as the number of tags is increased from 1-3. The percentage of duplicate concepts generated drops by 46.3% when 3 tags are used instead of 1. This is similar to the 50.6% drop observed in simulation scenario set 1.

Similarly, the percentage of fully mapped devices rises by 29.6% when 3 tags are used instead of 1, and again this is the same as the 30.7% percentage rise seen for the same stat in simulation scenario set 1.

The percentage of cases where all returned concepts had equal weights also follows the same trend, with usage of 3 tags seeing an increase of 6.7% in mapping ambiguity when compared to usage of a single tag. This is similar to the increase of 10.8% seen for the same stat in simulation scenario set 1.

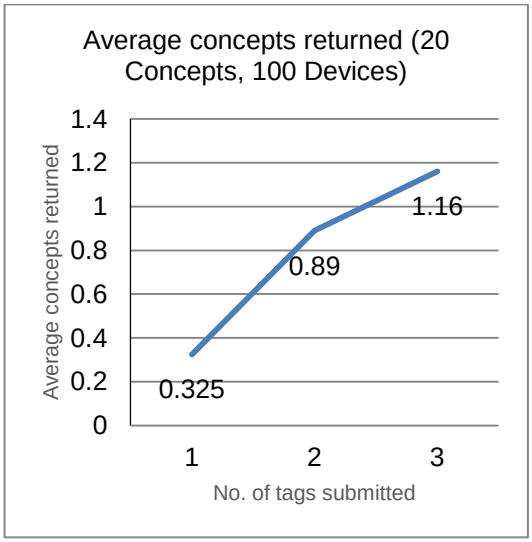
In summary, as the number of tags is increased, similar conclusions can be made for this simulation scenario set:

1. The generation of duplicate concepts is reduced (Figure 5-17 (a)).
2. The number of fully mapped concepts increase (Figure 5-17 (b)).
3. The average returned concepts increase (Figure 5-17 (c)).
4. The ambiguity increases (number of cases where one concept has a bigger weight than the rest decrease) (Figure 5-17 (d)).

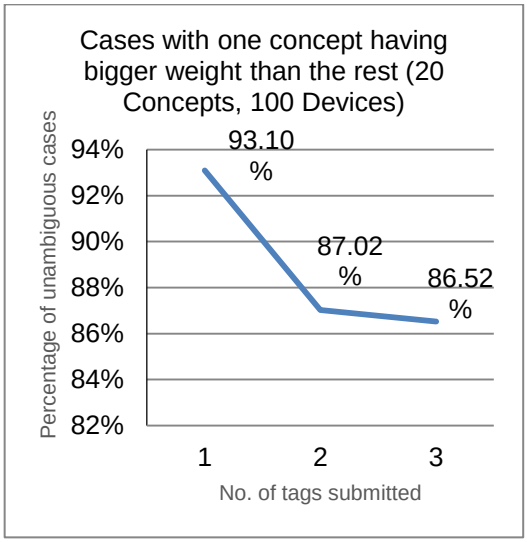


(a)

(b)



(c)



(d)

Figure 5-17: Simulation scenario set 2 results showing duplicate concept generation, fully mapped concepts, average returned concepts and cases with one concept having bigger weight than the rest

5.6.3 Semantic Profiling Simulation Scenario Set 3: 50 Concepts and 100 Devices

In this simulation, the number of concepts has been raised to 50 while the number of devices is still kept at 100. This means that each concept has 2 possible devices that can be mapped to it. The ratio of concepts to devices is thus 1:2. It should be noted that whilst this simulation is carried out for analysis purposes, it is highly improbable that in real-life operation of the system, the concepts to devices ratio would be this small. In real-life

operation, the number of concepts will be few (these are the different types of sensing devices and platforms) and the number of devices many (these are the individual instances of the aforementioned devices). Having said that, this scenario is still simulated in order to learn the effects of increasing and decreasing the ratio of concepts to devices. The results for this simulation scenario set are shown in Table 5-14.

Table 5-14: Results for simulation scenario set 3

Statistic	100 devices mapped with 1 tag (S1)	100 devices mapped with 2 tags (S2)	100 devices mapped with 3 tags (S3)
Total concepts generated	91	72	52
Duplicate concepts	41	2 -14.5% from S1	2 -41.2% from S1 -26.7% from S2
Total concepts with 2 mapped devices	10	27 +26.5% from S1	47 +79.4% from S1 +52.9% from S2
Total concepts with less than 2 mapped devices	81	44	4
Total concepts with more than 2 mapped devices	0	1 -1.4% from S1	1 -1.9% from S1 -0.5% from S2
Average concepts returned by system when tags are submitted	0.1	0.39 +390% from S1	0.59 +590% from S1 +151.3% from S2
Percentage of primary concepts returned by system when tags are submitted	100%	100%	100%
Percentage of secondary concepts returned by system when tags are submitted	0%	0%	0%
Total cases where one concept has a bigger weight than the rest of the returned concepts	10	29	49
Total cases where all returned concepts had equal weight	0	0	0

Compared to the preceding scenarios, the following observations can be made straight away:

- The percentage of duplicate concepts generated is around 20-30% less compared to the earlier scenarios, with 3 tags only generating 3.9% duplicate concepts (a reduction of 41.2% compared to usage of 1 tag in the same simulation).
- The percentage of fully mapped concepts is almost 60% higher with the usage of 3 tags compared to the earlier scenarios as it crosses 90%. This translates to an increase of 79.4% in this stat when compared to the usage of 1 tag.
- The average concepts returned by the system during the mapping process is a lot lower. This is because of the high mapping accuracy which leads to only the correct concepts being returned by the system.
- In all instances, no secondary concepts are returned by the system. Again, this is due to the high mapping accuracy of the system where all mappings are being achieved through primary tags.
- In all instances, the system returned a concept with a higher weight than the rest of the concepts for 100% of the mappings. This is related to the low average concepts returned stat and the high mapping accuracy. This translates to a 0% ambiguity for autonomous agents in all cases which is excellent. However, as mentioned before, this simulation scenario is unrealistic and not applicable to real-life operation.

In summary, as the number of tags is increased, similar conclusions can be made for this simulation scenario set as well:

1. The generation of duplicate concepts is reduced (Figure 5-18 (a)).

2. The number of fully mapped concepts increases dramatically (Figure 5-18 (b)).
3. The average returned concepts increase, but never rise above 1 (Figure 5-18 (c)).
4. The ambiguity remains at 0% throughout (Figure 5-18 (d)).

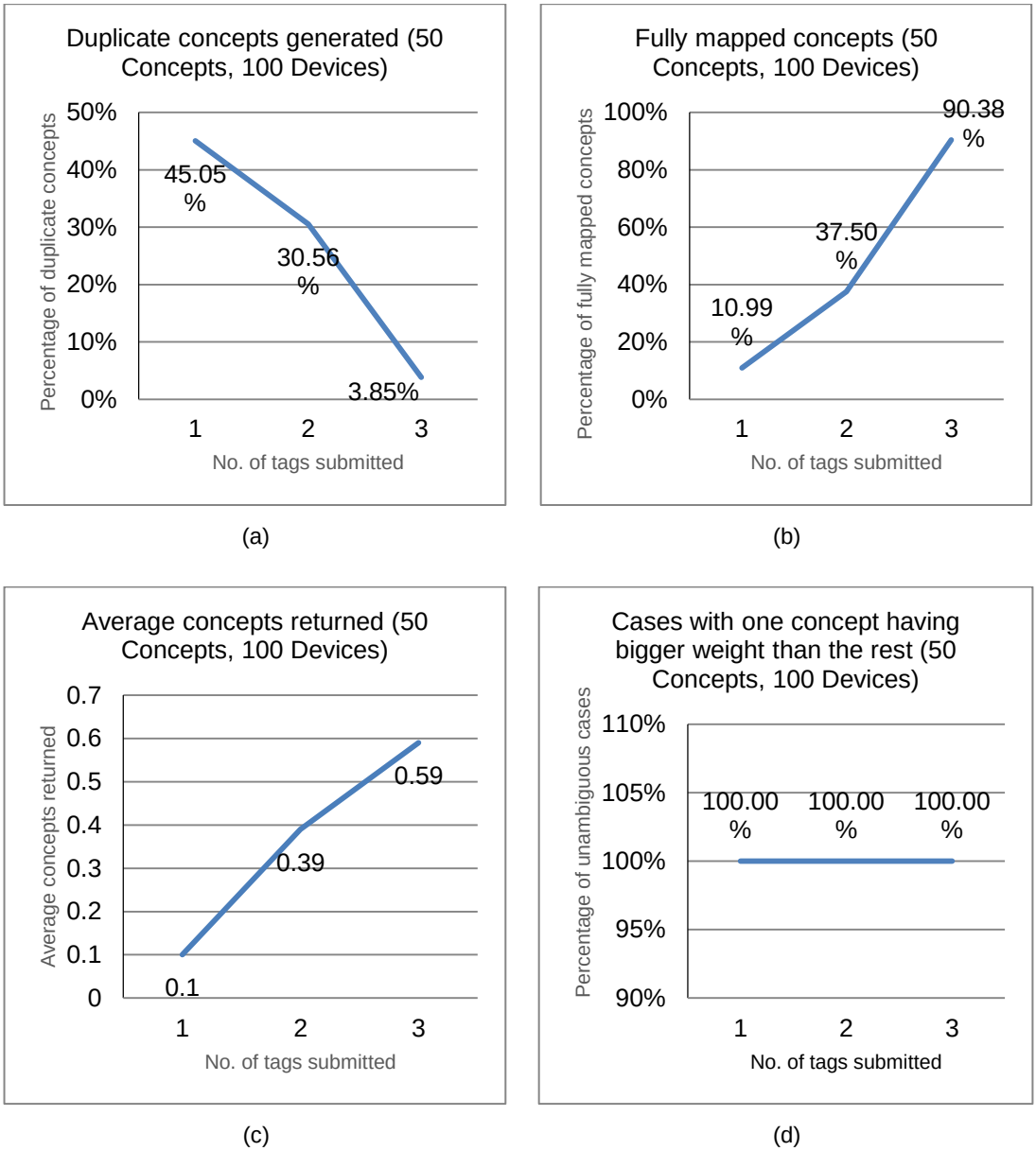


Figure 5-18: Simulation scenario set 3 results showing duplicate concept generation, fully mapped concepts, average returned concepts and cases with one concept having bigger weight than the rest

5.6.4 Semantic Profiling Simulation Scenario Set 4: 10 Concepts and 100 Devices

This simulation uses the same setup as simulation scenario set 1 but instead the number of devices has been increased to 100, such that each concept has 10 possible devices that can be mapped to it. The ratio of concepts to devices is now 1:10.

Table 5-15: Results for simulation scenario set 4

Statistic	100 devices mapped with 1 tag (S1)	100 devices mapped with 2 tags (S2)	100 devices mapped with 3 tags (S3)
Total concepts generated	56	24	16
Duplicate concepts	46	14 -**23.8%** from S1	6 -**44.6%** from S1 -**20.8%** from S2
Total concepts with 10 mapped devices	0	2 +**8.3%** from S1	4 +**25.0%** from S1 +**16.7%** from S2
Total concepts with less than 10 mapped devices	55	21	9
Total concepts with more than 10 mapped devices	1	1 -**2.4%** from S1	3 -**17.0%** from S1 -**14.6%** from S2
Average concepts returned by system when tags are submitted	0.78	1.43 +**183.3%** from S1	1.71 +**219.2%** from S1 +**119.6%** from S2
Percentage of primary concepts returned by system when tags are submitted	100%	87.6%	78%
Percentage of secondary concepts returned by system when tags are submitted	0%	12.4%	21.9%
Total cases where one concept has a bigger weight than the rest of the returned concepts	28	66 +**23.5%** from S1	77 +**29.4%** from S1 +**6.0%** from S2
Total cases where all returned concepts had equal weight	17	11	7

It was mentioned before that in real-life operation, the number of concepts will be few and the number of devices many. This simulation scenario set is therefore conducted primarily to compare it against the simulation scenario

set 1, where the concepts to devices ratio of 1:5. The results for this simulation scenario set are shown in Table 5-15.

In general, this simulation scenario set has similar results to simulation scenario set 1.

- The number of duplicate concepts generated decrease as the number of tags is increased. The percentage of duplicate concepts generated is around 9%-14% higher in each case compared to simulation scenario set 1, however.
- The number of fully mapped devices increases as the number of tags is increased. However, this percentage is smaller compared to simulation scenario set 1.
- The average returned concepts increase as the number of tags is increased. However, there are two notable differences:
    - The average returned concepts are higher in each case compared to simulation scenario set 1.
    - The increase in the average returned concepts as the number of tags is increased is noticeably lower.

However, there is a major difference in this simulation scenario set compared to simulation scenario set 1. Whereas before the ambiguity would increase as the number of tags increased, now the ambiguity is seen to decrease as the number of tags increases. This is apparent from the increase of cases where one concept has higher weight than the rest of the returned concepts as the number of tags is increased. This appears to show a trend where having a high concepts to devices ratio actually helps to reduce the ambiguity of the semantic mapping process as the number of tags is

increased. It is also important to note that the ambiguity hovers around the 10% mark for both simulation scenario sets when 3 tags are used.

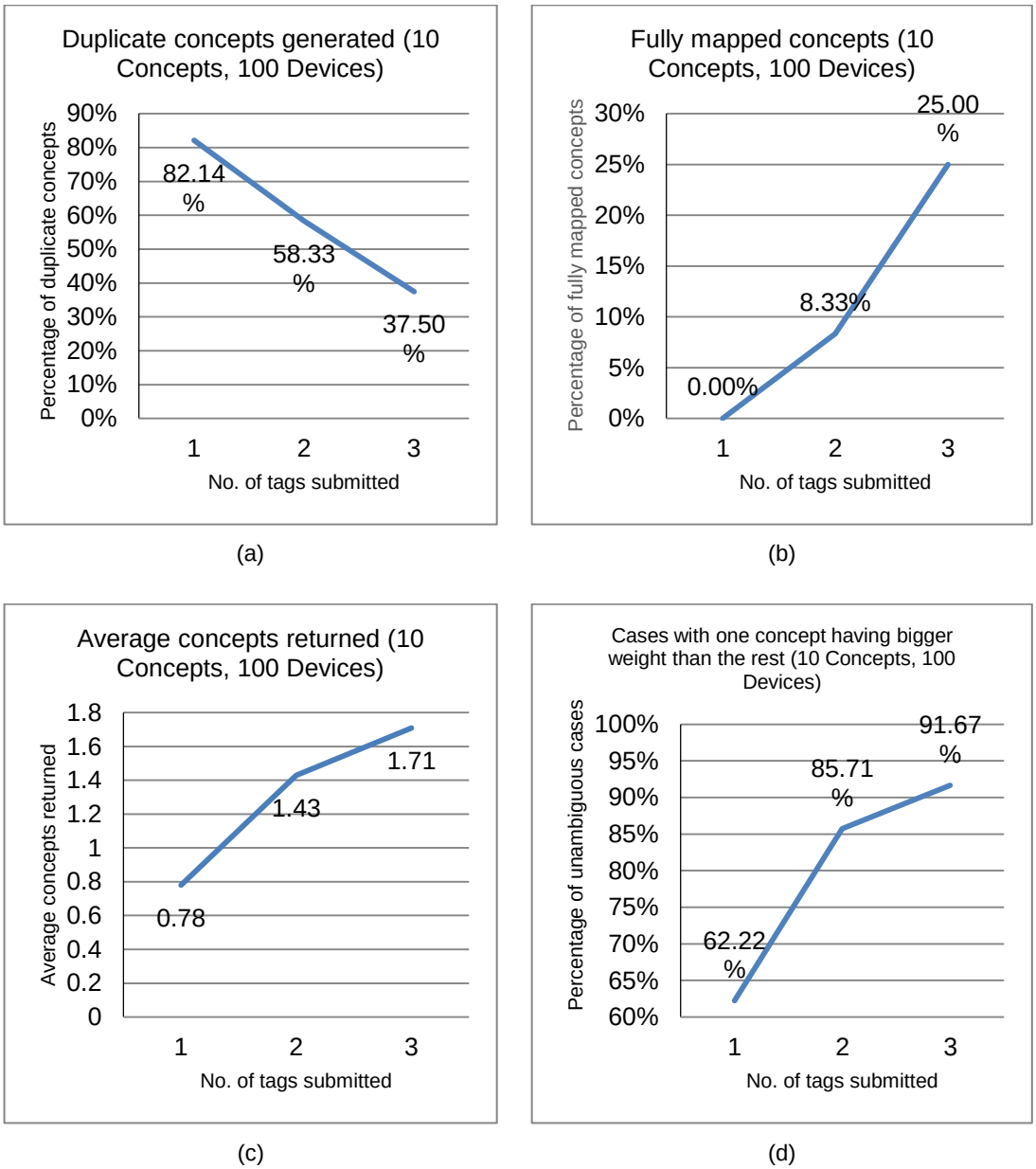


Figure 5-19: Simulation scenario set 4 results showing duplicate concept generation, fully mapped concepts, average returned concepts and cases with one concept having bigger weight than the rest

In summary, as the number of tags is increased, the following conclusions can be made for this simulation scenario set:

1. Duplicate concepts generation is reduced significantly (Figure 5-19 (a)).

2. Number of fully mapped concepts increases dramatically (Figure 5-19 (b)).
3. The average returned concepts increase, and are much higher than the preceding simulation scenario sets (Figure 5-19 (c)).
4. Bucking the trend so far, the ambiguity decreases as the number of tags is increased (Figure 5-19 (d)).

5.6.5 Semantic Profiling Simulation Scenario Set 5: 10 Concepts and 500 Devices

This simulation uses the same setup as before but instead the number of devices has been increased to 500, such that each concept has 50 possible devices that can be mapped to it. In this scenario, the ratio of concepts to devices is 1:50, making it even more near to real-life operation than simulation scenario set 4. The results for this simulation scenario set are shown in Table 5-16.

Most of the statistics follow the same trend as seen in simulation scenario set 4:

- The number of duplicate concepts generated decrease as the number of tags is increased. The percentage of duplicate concepts generated is around 7%-15% higher in each case compared to simulation scenario set 4.
- The number of fully mapped devices increases as the number of tags is increased. However, this percentage is smaller compared to simulation scenario set 4, and the difference is even greater when compared to simulation scenario set 1.

- The average returned concepts increase as the number of tags is increased. Once again, the average returned concepts in each case are greater when compared to simulation scenario set 4.

Table 5-16: Results for simulation scenario set 5

Statistic	500 devices mapped with 1 tag (S1)	500 devices mapped with 2 tags (S2)	500 devices mapped with 3 tags (S3)
Total concepts generated	142	38	18
Duplicate concepts	132	28 -**19.3%** from S1	8 -**48.5%** from S1 -**29.2%** from S2
Total concepts with 50 mapped devices	0	1 +**2.6%** from S1	4 +**22.2%** from S1 +**19.6%** from S2
Total concepts with less than 50 mapped devices	142	35	13
Total concepts with more than 50 mapped devices	0	2 -**5.3%** from S1	1 -**5.6%** from S1
Average concepts returned by system when tags are submitted	1.29	2.30 +**178.1%** from S1	2.42 +**187.5%** from S1 +**105.3%** from S2
Percentage of primary concepts returned by system when tags are submitted	100%	55.8%	53%
Percentage of secondary concepts returned by system when tags are submitted	0%	44.3%	47.2%
Total cases where one concept has a bigger weight than the rest of the returned concepts	277	415 +**12.5%** from S1	444 +**14.8%** from S1 +**2.3%** from S2
Total cases where all returned concepts had equal weight	82	48	39

In summary, as the number of tags is increased, similar conclusions can be made for this simulation scenario set as made for the simulation scenario set 4:

1. The generation of duplicate concepts is reduced (Figure 5-20 (a)).
2. Number of fully mapped concepts increases dramatically (Figure 5-20 (b)).
3. The average returned concepts increase (Figure 5-20 (c)).

4. Similar to simulation scenario set 4, the ambiguity continues to decrease as the number of tags is increased (Figure 5-20 (d)).

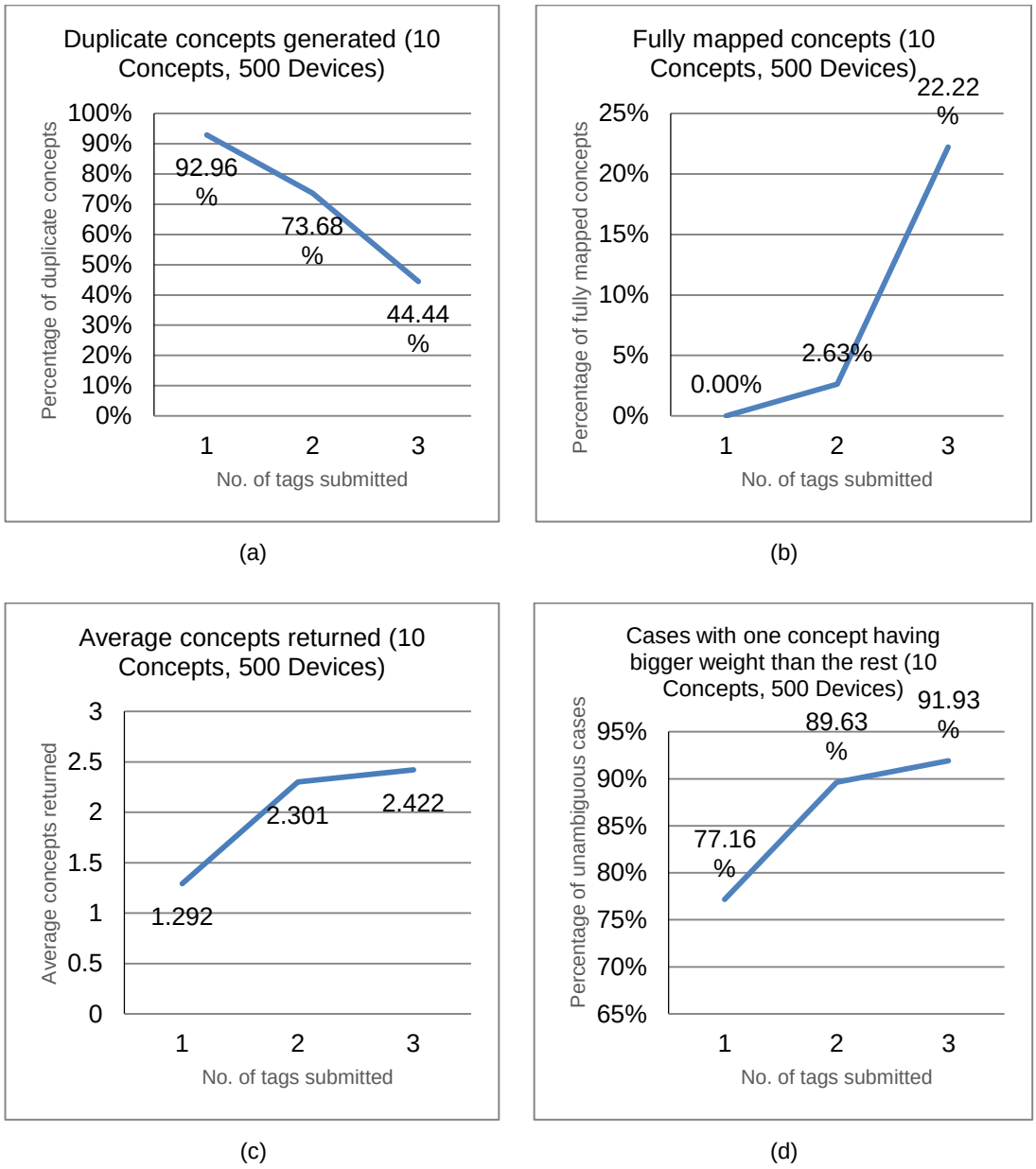


Figure 5-20: Simulation scenario set 5 results showing duplicate concept generation, fully mapped concepts, average returned concepts and cases with one concept having bigger weight than the rest

5.6.6 Comparison of the Varying Concepts to Devices Ratio

This section aims to cross-compare the results obtained for the difference scenarios presented above.

5.6.6.1 Comparison of Duplicate Concepts Generated

The comparison of duplicate concepts generated for concepts to devices ratios of 1:5, 1:10 and 1:50 is displayed in Table 5-17. The same results are plotted and displayed in Figure 5-21.

Table 5-17: Comparison of duplicate concepts generated for 10 concepts with 50, 100 and 500 devices.

Simulation scenario set	1 tag	2 tag	3 tags
10 Concepts, 50 Devices Concepts to devices ratio: 1:5	74%	44%	23%
10 Concepts, 100 Devices Concepts to devices ratio: 1:10	82%	58%	38%
10 Concepts, 500 Devices Concepts to devices ratio: 1:50	93%	74%	44%

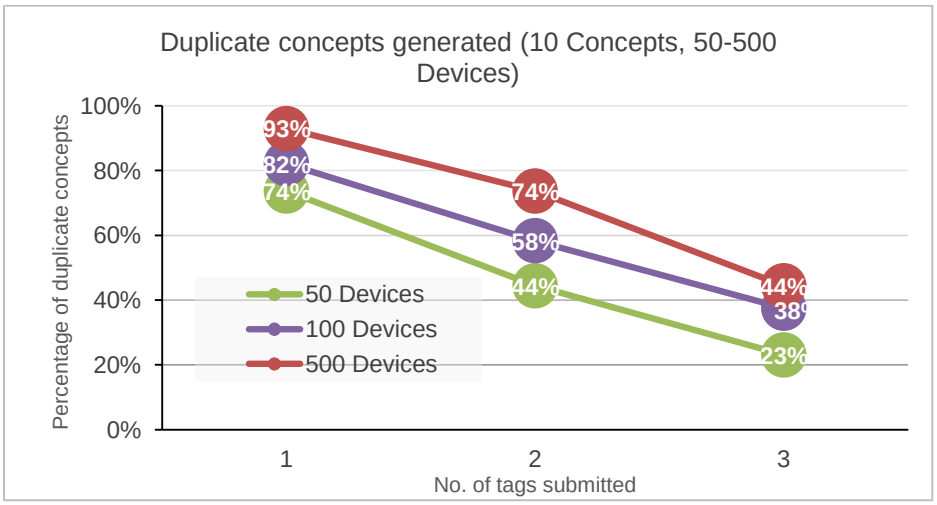


Figure 5-21: Comparison of duplicate concepts generated for 10 concepts with 50, 100 and 500 devices

The results show a similar trend in all cases where increasing the number of tags decreases the percentage of duplicate concepts generated. However, it is important to note that as the concepts to devices ratio increases, the actual percentage of duplicate concepts generated increases. There is a rise of 21% in duplicate concepts generated with the usage of 3 tags as the concepts to devices ratio increases from 1:5 to 1:50. Taken in the context of

a real-life application, these results state that as the concepts to devices ratio increases, the percentage of duplicate concepts generated will also increase. However, increasing the number of tags used in the mapping process will help to reduce the generation of duplicate concepts and therefore improve the mapping accuracy. So this trend is expected to continue as the number of tags is increased. Further work in this area can look at identifying the optimum number of tags to be used to achieve the best balance between the various parameters.

5.6.6.2 Comparison of Fully Mapped Devices

The comparison of fully mapped devices for concepts to devices ratios of 1:5, 1:10 and 1:50 is displayed in Table 5-18. The same results are plotted and displayed in Figure 5-22.

Table 5-18: Comparison of fully mapped devices for 10 concepts with 50, 100 and 500 devices.

Simulation scenario set	1 tag	2 tag	3 tags
10 Concepts, 50 Devices Concepts to devices ratio: 1:5	0%	11%	31%
10 Concepts, 100 Devices Concepts to devices ratio: 1:10	0%	8%	25%
10 Concepts, 500 Devices Concepts to devices ratio: 1:50	0%	3%	22%

The results show a similar trend in all cases where increasing the number of tags increases the percentage of fully mapped concepts. However, it is important to note that as the concepts to devices ratio increases, the actual percentage of fully mapped concepts decreases. Even with this fact in mind, it can be seen that there is only a drop of 10% in duplicate concepts generated with the usage of 3 tags as the concepts to devices ratio

increases from 1:5 to 1:50. The drop in percentage is only 3% when the concepts to devices ratio increases from 1:10 to 1:50.

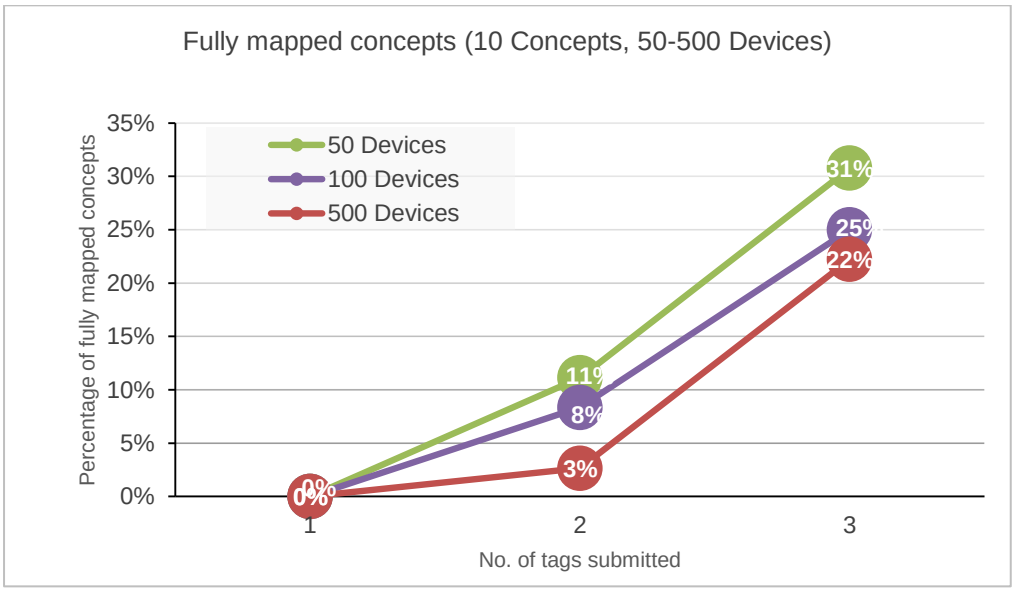


Figure 5-22: Comparison of fully mapped concepts for 10 concepts with 50, 100 and 500 devices

5.6.6.3 Comparison of Average Concepts Returned

The comparison of average returned concepts for concepts to devices ratios of 1:5, 1:10 and 1:50 is displayed in Table 5-19. The same results are plotted and displayed in Figure 5-23.

Table 5-19: Comparison of average concepts returned for 10 concepts with 50, 100 and 500 devices.

Simulation scenario set	1 tag	2 tag	3 tags
10 Concepts, 50 Devices Concepts to devices ratio: 1:5	0.26	1.04	1.28
10 Concepts, 100 Devices Concepts to devices ratio: 1:10	0.78	1.43	1.71
10 Concepts, 500 Devices Concepts to devices ratio: 1:50	1.29	2.30	2.42

Once again the results show a similar trend in all cases where increasing the number of tags increases the average concepts returned. However, as the concepts to devices ratio increases, the number of average concepts returned also increases in response. This is expected because having a

higher number of devices in the knowledgebase increases the likelihood of getting more matches (regardless of whether the actual provided mappings are accurate or not). Translating these results into a real-life application, this shows that with a large concepts to devices ratio, the system is expected to at least return 1 correct or incorrect mapping when any number of tags are used to map the device to an appropriate semantic concept. With the usage of 3 tags, the system is expected to return at least 2 concepts for each mapping.

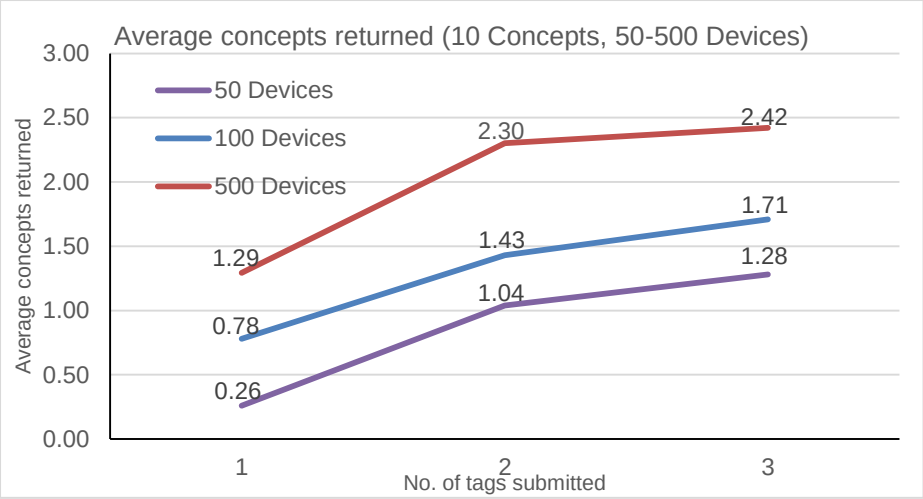


Figure 5-23: Comparison of average concepts returned for 10 concepts with 50, 100 and 500 devices

5.6.6.4 Comparison of Cases with One Concept Having Bigger Weight than the Rest

The comparison of cases with one concept having bigger weight than the rest for concepts to devices ratios of 1:5, 1:10 and 1:50 is displayed in Table 5-20. The same results are plotted and displayed in Figure 5-24.

This is the only case where the general trend varies between the different simulation scenario sets. For a concepts to devices ratio of 1:5 as explained in simulation scenario set 1, the trend is that the ambiguity increases as the number of tags is increased. However, in the following two simulation scenario sets (4 and 5), the trend is the opposite: the ambiguity decreases as

the number of tags is increased. In all simulation scenario sets, when 3 tags are used, the ambiguity hovers around the 10% mark (the percentage of cases with one concept having bigger weight than the rest is around 90%).

Table 5-20: Comparison of cases with one concept having bigger weight than the rest for 10 concepts with 50, 100 and 500 devices.

Simulation scenario set	1 tag	2 tag	3 tags
10 Concepts, 50 Devices Concepts to devices ratio: 1:5	100%	90%	89%
10 Concepts, 100 Devices Concepts to devices ratio: 1:10	62%	86%	92%
10 Concepts, 500 Devices Concepts to devices ratio: 1:50	77%	90%	92%

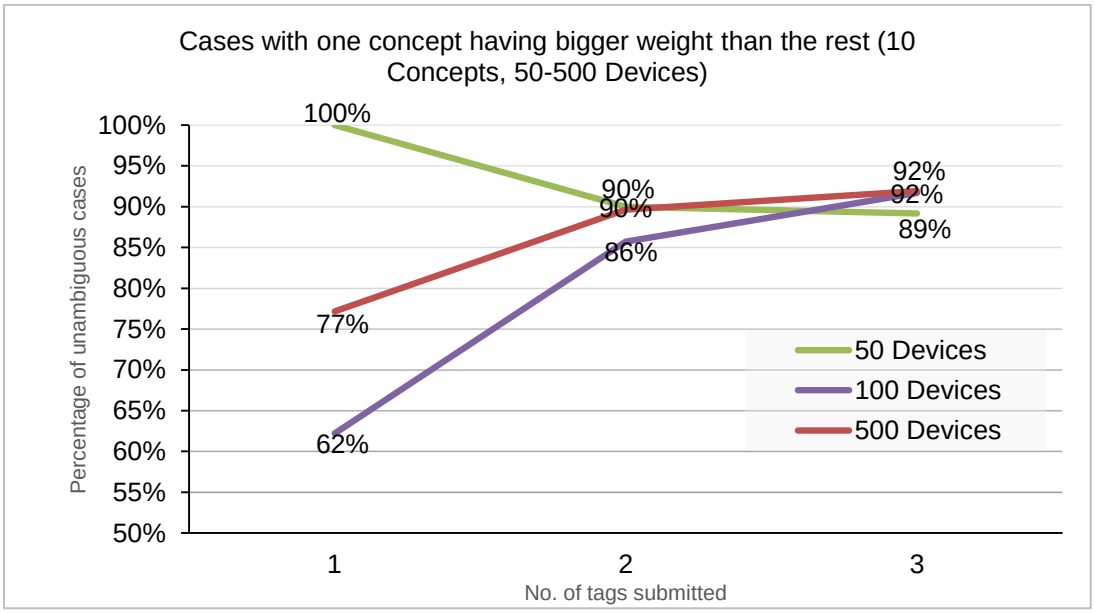


Figure 5-24: Comparison of cases with one concept having bigger weight than the rest for 10 concepts with 50, 100 and 500 devices

This shows two things:

- As the concepts to devices ratio increases, usage of 1 tag increases the ambiguity and creates difficulty for autonomous agents.
- Usage of 3 tags reduces the ambiguity to the 10% mark regardless of the concepts to devices ratio.

5.7 Final Recommendations

SAW is an extensive collaboration framework powered by semantics and this is why it has been necessary to propose and implement both a suitable access control as well as a semantic annotation mechanism. This chapter has presented simulations for both of these mechanisms in an effort to evaluate their performance and identify areas of further research and development.

In regards to CPPM-TBAC, it was observed and evaluated that the proposed mechanism is highly suitable for scaling in large cloud environments such as the WoT. Further areas of research in regards to this include investigations into better forms of security and developing a token propagation mechanism to securely transmit tokens to clients of interest. Another area of further exploitation is the idea of using aggregated payloads and determining how this can best work with the SAW network, and how the limitation on usage of a single token with each request can be overcome in this case.

With regards to the semantic annotation mechanism, it has been demonstrated that the proposed mechanism is suitable for annotating sensing devices in the WoT. It has been shown that the performance of the proposed mechanism increases as both the quality (more primary tags with a high semantic correlation to the device being mapped) and quantity (enrichment of the database and usage of extra tags) of information supplied during the profiling phase increases. Further areas of research and development in this regard include increasing the number of tags to more than 3 to find the saturation point for many of the trends seen in the semantic

annotation results, and investigating if there start to appear any other trends or trade-offs as the concepts to devices ratio is increased even further.

Chapter 6: Verification & Validation of the SAW Framework

The SAW framework has two major components that need to be tested to ensure correct operation: (1) The semantics engine powered by the Apache Jena framework and which takes care of the semantic annotation of resources, and (2) The PHP-based web application which exposes the functionalities of the SAW framework to external networks. Both components were unit tested to ensure correct operation of their individual components. Unit testing is a testing mechanism whereby small units of functionality (e.g. a particular function/method within a class) are tested in isolation of other functionalities to ensure that they operate as expected. It can be used to perform integration testing as well by using multiple units in a single test. It involves *asserting* entities to check whether they match the given criteria (e.g. does it contain the given string, does it match the given value, is it of the correct type, etc.). The necessary data required for the unit test is setup before each test. Similarly, any changes made by the unit test are reversed/rolled-back after the unit test has finished executing to ensure that all unit tests are independent of the operation of other unit tests.

The semantic engine extension was written in the Java programming language. Because Java is very strongly embedded in Object Oriented Programming (OOP) principles and is strictly typed (e.g. a variable declared as an integer can only accept integers and not strings), basic type testing is taken care of during code compilation. Still, it is a good practice to carry out proper testing to validate the functionality of the system and thus the JUnit unit testing framework was used to test the functionalities of each method in each class. Typically each method in a class has one corresponding unit

test. However, more complex methods which can fail at multiple points and can have a range of varying parameters usually have more than one unit test to take into account all possible cases.

Similarly, PHPUnit was used to write unit tests for the web application which was written in PHP. PHP, whilst having support for OOP, is not as strictly typed nor as strongly embedded in OOP principles. This means that PHP, unlike Java, can have standalone functions and code that does not exist in a class. Similarly, a variable initially declared as an integer in PHP can later accept strings and vice versa. However, since the web application was built using a PHP framework called Laravel, it simplified the testing procedure somewhat as it forced the application to use strong OOP principles and create separation between the back-end logic and the front-end exposition. As with the Java code, unit tests for written for the PHP code to test methods of crucial classes and even to test the operation of complete classes and their integration with other classes.

Each time the semantics engine or the web application code would be updated, the unit tests would be executed to ensure that the new functionalities operated as expected and none of the existing functionalities were broken as a result of the introduction of the new functionalities. This ensured that the SAW framework was thoroughly tested and validated to ensure correct operation at all times.

An example of a unit test is as follows. There is a method called “insertNewDevice” in the semantic engine in the “src/main/java/org.saw.query.AnnotationAgent.java” class which accepts a map consisting of the device properties and the semantic concept ID for

which the device should be inserted as its parameters. To unit test this method, a new class will be created in the test package called “src/test/java/org.saw.query.AnnotationAgentTest.java”, and it will contain a method called “insertNewDeviceTest”. Inside this method attempts are made to call the “insertNewDevice” method with incorrect or invalid parameters (e.g. specifying a concept ID which does not exist, or leaving out vital properties for the device being inserted) and test to see if it fails as expected. This can be done by catching the expected exceptions or monitoring the return of the method and asserting it to be the expected invalid output (or the absence of an expected output). Similarly, attempts are made to supply the correct parameters to the method being tested and test to see if it succeeds as expected. Again, this can be done by asserting the return/output of the method to be equal to the expected value, and also by checking other entities that might have been affected by the method call (e.g. checking the datastore for the insertion of the new device). At the end of the unit test, any data generated through the test is deleted to ensure that all unit tests are independent of each other and that data generated from one unit test does not affect other unit tests (achieves isolation which is a requirement for successful execution of unit tests).

Chapter 7: SAW Use Case: Flood Disaster Management in London

This section presents a quick use case study for the SAW framework to illustrate how it can be used in a real-life situation. Take, for example, a scenario involving a major flood in London, UK. Management of this type of disaster will not only involve coordination between different emergency departments like the Police, Fire Brigade Service, Ambulance Service, HM Coastguard, etc. but equally important will be the task of disseminating critical information to the general public, of which include affected people, people likely to be affected, relative and friends of those in distress, the general public and of course the media. Thus the problem here is not only of timely and controlled data dissemination and collaboration amongst the "active" actors tackling to manage, contain and resolve the disasters but there is also a problem of distributing useful information and updates to "passive" parties so as to inform the general masses with the correct and most up-to-date status information and the relevant procedures to undertake.

Whilst governmental bodies will employ the necessary measures to monitor this type of event and to keep track of developments (e.g. water level across areas of high risk), keeping this data confined internally will hinder public use of this critical information. Exposition of this information would enable interested parties to compose intelligent agents that monitor key events and push alerts or compose mashups to not only aid in the awareness of the disaster situation, but to also prepare a response in a timely manner. But this can only really become possible (both in terms of exposing data as public resources and consuming the resources by the general public) if the mechanisms behind doing so are intuitive, flexible and speedy. If the

governmental body has to setup a horde of accounts and roles and if the public agents have to register accounts to publish or use this data, then the likelihood of its adoption and the usefulness of its exposure will quickly deteriorate due to the expensive investment in time. Instead, if all this access control information could be stored in a few well designed tokens, and then these tokens distributed to those with a need to consume the data without requiring them to register an account, then it can be seen that the effort is more likely to be rewarded with higher adoption and consumption. In this regard, SAW can be used to provide audited access to resources, and semantically annotate them to make them more useful and enable autonomous agent collaboration. Here is a rough list of steps that might be taken to realise this:

1. Create a secure network with appropriate token generation and distribution mechanisms;
2. Register devices to the network and generate admin tokens to enable their administration. Distribute these admin tokens to the parties who are responsible for the management of the sensing infrastructure;
3. Carry out semantic annotation of the devices to enable their representation in a unified schema and interrogation through semantic technologies by participating agents (whether human or machine);
4. Create additional tokens with the appropriate access policies to expose the sensing infrastructure to those who need to consume the data;
5. Revoke tokens for agents that no longer need to consume the data;

6. Repeat steps 4-5 as appropriate. More details on how tokens can be used in a real-life deployment are provided in section 3.4.3.4.

Chapter 8: Conclusion & Future Work

8.1 Conclusion

8.1.1 Summary of problem statement and proposed solutions

This thesis identified various issues pertaining to the representation, annotation and sharing of data. These issues were found to be more apparent and significant when collaborating in multi-party and cross-organisation settings. An analysis of the existing literature revealed no suitable or optimised solutions for enabling efficient collaboration and data exchange in applications involving heterogeneous actors.

Hence, it was the goal of this study to tackle two underlying problems:

- Syntactic-level interoperability: Achieving a consistent data representation;
- Semantic-level interoperability: Achieving a consistent data meaning.

Syntactic-level interoperability is necessary to model and represent data in a standardised way across multiple systems. This facilitates interoperability in terms of terminology and mark-up. To achieve syntactic-level interoperability, this study presented a resource-based asset model.

Semantic-level interoperability is essential for maintaining a consistent meaning of data and definitions across multiple systems and platforms, exposed to multiple actors and vendors. This facilitates interoperability in terms of meaning and understanding. To achieve semantic-level interoperability, this study developed a novel semantic annotation and KM system.

The study has outlined the procedures for developing both the resource-based asset model and also the semantic interaction model for annotating resources. In unison, these models form the SAW network which consists of

an OSGi-based WSN and a cloud-based SAW framework in deployment terms.

8.1.2 Summary of results

The study has carried out an extensive analysis of the three major components of the SAW framework:

- The asset model;
- CPPM-TBAC;
- Semantic profiling.

8.1.2.1 The Asset Model

Simulations were carried out to test the performance of the OSGI-SGN vs Native Java-SGN by registering new devices to the network and also updating definitions of existing devices. It was discovered that the OSGi requests were faster than the Native Java-SGN requests.

The response times for registration of DF and uploading of DP to the SAW network with varying payloads were also measured. Each simulation was performed with both TBAC enabled and disabled. It was ascertained that the usage of TBAC introduced a noticeable added delay in the response times, both when registering new devices and uploading data to the SAW network. This delay increased with the increase in both the payload size and the number of payloads. More importantly, it was discovered that the percentage added delay only increased by a few percent as the number of DF/DS/DP increased from 100 to 1,000. This proved that the proposed CPPM-TBAC scheme scaled very well with an increase in the number of devices on the network. This measure was crucial for proving the scalability of the proposed scheme in a dynamic, temporal and high-load environment.

8.1.2.2 CPPM-TBAC

A comprehensive analysis of the proposed access control scheme resulted in the deduction of the following main advantages of using the scheme:

- The proposed scheme makes it possible for network administrators to dynamically assign and revoke grants for each and every single token for any level of granularity by either using visibility level groupings for coarser control or specific feed and stream ids for fine-grained access management.
- Temporal tokens can be used to increase security.
- Extended access restrictions (e.g. source IP) can be used to increase the security of the tokens.
- Access grants can be automated using visibility groups.

8.1.2.3 Semantic profiling

Finally, a thorough analysis of the semantic profiling process was undertaken to measure the performance and reliability of the proposed tag-based semantic annotation process. A number of conclusions were drawn from this analysis:

- First and foremost, increasing the number of tags used in the profiling phase lead to a decrease in the percentage of duplicate concepts generated. However, as the concepts to devices ratio was increased, the percentage of duplicate concepts generated also increased. Taken in the context of a real-life application, these results state that as the concepts to devices ratio increases, the percentage of duplicate concepts generated will also increase. However, increasing the number of tags used in the mapping process will help to reduce

the generation of duplicate concepts and therefore improve the mapping accuracy.

- Secondly, the results showed that increasing the number of tags increased the percentage of fully mapped concepts. However, as the concepts to devices ratio increased, the percentage of fully mapped concepts also decreased, albeit there was only a drop of 10% as the concepts to devices ratio increased tenfold from 1:5 (less real-life-like) to 1:50 (more real-life-like).
- A similar trend was observed for the *average concepts returned* statistic. Increasing the number of tags increased the average concepts returned. However, as the concepts to devices ratio increased, the number of average concepts returned also increased in response. This is expected because having a higher number of devices in the knowledgebase increases the likelihood of getting more matches (regardless of whether the actual provided mappings are accurate or not). Translating these results into a real-life application, this shows that with a large concepts to devices ratio, the system is expected to return at least 1 correct or incorrect mapping when any number of tags are used to map the device to an appropriate semantic concept. With the usage of 3 tags, the system is expected to return at least 2 concepts for each mapping.
- The aforementioned trend was broken in the comparison of the final statistic: *cases with one concept having bigger weight than the rest*. For a concepts to devices ratio of 1:5, the trend was that the ambiguity increased as the number of tags was increased. However, in all other

simulations, the trend was the opposite: the ambiguity decreased as the number of tags was increased. In all simulation scenarios, when 3 tags were used, the ambiguity hovered around the 10% mark (the percentage of cases with one concept having bigger weight than the rest was around 90%). This showed that as the concepts to devices ratio increased, usage of 1 tag increased the ambiguity and created difficulty for autonomous agents. It also showed that usage of 3 tags reduced the ambiguity to the 10% mark regardless of the concepts to devices ratio.

The results obtained from vigorous testing and a critical analysis of the performance metrics reveal that SAW is fit for the purpose it was designed for, and is successful in achieving both syntactic as well as semantic interoperability.

8.1.3 Summary of key contributions

SAW primarily contributes 3 main systems that help to produce an overall distributed and collaboration system for the WoT domain:

1. Resource-based asset model:
    a. Provides the capability to represent assets at different levels of granularity;
    b. Provides a logical data hierarchy;
    c. Provides generic and extensible data templates.
2. CPPM-TBAC – A resource-based access control mechanism:
    a. Allows distributed access to resources of any granularity;
    b. Scales efficiently for large number of resources without projecting a noticeable impact on network performance.

The extensive set of tests carried out and the results recorded in relation to the asset model also present a baseline for future studies to compare against for further work in this area. Existing studies were lacking this statistical analysis into existing access control mechanisms and their impact on the added delay as the number of resources increased in a network.

3. Service-oriented and semantic interaction model: Enabling the capability to semi-autonomously profile and annotate resources from external networks such as Xively so that resources which are already published on the web but lack semantics can be used effectively.

Once again, the comprehensive set of tests carried out and the metrics measured present a springboard for future studies to compare their proposed mechanisms against. Currently, no existing studies present statistical measures of semantic annotation mechanisms which makes it difficult to compare new methodologies in terms of their effectiveness. It is hoped that the findings of this study form this much needed baseline.

8.2 Future Work

There are various areas in the proposed asset model, the CPPM-TBAC and the tag-based semantic annotation mechanisms that can be improved to achieve better performance metrics and to also extend the underlying capabilities of the SAW framework. These improvements and further areas of potential research are discussed below.

8.2.1 Potential improvements and future work for the asset model

The following improvement is a noteworthy future undertaking for the asset model:

- Extend the asset model to support circular relationships. This will enable the asset data templates to have DF within DF. This will be useful for modelling and representing coarser devices.

An example of this is a laptop which can have built-in sensors. The laptop can also have other multi-sensor platforms attached to it. With the current asset model data hierarchy, the attached multi-sensor platforms would be modelled as DF and their sensors as DS. This presents a problem when representing the laptop because it would be modelled as a DF and the attached multi-sensor platforms as DS which is semantically incorrect. With circular relationships, the laptop can be modelled as a DF containing other DF (the multi-sensor platforms).

8.2.2 Potential improvements and future work for the CPPM-TBAC

The main improvements and areas of further work in terms of the CPPM-TBAC mechanism are the following:

- Enable overwriting of extended access restrictions in the local scope: Currently, extended access restrictions (e.g. IP restrictions, API invocation limits, token expiration, etc.) are provided in the global scope, so they apply to all resources that the token applies to. There is no way to refine the scope of the global access restrictions for particular resources within the same token. View section 3.4.3.3 for more details about this feature and the recommendation to enable overwriting of access restrictions in the local scope.
- Investigate methods of improving security for the proposed token-based access control mechanism. In the proposed CPPM-TBAC, the production of a valid token is all that is required to access the

corresponding resources. There is no need to login, so there is no authentication process (who does the token belong to, although the extended access restrictions can be used to limit token to certain source IPs), only an authorisation process (what the token can access). In contrast, RBAC would typically have a two-step authentication procedure (provide a *username/email* and a *password/secret*) as well as an authorisation feature (roles). This is a potential area of future work that can be investigated to improve the security of the SAW framework.

8.2.3 Potential improvements and future work for the tag-based semantic annotation mechanism

Finally, it is believed that the following list of improvements will help in extending the capability of the semantic annotation mechanism and increasing its performance:

- Create persistent weights/rankings for secondary tags so that system operators can promote oft-used secondary tags to primary tags. Primary tags have a higher weighting than secondary tags and represent a higher likelihood of the tag accurately representing the resource being annotated. Over time, the knowledgebase may consist of secondary tags that are as non-ambiguous as primary tags in their relation to the resources being annotated, but there is currently no way to promote these high-quality secondary tags to primary tags. On the surface, this seems to be a good mechanism of increasing the accuracy of the annotation mechanism, but thorough analysis after its implementation will be required to evaluate the effectiveness of this feature.

- Community-based semantic annotation mechanism: Section 3.4.5.2 discusses and proposes an indirect and community-based semantic annotation mechanism as a complementary annotation mechanism to the tag-based one. The main advantages of this complementary mechanism are believed to be the ability to flag incorrect annotations, and the ability to contribute relevant semantic annotations for existing resources. Both of these have the potential to improve the accuracy of the annotation process and lead to a further enrichment of the knowledgebase.
- Investigate the usage of measurement ontologies. Currently SAW is based on the SSN ontology which is very effective in semantically annotating properties of sensing devices. The SSN ontology also lays the foundations for using specialised measurement ontologies alongside it to provide semantic concepts defining the measurement characteristics of sensing devices and data.

Another item for future work is the utilisation of cloud computing for hosting the SAW framework in order to dynamically allocate the necessary computing resources such as Central Processing Unit (CPU), Random Access memory (RAM), hard-disk space and network bandwidth. It is anticipated that this will dramatically increase the performance of the framework since computing limitations in the underlying infrastructure can effectively be eliminated by utilising the elastic scaling capabilities of cloud computing. Adoption of cloud computing can also lead to higher uptime of the system and therefore increase the utilisation of the framework and provide more useful collaboration facilities with other online system.

References

[1] S. Vieweg, A. L. Hughes, K. Starbird and L. Palen, “Microblogging during two natural hazards events: What twitter may contribute to situational awareness,” in *SIGCHI Conference on Human Factors in Computing Systems*, New York, 2010.

[2] B. Rubenstein-Montano, J. Liebowitz, J. Buchwalter, D. McCaw, B. Newman and K. Rebeck, “A systems thinking framework for knowledge management,” *Decision Support Systems,* vol. 31, no. 1, pp. 5-16, 2001.

[3] M. Dorasamy, M. Raman and M. Kaliannan, “Knowledge management systems in support of disasters management: A two decade review,” *Technological Forecasting & Social Change,* 2013.

[4] S. Jennings, “Time’s Bitter Flood - Trends in the number of reported natural disasters,” OXFAM, 2011.

[5] P. Peduzzi, “Is climate change increasing the frequency of hazardous events?,” in *Environment & Poverty Times*, Kobe, 2005.

[6] UN/ISDR, “Terminology - UNISDR,” 23 August 2007. [Online]. Available: http://www.unisdr.org/we/inform/terminology. [Accessed 27 January 2012].

[7] R. W. Perry and M. K. Lindell, “Preparedness for Emergency Response - Guidelines for the Emergency Planning Process,” *Disasters,* vol. 27, no. 4, pp. 336-350, 2003.

[8] G. O'Brien, “UK emergency preparedness: a holistic response?,” *Disaster Prevention and Management,* vol. 17, no. 2, pp. 232-243, 2008.

[9] I. Davis and Y. O. Izadkhah, “Tsunami early warning system (EWS) and its integration within the chain of seismic safety,” *Disaster Prevention and Management,* vol. 17, no. 2, pp. 281-291, 2008.

[10] M. L. Collins and N. Kapucu, “Early warning systems and disaster preparedness and response in local government,” *Disaster Prevention and Management,* vol. 17, no. 5, pp. 587-600, 2008.

[11] Geographical Survey Institute Japan, “Disaster Prevention Activities,” in *Eighteenth United Nations Regional Cartographic*, Bangkok, 2009.

[12] D. C. Whybark, “Issues in managing disaster relief inventories,” *International Journal of Production Economics,* vol. 108, no. 1-2, pp. 228-235, 2007.

[13] M. J. Widenera and M. W. Horner, “A hierarchical approach to modeling hurricane disaster relief goods distribution,” *Journal of Transport Geography,* vol. 19, no. 4, p. 821–828, 2011.

[14] E. E. Ozguven and K. Ozbay, “A secure and efficient inventory management system for disasters,” *Transportation Research Part C,* 2011.

[15] H. R. Rao, V. S. Jacob and F. Lin, “Hemispheric Specialization, Cognitive Differences, and Their Implications for the Design of Decision Support Systems,” *MIS Quarterly,* vol. 16, no. 2, pp. 145-151 , 1992.

[16] S. Celik and S. Corbacioglu, “Role of information in collective action in dynamic disaster environments,” *Disasters,* vol. 34, no. 1, p. 137−154, 2010.

[17] B. Balcik, B. M. Beamon, C. C. Krejci, K. M. Muramatsu and M. Ramirez, “Coordination in humanitarian relief chains: Practices, challenges and opportunities,” *International Journal of Production Economics,* vol. 126, no. 1, pp. 22-34, 2010.

[18] J. Max Stephenson, “Making humanitarian relief networks more effective: operational coordination, trust and sense making,” *Disasters,* vol. 29, no. 4, pp. 337-350, 2005.

[19] M. S. Jr. and M. H. Schnitzer, “Inter-Organizational Trust, Boundary Spanning and Humanitarian Relief Coordination,” *Nonprofit Management and Leadership,* vol. 17, no. 2, pp. 211-232, 2006.

[20] P. Barnaghi et al., “Semantics for the Internet of Things: Early Progress and Back to the Future,” *International Journal On Semantic Web and Information Systems,* vol. 8, no. 1, pp. 1-21, 2012.

[21] P. Barnaghi et al, “Sense and Sens’ability: Semantic Data Modelling for Sensor,” in *ICT Mobile Summit 2009*, Santander, Spain, 2009.

[22] S. Auer et al, “DBpedia: A Nucleus for a Web of Open Data,” in *6th Int’l Semantic Web Conference*, Busan, Korea, 2007.

[23] Guardian News and Media Limited, “Flood hack: UK's top developers join forces to build flood-relief apps,” 18 February 2014. [Online]. Available: http://www.theguardian.com/technology/2014/feb/17/uk-flood-relief-apps-hack-day. [Accessed 19 February 2014].

[24] M. Turoff et al., “The design of a dynamic emergency response management information system (DERMIS),” *Journal of Information Technology Theory and Application,* vol. 5, no. 4, pp. 1-35, 2004.

[25] R. Bose, “Knowledge management-enabled health care management systems: capabilities, infrastructure, and decision-support,” *Expert Systems with Applications,* vol. 24, no. 1, p. 59–71, 2003.

[26] D. Stenmark and R. Lindgren, “Knowledge Management Systems: Towards a Theory of Integrated Support,” in *Current Issues in Knowledge Management*, IGI Global, 2008, pp. 181-205.

[27] M. Turoff, C. White and L. Plotnick, “Dynamic Emergency Response

Management for Large Scale Decision Making in Extreme Hazardous Events," in *5th International ISCRAM Conference*, Washington, 2011.

[28] M. Jennex, "Emergency response systems: lessons from utilities and Y2K," in *Tenth Americas Conference on Information*, New York, 2004.

[29] E. Iakovou and C. Douligeris , "An information management system for the emergency management of hurricane disasters," *International Journal of Risk Assessment and Management,* vol. 2, no. (3-4) 2001, pp. 243-262, 2004.

[30] T. Murphy and M. E. Jennex, "Knowledge Management, Emergency Response, and Hurricane Katrina," *International Journal of Intelligent Control and Systems,* vol. 11, no. 4, pp. 199-208, 2006.

[31] M. Dorasamy and M. Raman, "Information Systems to Support Disaster Planning and Response: Problem Diagnosis and Research Gap Analysis," in *8th International ISCRAM Conference*, Lisbon, 2011.

[32] L. V. S. Lakshmanan and F. Sadri, "Information Integration and the Semantic Web," *IEEE Data Engineering Bulletin,* vol. 26, no. 4, pp. 19-25, 2003.

[33] M. Botts et. al, "OGC Sensor Web Enablement: Overview And High Level Architecture," Open Geospatial Consortium, 2013.

[34] A. Sheth, C. Henson and S. S. Sahoo, "Semantic Sensor Web," *IEEE Internet Computing,* vol. 12, no. 4, pp. 78 - 83, 2008.

[35] W3C, "Resource Description Framework (RDF): Concepts and Abstract Syntax," 10 February 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. [Accessed 25 September 2013].

[36] A. Gyrard, C. Bonnet and K. Boudaoud, "A Machine-to-Machine Architecture to Merge Semantic Sensor Measurements," in *22nd International World Wide Web Conference*, Rio de Janeiro, Brazil, 2013.

[37] W3C, "SPARQL 1.1 Query Language," 21 March 2013. [Online]. Available: http://www.w3.org/TR/sparql11-query/. [Accessed 25 September 2013].

[38] L. Feigenbaum and E. Prud'hommeaux, "SPARQL by Example," 30 05 2013. [Online]. Available: http://www.cambridgesemantics.com/en_GB/semantic-university/sparql-by-example. [Accessed 13 05 2014].

[39] W3C, "OWL 2 Web Ontology Language Document Overview (Second Edition)," 11 December 2012. [Online]. Available: http://www.w3.org/TR/owl2-overview/. [Accessed 25 September 2013].

[40] M. Compton et al., "The SSN ontology of the W3C semantic sensor

network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 17, p. 25–32, 2012.

[41] D. Brickley and L. Miller, "FOAF Vocabulary Specification 0.99," 14 January 2014. [Online]. Available: http://xmlns.com/foaf/spec/. [Accessed 19 May 2014].

[42] A. Gangemi, "DOLCE+DnS Ultralite," [Online]. Available: http://www.ontologydesignpatterns.org/ont/dul/DUL.owl. [Accessed 19 May 2014].

[43] V. Mascardi, V. Cordì and P. Rosso, "A Comparison of Upper Ontologies (Technical Report DISI-TR-06-21)," Università degli Studi di Genova, Genova, Italy, 2007.

[44] N. Bharosa and M. Janssen, "Extracting principles for information management adaptability during crisis response: A dynamic capability view," in *Proceedings of the 43rd Hawaii International Conference on System Sciences*, Hawaii, USA, 2010.

[45] SENSEI, "The SENSEI Real World Internet Architecture," March 2010. [Online]. Available: http://www.sensei-project.eu/index.php?option=com_docman&task=doc_download&gid=83&Itemid=49. [Accessed 25 September 2013].

[46] M. Büscher, P. H. Mogensen and M. Kristensen, "When and how (not) to trust it? Supporting virtual emergency teamwork," *International Journal of Information Systems for Crisis Response and Management (IJISCRAM),* vol. 1, no. 2, pp. 1-15, 2009.

[47] C. Caragea et al., "Classifying Text Messages for the Haiti Earthquake," in *Proceedings of the 8th International ISCRAM Conference*, Lisbon, Portugal, 2011.

[48] H. G. Bressler, E. M. Jennex and G. E. Frost, "Exercise 24: Using social media for crisis response," *International Journal of Information Systems for Crisis Response and Management (IJISCRAM),* vol. 3, no. 4, pp. 36-54, 2011.

[49] I. Becerra-Fernández et al., "Design and Development of a Virtual Emergency Operations Center for Disaster Management Research, Training, and Discovery," in *Proceedings of the 41st Hawaii International Conference on System Sciences*, Hawaii, USA, 2008.

[50] G. Wickler et al., "The Virtual Collaboration Environment: New Media for Crisis Response," in *Proceedings of the 8th International ISCRAM ConferenceLisbon, Portugal*, Lisbon, Portugal, 2011.

[51] R. A. Trancoso et al., "Early warning system for meteorological risk in Lisbon Municipality: description and quality evaluation," in *Proceedings of the 8th International ISCRAM*, Lisbon, Portugal, 2011.

[52] D. Guinard et al., "From the Internet of Things to the Web of Things:

Resource Oriented Architecture and Best Practices," in *Architecting the Internet of Things*, Springer Berlin Heidelberg, 2011, pp. 97-129.

[53] Microformats, "Microformats," [Online]. Available: http://microformats.org/. [Accessed 24 September 2013].

[54] "Project Deliverable D1.2 – Initial Architectural Reference Model for IoT," IoT-A, 2011.

[55] LogMeIn, Inc., "Xively REST API," [Online]. Available: https://xively.com/dev/docs/api/. [Accessed 20 September 2013].

[56] ThingSpeak.com, "ThingSpeak API," [Online]. Available: http://community.thingspeak.com/documentation/api/. [Accessed 20 September 2013].

[57] A. Pintus, D. Carboni and A. Piras, "Paraimpu: a Platform for a Social Web of Things," in *Companion of WWW2012 - Demonstrations*, Lyon, France, 2012.

[58] N. Chen and A. Dahanayake, "Role-based Situation-aware Information Seeking and Retrieval for Crisis Response," *Internation Journal of Intelligent Control and Systems,* vol. 12, no. 2, pp. 186-197, 2007.

[59] J. B. Vercher et al., "An experimental platform for large-scale research facing FI-IoT scenarios," in *Future Network & Mobile Summit (FutureNetw)*, Warsaw, 2011.

[60] H. Patni, C. Henson and A. Sheth, "Linked Sensor Data," in *2010 International Symposium on Collaborative Technologies and Systems (CTS 2010)*, Chicago, IL, 2010.

[61] OGC, "Observations and Measurements," [Online]. Available: http://www.opengeospatial.org/standards/om. [Accessed 20 September 2013].

[62] P. Barnaghi, M. Presser and K. Moessner, "Publishing Linked Sensor Data," in *ISWC 2010*, Shanghai, China, 2010.

[63] D. Le Phuoc, ""SensorMasher: publishing and building mashup of sensor data," in *5th International Conference on Semantic Systems (I-Semantics 2009)*, 2009.

[64] NASA, "SWEET Ontologies," 6 August 2012. [Online]. Available: http://sweet.jpl.nasa.gov/. [Accessed 25 September 2013].

[65] "SANY Sensor Taxonomy," [Online]. Available: http://sany-ip.eu/publications/1954. [Accessed 25 September 2013].

[66] D. Pfisterer et al., "SPITFIRE: toward a semantic web of things," *IEEE Communications Magazine,* vol. 49, no. 11, pp. 40-48, 2011.

[67] OASIS, "Security Assertion Markup Language (SAML) V2.0 Technical Overview," 2008.

[68] J. Li and A. H. Karp, “Access Control for the Services Oriented Architecture,” in *Proceedings of the 2007 ACM workshop on Secure web services*, New York, NY, USA, 2007.

[69] A. H. Karp, “Authorization-Based Access Control for the Services Oriented Architecture,” in *4th International Conference on Creating, Connecting, and Collaborating through Computing (C5)*, 2006.

[70] S. Gusmeroli, . S. Piccione and . D. Rotondi, “A capability-based security approach to manage access control in the Internet of Things,” *Mathematical and Computer,* 2013.

[71] I. Aedo, P. Díaz and D. Sanz, “An RBAC model-based approach to specify the access policies of Web-based emergency information systems,” *International Journal of Intelligent Control and Systems,* vol. 11, no. 4, pp. 272-283 , 2006.

[72] H. Scholten, “Context Based Access Control,” Everett BV, Nieuwegein, The Netherlands, 2007.

[73] “A Token-Based Access Control System for RDF Data in the Clouds,” in *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, Washington, DC, USA, 2010.

[74] uh@pachube, “Crowd-sourced realtime radiation monitoring in Japan,” Cosm Ltd, 24 March 2011. [Online]. Available: http://community.cosm.com/node/611. [Accessed 6 March 2013].

[75] D. O'Byrne, R. Brennan and D. O'Sullivan, “Implementing the draft W3C semantic sensor network ontology,” in *8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, Mannheim, 2010.

[76] J. C. Goodwin, J. Qualls and D. J. Russomanno, “Survey of Semantic Extensions to UDDI: Implications for Sensor Services,” in *International Conference on Semantic Web & Web Services (SWWS)*, Las Vegas, Nevada, USA, 2007.

[77] A. Underbrink et al., “Autonomous Mission Operations for Sensor Webs,” in *American Geophysical Union Fall Meeting*, 2008.

[78] M. Eid, R. Liscano and A. El Saddik, “A Universal Ontology for Sensor Networks Data,” in *Computational Intelligence for Measurement Systems and Applications (CIMSA)*, Ostuni, 2007.

[79] M. Calder, R. A. Morris and F. Peri, “Machine reasoning about anomalous sensor data,” *Ecological Informatics,* vol. 5, no. 1, p. 9–18, 2010.

[80] Y. Hu, Z. Wu and M. Guo, “Ontology driven adaptive data processing in wireless sensor networks,” in *2nd international conference on Scalable information systems (InfoScale '07)*, Brussels, Belgium,

2007.

[81] G. Stevenson et al., “Ontonym: a collection of upper ontologies for developing pervasive systems,” in *1st Workshop on Context, Information and Ontologies (CIAO '09)*, New York, NY, USA, 2009.

[82] F. Probst, “Semantic reference systems for observations and measurements,” Westfälische Wilhelms - Universität Münster, 2007.

[83] C. Stasch et al., “A Stimulus-Centric Algebraic Approach to Sensors and Observations,” in *GeoSensor Networks (GSN) Third International Conference*, Oxford, UK, 2009.

[84] P. Barnaghi, S. Meissner and M. Presser, “Sense and sensability: Semantic data modelling for sensor networks,” in *ICT Mobile Summit*, Santander, Spain, 2009.

[85] C. A. Henson et al., “SemSOS: Semantic sensor Observation Service,” in *International Symposium on Collaborative Technologies and Systems (CTS '09)*, Baltimore, MD , 2009.

[86] B. van der Werf et al., “SERONTO: a Socio-Ecological Research and Observation oNTOlogy,” in *TDWG*, Fremantle, Australia, 2008.

[87] W3C Incubator Group, “Semantic Sensor Network XG Final Report,” 28 June 2011. [Online]. Available: http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/. [Accessed 6 October 2013].

[88] K. Janowicz and M. Compton, “The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology,” in *3rd International Workshop on Semantic Sensor Networks*, 2010.

[89] D. Guinard and V. Trifa, “Towards the Web of Things: Web Mashups for Embedded Devices,” in *2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009)*, Madrid, Spain, 2009.

[90] R. T. Fielding and R. N. Taylor, “Principled design of the modern Web architecture,” *ACM Transactions on Internet Technology (TOIT),* vol. 2, no. 2, pp. 115-150, 2002.

[91] G. Iachello and G. D. Abowd , “A Token-based Access Control Mechanism for Automated Capture and Access Systems in Ubiquitous Computing,” Georgia Institute of Technology, Atlanta, GA, USA, 2005.

[92] D. Miorandi, S. Sicari, F. D. Pellegrini and I. Chlamtac, “Internet of things: Vision, applications and research challenges,” *Ad Hoc Networks,* vol. 10, no. 7, pp. 1497-1516, 2012.

[93] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess and D. Savio, “Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services,” *IEEE*

TRANSACTIONS ON SERVICES COMPUTING, vol. 3, no. 3, pp. 223-235, 2010.

[94] G. A. Miller, “WordNet: A Lexical Database for English,” *Communications of the ACM,* pp. 39-41, 1995.

[95] Princeton University, “About WordNet,” Princeton University, [Online]. Available: https://wordnet.princeton.edu/. [Accessed 27 April 2015].

[96] Cycorp Inc., “ResearchCyc,” Cycorp Inc., [Online]. Available: http://vps9304.inmotionhosting.com/node/184.html. [Accessed 27 April 2015].

[97] A. Zaslavsky, . C. Perera and D. Georgakopoulos, “Sensing as a Service and Big Data,” in *International Conference on Advances in Cloud Computing (ACC)*, Bangalore, India, 2012.

[98] M. Kuna, H. Kolaric, I. Bojic, M. Kusek and G. Jezic, “Android/OSGi-based Machine-to-Machine context-aware system,” in *IEEE 11th International Conference on Telecommunications (ConTEL)*, Graz, 2011.

[99] R. Bhaskar, G. Hegde and P. Vaya, “An efficient hardware model for RSA Encryption system using Vedic mathematics,” in *International Conference on Communication Technology and System Design*, Coimbatore, 2011.

Appendices

Appendix A – SAW Ontology

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="owl2html.xslt"?>
<!DOCTYPE rdf:RDF [<!ENTITY dct "http://purl.org/dc/terms/"
><!ENTITY cc "http://creativecommons.org/ns#" ><!ENTITY owl
"http://www.w3.org/2002/07/owl#" ><!ENTITY dc
"http://purl.org/dc/elements/1.1/" ><!ENTITY xsd
"http://www.w3.org/2001/XMLSchema#" ><!ENTITY ssn
"http://purl.oclc.org/NET/ssnx/ssn#" ><!ENTITY skos
"http://www.w3.org/2004/02/skos/core#" ><!ENTITY rdfs
"http://www.w3.org/2000/01/rdf-schema#" ><!ENTITY DUL
"http://www.loa-cnr.it/ontologies/DUL.owl#" ><!ENTITY rdf
"http://www.w3.org/1999/02/22-rdf-syntax-ns#" >]>

<rdf:RDF xmlns="http://saw.local/sw/ontology#"
xml:base="http://saw.local/sw/ontology"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:DUL="http://www.loa-cnr.it/ontologies/DUL.owl#"
xmlns:dct="http://purl.org/dc/terms/"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:ssn="http://purl.oclc.org/NET/ssnx/ssn#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:skos=http://www.w3.org/2004/02/skos/core#
xmlns:cc="http://creativecommons.org/ns#">

    <owl:Ontology rdf:about="http://saw.local/sw/ontology">
        <dc:creator rdf:datatype="&xsd;string">Mohammad
Amir</dc:creator>
        <rdfs:comment rdf:datatype="&xsd;string">Describes concepts
for tagging sensing devices.</rdfs:comment>
        <rdfs:comment rdf:datatype="&xsd;string">Developed by
Mohammad Amir, University of Bradford.</rdfs:comment>
        <dc:identifier>http://saw.local/sw/ontology</dc:identifier>
        <dc:rights>Copyright 2013 University of
Bradford.</dc:rights>
        <dct:created>2013-10-14</dct:created>
        <dct:modified>2013-10-14</dct:modified>
        <rdfs:seeAlso>http://saw.local/sw/ontology</rdfs:seeAlso>
        <dc:title>SAW Ontology</dc:title>
        <owl:imports
rdf:resource="http://purl.oclc.org/NET/ssnx/ssn"/>
        <cc:license
rdf:resource="http://www.w3.org/Consortium/Legal/2002/copyright-
software-20021231.html"/>
    </owl:Ontology>
```