Symbiosis of smart objects across IoT environments

688156 - symbIoTe - H2020-ICT-2015

D2.1 – Semantics for IoT and Cloud resources

The symbIoTe Consortium

Intracom SA Telecom Solutions, ICOM, Greece
Sveučiliste u Zagrebu Fakultet elektrotehnike i računarstva, UNIZG-FER, Croatia
AIT Austrian Institute of Technology GmbH, AIT, Austria
Nextworks Srl, NXW, Italy
Consorzio Nazionale Interuniversitario per le Telecomunicazioni, CNIT, Italy
ATOS Spain SA, ATOS, Spain
University of Vienna, Faculty of Computer Science, UNIVIE, Austria
Unidata S.p.A., UNIDATA, Italy
Sensing & Control System S.L., S&C, Spain
Fraunhofer IOSB, IOSB, Germany
Ubiwhere, Lda, UW, Portugal
VIPnet, d.o.o, VIP, Croatia
Instytut Chemii Bioorganicznej Polskiej Akademii Nauk, PSNC, Poland
NA.VI.GO. SCARL, NAVIGO, Italy

For more information on this document or the symbIoTe project, please contact:
Sergios Soursos, INTRACOM TELECOM, souse@intracom-telecom.com

Document Control

Title: Semantics for IoT and Cloud resources

Type: Public

Editor(s): Michael Jacoby (IOSB)

E-mail: michael.jacoby@iosb.fraunhofer.de

Author(s): Aleksandar Antonić (UNIZG-FER), Gerhard Dünnebeil (AIT), Reinhard Herzog (IOSB), Michael Jacoby (IOSB), Roman Łapacz (PSNC), Matteo Pardi (NXW), Svenja Schröder (UNIVIE), Ivana Podnar Žarko (UNIZG-FER)

Doc ID: D2.1-v1.3.docx

Amendment History

Version	Date	Author	Description/Comments
v0.0	11/04/2016	Michael Jacoby (IOSB)	Table of Contents
v0.1	08/09/2016	Michael Jacoby (IOSB)	Updated Table of Contents
v0.2	06/10/2016	Michael Jacoby (IOSB)	Updated ToC, added content to Section 3, 4, 5
v0.3	13/10/2016	Gerhard Dünnebeil (AIT)	Added a first draft of openUwedat's data model
		Svenja Schröder (UNIVIE)	Updated Sections 2.2, 2.3 and 2.4
v0.4		Gerhard Dünnebeil (AIT)	Updated Section 3.4.1
		Svenja Schröder (UNIVIE)	Added Section 8
		Reinhard Herzog (IOSB)	Added Section 1
		Michael Jacoby (IOSB)	Added Table of Figures, Section 5.2, Section 5.4, Section 6; updated Section 2, Section 3.2.1, Section 5; added references for Section 3.3
v0.5	25/10/2016	Reinhard Herzog (IOSB)	Updated Section 1
		Matteo Pardi (NXW)	Added Section 3.4.2
		Michael Jacoby (IOSB)	Added Section 3.4
		Aleksandar Antonić (UNIZG-FER)	Added Section 5.3, Section 5.5
		Roman Łapacz (PSNC)	Added Section 4.3
v1.0	28/10/2016	Matteo Pardi (NXW)	Updated Section 3.4.2
		Michael Jacoby (IOSB)	Updated figures in Section 3.4.2, Updated all sections (incorporated feedback)
v1.1	07.11.2016	Aleksandar Antonić (UNIZG-FER)	Updated Section 5.5
		Ivana Podnar Žarko (UNIZG-FER)	Updated all sections
		Gerhard Dünnebeil (AIT)	Updated Section 3.5.1
		Reinhard Herzog (IOSB)	Updated Section 1
		Michael Jacoby (IOSB)	Updated all Sections
v1.2	23.11.2016	Ivana Podnar Žarko (UNIZG-FER)	Updated all sections
		Michael Jacoby (IOSB)	Updated Section 5, Section 6
v1.3	29.11.2016	Sergios Soursos (ICOM)	Final Editing

Legal Notices

The information in this document is subject to change without notice.
The Members of the symbIoTe Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the symbIoTe Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Table of Contents

Table of Figures

Table of Tables

1 Executive Summary

One main objective of the symbIoTe project is to create a mediation framework to enable the discovery and sharing of connected devices across existing and future IoT platforms to enable platform federation and rapid development of cross-platform IoT applications. The problem is that these devices are managed by different Internet of Things (IoT) platforms, which are designed for different application domains. The requirement to enable information technology to deal with the semantic of data has been one of the grand challenges in the computer science domain in the past and probably will be one for the next future. Having that in mind, this document will not try to solve all open issues, but it will lay out a practical approach to the given task. Deliverable D2.1 is the first iteration of this deliverable with the purpose to analyse the general problem and to define the basic approach to be used by symbIoTe. The second and final version of this deliverable is scheduled for month 18.

To understand the motivation of this document one may imagine a typical IoT platform for any given domain, for example, a climate control system for a smart home. Such a platform deals with data in a given context where the meaning is predefined, e.g., the scale of a thermostat ranging from 0 to 5, where 0 means *no heating* and 5 *valve is open*. It also has implicit data models like the location of a radiator, which relates to a room and maybe to a heating circuit. Another IoT platform could manage for example self-monitoring devices which collect data within a different context and different predefined meanings of temperature and locations. To make such different IoT-platforms understand each other, the meaning of data and concepts must be explicitly defined. The background chapter on semantic interoperability introduces possible technical solutions for this task. With the developments influenced by the so called ‘Semantic Web’ there are established standards, methods and tools available, like the RDF and OWL format to describe the semantics of data which will be used by symbIoTe. The document also describes the achievements in the development of semantic mapping, to translate one information model into another which is semantically similar but structurally different. This capability is required for enabling interoperability between platforms which are based on individual data models.

There are several ways to achieve semantic interoperability. Section 4 explains and discusses the possible approaches, from a simple approach where everybody shares the same understanding, to the most complex one where everybody may use different concepts and interpretations. From a practical perspective, neither the easy nor the complex approach is feasible for most real-life applications. symbIoTe proposes an approach where it starts from a set of basic concepts, which are common for all platforms connected via the symbIoTe framework. These basic concepts are sufficient to provide “meta”-understanding about the connected IoT platforms and their resources, so that symbIoTe can provide a generic interoperable mediation service. To cover the actual meaning of platform-specific data, more detailed platform specific concepts are required. symbIoTe thus proposes an approach which allows multiple extensions to the basic concepts and aims to provide semantic and syntactic transformation as a common interoperability service.

This deliverable provides an initial proposal how the approach to semantic interoperability presented in this document can be incorporated into the symbIoTe architecture presented in D1.2. Further refinements on this will be included in the final deliverable D2.4.

2 Introduction

In the context of this document, semantic can be understood as the meaning of things. Its main purpose in symbIoTe is to enable interoperability, especially semantic interoperability which “is the ability of computer systems to exchange data with unambiguous, shared meaning” [1] (see Section 3.1 for a detailed definition of semantic interoperability). Some parts of this deliverable are based on the unpublished paper “Semantic Interoperability as Key to IoT Platform Federation” [2] (accepted at the 2nd International Workshop on Interoperability & Open Source Solutions for the Internet of Things) as well as the paper “Semantic interoperability in IoT-based automation infrastructures” [3], both of which were authored as part of the academic dissemination within symbIoTe.

2.1 Semantics in symbIoTe

As the overall objective of symbIoTe is to create an interoperability framework for IoT platforms and to enable platform federation, achieving semantics is a key challenge. symbIoTe plans to use semantic technologies to bridge the semantic gap between existing and future IoT platforms with a goal to enable interoperability on a higher level, the semantic level, which is a step forward in comparison to state of the art solutions which primarily focus on syntactic interoperability.

2.2 Purpose of the document

The purpose of deliverable D2.1 “Semantics for IoT and Cloud resources” is to document the first outcomes and intermediate results regarding semantics in the symbIoTe project. The role of semantics within symbIoTe is described and approaches to achieve semantic interoperability are identified and discussed. Implementation plans for semantics in symbIoTe and how they’re represented in the current architecture are specified in this document.

This document will serve as a basis for the revised version of the semantics, which will be presented in deliverable D2.4 “Revised Semantics for IoT and Cloud Resources” in month 18.

2.3 Document Scope

This document reports on the work accomplished in T2.1 “Semantics for IoT and Cloud Resources” with a strong focus on semantic interoperability. It presents a theoretical analysis of the problem domain of semantic interoperability together with multiple possible approaches to address this problem. Furthermore, it discusses which of these approaches be used within symbIoTe and presents ideas how this approach can be realized.

Although mentioned in some parts, syntactic interoperability and how it is addressed and realized by symbIoTe is not the primary subject to this deliverable.

2.4 Relation to other deliverables

The content of this deliverable was motivated and influenced by the intermediate outcome of tasks T1.3 “System requirements” and T1.4 “System architecture” which has been published in D1.2 “Initial Report on System Requirements and Architecture”.

This deliverable will have impact on multiple tasks as semantic interoperability is a core functionality of symbIoTe. The need for new functionality and components realizing semantic interoperability will have a direct influence on T1.4 “System architecture”, T2.2 “Virtual IoT environment” and T3.3 “Specification & Implementation of IoT Federation”. Furthermore, T2.3 “Implementation of symbIoTe domain-specific enablers”, T3.2 “Security and Access Scopes” and T4.1 “Local Registration, Discovery and Interoperability of Smart Objects” will be influenced indirectly by the outcome of this deliverable.

2.5 Deliverable Outline

This deliverable is structured as follows: Section 3 introduces the term semantic interoperability and provides background information on current semantic web technologies and standards. Furthermore, the information models of a subset of existing platforms that will be used in symbIoTe are presented to get a better understanding of the current situation and the difficulty of the problem of semantic interoperability. In Section 4 the problem of achieving semantic interoperability between multiple IoT platforms is presented in detail, together with a number of possible approaches how to address it. Section 5 presents which of the presented approaches the symbIoTe consortium has decided to follow and sketches the current plans to implement the chosen approach. This section also presents details on how this will influence the architecture presented in D1.2. The document closes with Section 6 to present conclusions and next steps.

3 Background

3.1 What are semantics and why do we need it?

According to the Merriam-Webster dictionary, semantics is the study of meaning, especially "the meaning or relationship of meanings of a sign or set of signs" [4]. A sign here is a fundamental linguistic unit that designates an object or relation and therefore semantics can be understood as the mapping of signs to their meaning. For example, a sign could be a word, e.g. the word 'table', or a URI (Uniform Resource Identifier) like 'http://www.example.com/table' and the corresponding meaning would be the concept of a table that is defined by its properties and context specifying that it is a constructed thing which has some legs, a solid top and is normally found in a man-made environment. In every act of communication, the involved actors, human or computers, require a shared understanding of things. A typical way to express semantics is via a taxonomy or an ontology which is "an explicit specification of a [shared] conceptualization" [5] (explicit because all relevant elements must be explicitly named in order to avoid misinterpretations; shared because there must be a common agreement within a specific domain of interest).

Knowing this, it is obvious that semantics plays an important role in every act of communication. Without it, we would not be able to understand the meaning of the exchanged information, or at least we could not be sure that we are having the same understanding of it as our communication partner. To give an example image two people having a conversation about the weather and one says "…it had 40 degrees outside". In this case the semantics are not clearly defined as degrees could refer to degrees Fahrenheit or to degrees Celsius. Communication might work by chance if they both have the same understanding of degree in the context of temperature but as long as they don't use a shared or agreed upon vocabulary they cannot be sure that they both have the same understanding of the exchanged information.

Transferring this example to the technical level of the IoT platforms communicating and exchanging data where the information is exchanged, processed and interpreted by machines this clear and formal definition of meaning is even more important because machines (most of the time) do not have any capability of reasoning to come up with a probably correct interpretation of the meaning of received data from others (mainly because they don't have the information they need for this task, in the example this would be the cultural background of the person speaking). Therefore, semantics is essential to bring multiple IoT platforms which were created by different persons with different cultural backgrounds and with focus on different domains together as they most probably will have different understandings of things and interoperability will fail.

3.2 Semantic Interoperability

A common understanding of the concept of interoperability is described in the Levels of Conceptual Interoperability Model (LCIM) [6]. This definition is derived from simulation theory, but it has a much broader applicability. This definition distinguishes 7 levels of interoperability which are grouped in 3 parts [7].

- Integratability contends with the physical/ technical realms of connections between systems, which include hardware and firmware, protocols, etc.

- Interoperability contends with the software and implementation details of interoperations, including exchange of data elements based on a common data interpretation, etc.
- Composability contends with the alignment of issues on the modelling level. The underlying models are purposeful abstractions of reality used for the conceptualization being implemented by the resulting simulation systems.

For computer systems, the ability to have a clear and formalized way to express the meaning of things is an indispensable precondition to achieve semantic interoperability. Bringing both terms together, semantic interoperability can be defined as “the ability of computer systems to exchange data with unambiguous, shared meaning” [1].

3.3 Semantic Web

The term Semantic Web was first used by Tim Berners-Lee in his article “The semantic web” from 2001 stating that “the Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation” [8].

The main concept of the Semantic Web is to extend the existing World Wide Web from a web of (interlinked) documents to a web of (interlinked) machine-readable and processable data. This is achieved through a family of very specific technology standards driven by the World Wide Web Consortium (W3C) which are presented in the following.

3.3.1 Resource Description Format (RDF)

RDF[1] is the (metadata) data model for the Semantic Web and therefore can be seen as its cornerstone. All data on the Semantic Web is represented in RDF, even the schema description. The main advantage of RDF is its innate flexibility compared to the tabular data model of relational databases and the tree-based data model of XML.

As shown in Figure 1, data in RDF is often depicted as a labelled, directed graph where the nodes represent *resources* (depicted as ovals) or *literals* (depicted as rectangles) and the labelled edges represent *relations*. This representation clearly shows the power of RDF to represent data without previously defining its structure, unlike with relation databases.

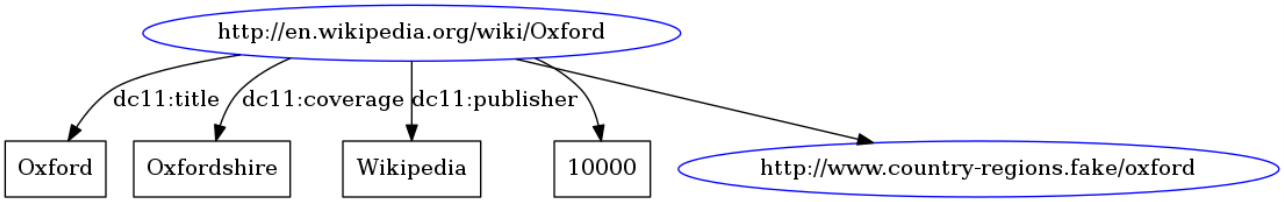


Figure 1 Example RDF data depicted as graph.

All data in RDF is described as a triple (also often called statement or 3-tuple) of the form *(subject, predicate, object)*. Subjects and predicates are resources which are represented by an URI (Universal Resource Identifier). Objects can also be a resource or a literal (which

[1] https://www.w3.org/TR/2004/REC-rdf-primer-20040210/

is only a fancy name for value). Datatypes of literals can be defined using XSD datatypes[2]. The use of URIs to identify resources (which can be seen as an atomic piece of information) allows globally unique addressing even between different databases and thus allows global interlinking of information.

Collections of triples are called a graph. For better data management (e.g. access control, simplified updating, trust), large collections of RDF data are usually segmented into different named graphs. Triples stored in a named graph are often referred to as quads as they are of the form *(graph, subject, predicate, object)*. Databases designed to store RDF data are referred to as triple (or quad) stores.

RDF is an abstract data model, which means there are multiple serialization formats that can be used to represent RDF data. The most popular are RDF/XML[3], N-Triples[4], Turtle[5], TriG[6], RDFa[7], Notation3 (N3)[8] and JSON-LD[9].

3.3.2 RDF Schema (RDFS) & Web Ontology Language (OWL)

RDFS[10] and OWL[11] are RDF schema languages which are used to define meta models for RDF data. These meta models are often referred to as vocabularies or ontologies which are explained in detail in the next section. Both, RDFS and OWL are themselves expressed using RDF.

3.3.2.1 Vocabularies and Ontologies

The terms vocabulary and ontology are terms used very frequently in the context of Semantic Web but are often defined and thus used differently. Generally, they are used to describe a set of triples with a strong logical cohesion. The W3C states, that “there is no clear division between what is referred to as vocabularies and ontologies. The trend is to use the word ontology for more complex, and possibly quite formal collection of terms, whereas vocabulary is used when such strict formalism is not necessarily used or only in a very loose sense” [9].

For the rest of the document we will use the term ontology and refer to it as a set of triples defining a meta model. This means ontologies only contain information about general concepts and no data of concrete instances (also often called individuals). The main idea behind developing ontologies is to structure data in a clear and machine-readable way to have a common understanding of things as well as to enable inference (making implicit knowledge explicit through reasoning).

[2] https://www.w3.org/TR/swbp-xsch-datatypes/
[3] https://www.w3.org/TR/rdf-syntax-grammar/
[4] https://www.w3.org/TR/n-triples/
[5] https://www.w3.org/TR/turtle/
[6] https://www.w3.org/TR/trig/
[7] https://www.w3.org/TR/rdfa-primer/
[8] https://www.w3.org/TeamSubmission/n3/
[9] https://www.w3.org/TR/json-ld/
[10] https://www.w3.org/TR/rdf-schema/
[11] https://www.w3.org/TR/owl2-overview/

3.3.2.2 RDFS

RDFS is the most basic schema language of the Semantic Web. It is a very minimalistic set of classes and properties used to describe classes of and relations between objects. RDFS also distinguishes between classes and individuals (instances of classes). The most important classes are listed in Table 1

Table 1 Most important classes of RDFS.

Class Name	Description
rdfs:Resource	all things declared by RDF are resources
rdfs:Class	describes the concepts of a class
rdfs:Literal	describes the concept of a literal
rdfs:Property	the class for properties

Table 2 Most important properties defined in RDFS.

Property Name	Description
rdfs:domain	defines to which subjects does a property applies
rdfs:range	defines the set of values a property can accept
rdf:type	used to state that a resource is an instance of a class
rdfs:subClassOf	defines one class as a subclass of another class
rdfs:label	provide human-readable version of resource's name
rdfs:comment	provide human-readable description of resource
rdfs:seeAlso	link to another resource that might provide additional information

3.3.2.3 OWL

OWL is another RDF schema language which is more expressive than RDFS and can express quite subtle ideas about data. It is very efficient as it comes in various flavours, called profiles, each with a different level of expressivity and therefore complexity and computational power needed for inference. It includes everything RDFS provides and adds a lot of new classes and properties like

- owl:TransitiveProperty
- owl:unionOf
- owl:sameAs
- owl:inverseOf
- owl:hasValue

and also some properties to model meta-meta-data like

- owl:import
- owl:versionInfo
- owl:deprecatedProperty

3.3.3 SPARQL Protocol and RDF Query Language

SPARQL[12] is the de facto standard query language for RDF data and quite similar to the query language for relational data SQL.

```
1  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3  PREFIX dbp: <http://dbpedia.org/ontology/>
4
5  SELECT ?city ?popTotal
6  FROM <http://example.com/dataset.rdf>
7  WHERE
8  {
9      ?city rdf:type <http://dbpedia.org/class/yago/CitiesInTexas> .
10     ?city dbp:populationTotal ?popTotal .
11 }
12 ORDER BY ?popTotal
```

Listing 1 An example SPARQL query.

Looking at the overall structure of the example SPARQL query in Listing 1 we see that it is quite similar to SQL. One main difference is the format of the WHERE clause as with SPARQL it consists of a list of so called triple patterns. These triple patterns are normal triples which can contain a variable (starting with a '?') on every position. When executed, the variables in the triple patterns are bound to concrete values whereat all occurrences of the same variable are bound to the same value. This concept is called graph pattern matching and the results of the query are all possible valid combinations of values bound to all mentioned variables. In SPARQL there exist multiple query types as specified in Table 3.

Table 3 SPARQL query types.

Query Type	Description
SELECT	returns a list of bindings which is basically a table like in SQL
CONSTRUCT	returns a RDF graph which is basically a list of triples
DESCRIBE	returns information about a single resource. What will be returned is not generally defined but rather implementation dependent
ASK	returns true if the query has at least one result, otherwise false

3.4 Semantic Mapping

Ontologies are a way to formally describe the concepts and relations of a domain. But even if two ontologies cover the same domain they can describe the domain quite differently, e.g. use a taxonomy with another scope or granularity, use the same terminology but in a different language or even use a different terminology. Such differences between ontologies

[12] https://www.w3.org/TR/sparql11-overview/

are called ontology mismatches. There exist multiple classifications describing and structuring ontology mismatches in literature [10] [11] [12] [13] [14].

Semantic mapping refers to the idea to resolve ontology mismatches by defining statements and rules how data expressed using one ontology can be translated into the terms of another ontology. Such a statement or rule is called a *correspondence pattern* and consists of a source ontology, a target ontology and some correspondence/transformation information. All correspondence patterns having the same source and destination ontology together form an *alignment* between the two ontologies. Such an alignment contains all correspondence patterns necessary to translate instances of the source ontology into instances of the target ontology. Some mismatches are so profound that this translation is not possible without loss of information. In fact, this is quite common as only data modelled in both ontologies, i.e. that is part of the semantic intersecting set of the two ontologies, can be safely translated between them.

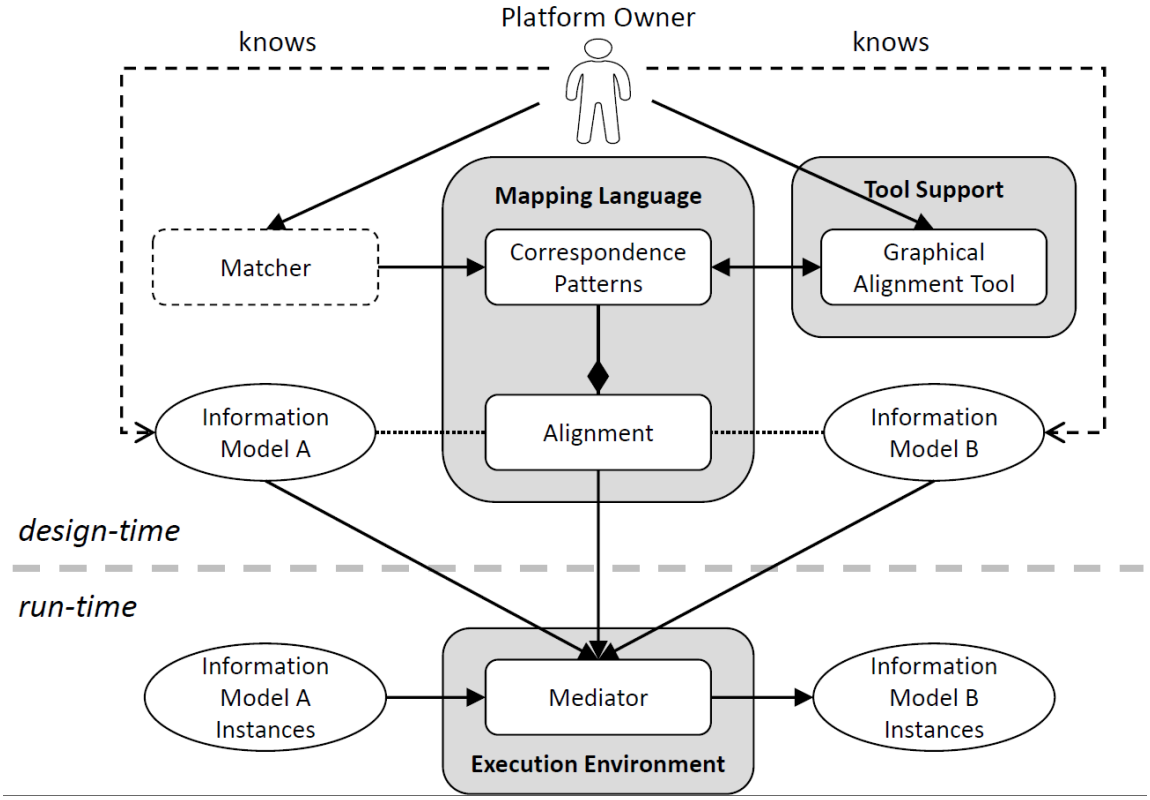


Figure 2 Schematic representation of an example usage of semantic mapping for semantic interoperability.

Figure 2 depicts a schematic representation how semantic mapping can be used for achieving semantic interoperability and which kind of software and tools are involved. In the centre we see an alignment as an aggregation of multiple correspondence patterns. To formulate, express and exchange such correspondence patterns a mapping language is needed. There are multiple existing mapping languages that can be used. The main criteria to classify them is their expressivity, i.e. what kind of ontology mismatches they can resolve, as this defines the capability of a system providing semantic interoperability to enable interoperability of systems even if their internal information models differ strongly. Some available languages are Alignment Format [15], EDOAL (Expressive and Declarative Ontology Alignment Language) [16], C-OWL [17], SWRL (Semantic Web Rule Language)

[18], MAFRA (Ontology MApping FRAmework) [19], SPARQL Construct[13] and SPARQL Inferencing Notation[14] (SPIN).

In the upper centre of Figure 2 we see a user which in our case is a platform owner of an IoT platform. He has knowledge about the information model used in his own platform (Information Model A) as well as about the one used in another platform (Information Model B). Based on this knowledge he wants to formulate a mapping/alignment between the two information models using a mapping language. To ease this process, he could use a *matcher* which is a tool that automatically discovers correspondences between ontologies, e.g. by applying different similarity measures [20] [21] [22] [23]. Furthermore, he probably would like to use some visual editor for the mapping language because the correspondences and the alignment can become quite complex [24] [25] [26] [27]. Ontology matching and graphical alignment tools are both still an active area of research.

After an alignment between two information models is defined at design-time it can be used at run-time by some kind of *mediator* to translate information expressed using information model A to information model B.

3.5 Information Models of Existing Platforms used in symbIoTe

In this section we highlight the problem of enabling interoperability between multiple platforms by providing a practical example. Therefore, we present the information models of two existing platforms, openUwedat from AIT and Symphony from NXW, that will be used within symbIoTe. These real world examples highlight the problem of semantic interoperability to demonstrate which services are needed when different IoT platforms want to exchange information.

3.5.1 openUwedat

openUwedat is not a closed system but rather a library and framework for arbitrary time series oriented applications. It does not use a single data model, but rather adapts the data model to the needs of specific applications. Nevertheless, there is a core data model that applies to all applications and there are extensions that apply to individual applications. Thus, the description of the data model is split into two parts.

3.5.1.1 General Information Model

The core data model of openUwedat is constructed around *datapoints*. These datapoints are sources and destinations of time series data.

Each datapoint can be queried to emit *TimeSeries* data. A TimeSeries is mainly a container for *Slots*. TimeSeries object are comparable to the *Observations* collection of the OData interface, which is currently used for syntactic interoperability in symbIoTe (further information is provided in Section 5).

Each Slot has a reference time and zero or more *values* assigned to it. Thus, it is closely related to symbIoTe's concept of an *Observation*, as defined in Section 5.1.2.

[13] https://www.w3.org/TR/rdf-sparql-query/#construct
[14] https://www.w3.org/Submission/spin-overview/

Each slot's value can be any type (restricted to Java types at the moment). This is related to symbIoTe's idea of an observation value.

TimeSeries have *Properties*. They are addressed in a dictionary style by using key-strings. For this reason, the property system is easily extendable with new properties needed for particular applications.

There is a set of Properties within the core model whose existence is mandatory or at least strongly recommended:

- ValueKeys: This key describes which values are available within a slot.
- ValueUnits: A description of the Units of Measurements (UoM) related to each value.
- QueryInterval: This property describes the interval covered by the TimeSeries. This property is especially useful if the timely pattern of slots is only sparsely populated so that deriving the coverage from contained slots is not possible. This Property can also be used for paging in case a datapoint decided to not give back the complete query interval that was requested.

Datapoints also have properties. The only mandatory property is *ObservedProperty*.

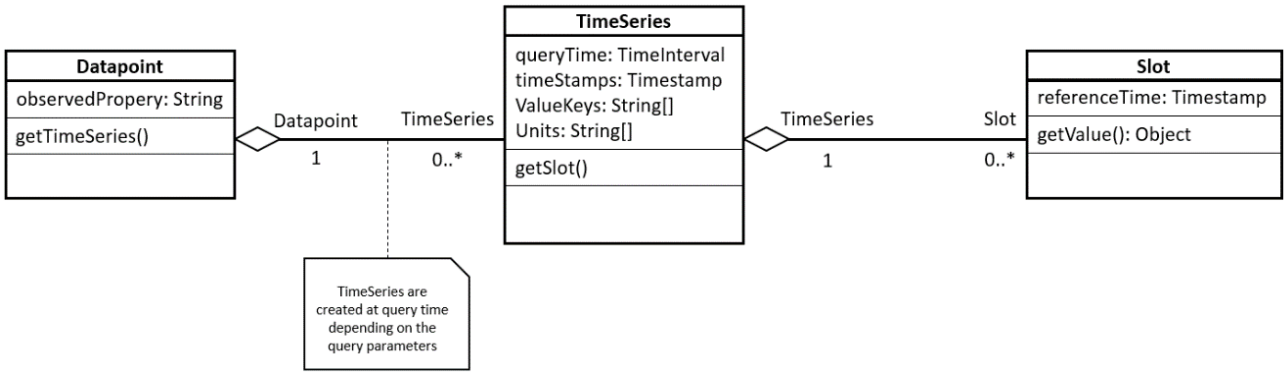


Figure 3 The core Information Model of openUwedat.

3.5.1.2 Information Model of the SymbIoTe installation

For symbIoTe most of the core model was already covered by openUwedat's core model.

Two important extensions were needed nevertheless:

- We needed a simple ID that can be exposed via the OData interface. This was added to the set of properties.
- Each datapoint (aka Sensor or Resource) has a concept of location. For the symbIoTe use case this concept is simple as we are dealing with fixed stations. So we just added a property *location* which is composed of longitude, latitude and altitude.

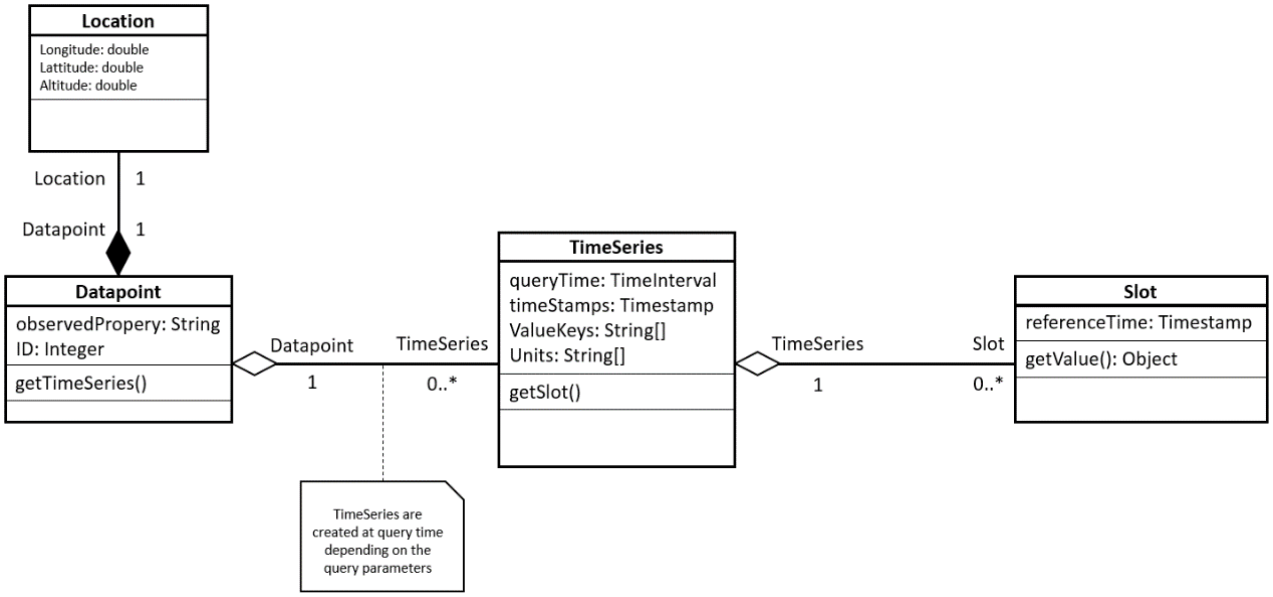


Figure 4 openUwedat Information Model for SymbIoTe.

3.5.2 Symphony

Symphony is the NXW platform for the integration of home/building control functions, devices and heterogeneous subsystems. Symphony is a service-oriented middleware integrating several functional subsystems into a unified IP–based platform. As hardware/software compound, Symphony encompasses media archival and distribution, voice/video communications, home/building automation and management, and energy management. The platform owns a generalized abstract model for all the Internet Connects Objects (e.g., smartphones, printers, sensors, actuators, etc.), managing a set of context-driven decisions/actions. This leads to a quite complex data model, partitioned into subcomponents, one for each service provided by the platform.

Figure 5 depicts a high-level diagram which explains the Symphony platform data model.

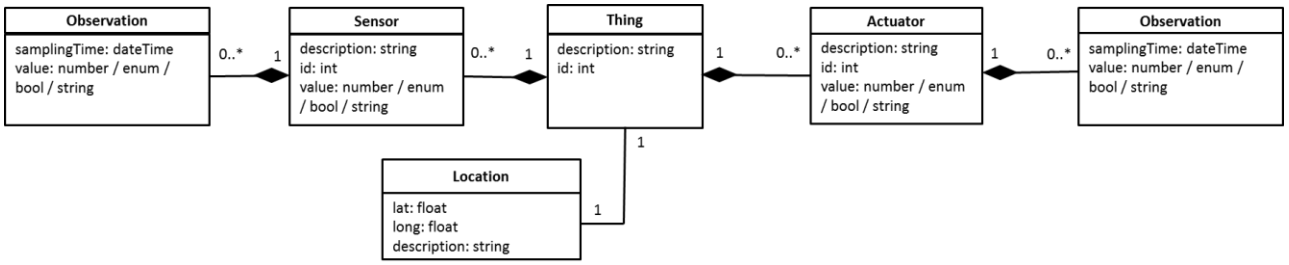


Figure 5 The Symphony data model.

Since Symphony integrates a large number of sensors and actuators, two examples are shown below:

- a temperature sensor (depicted in Figure 6)

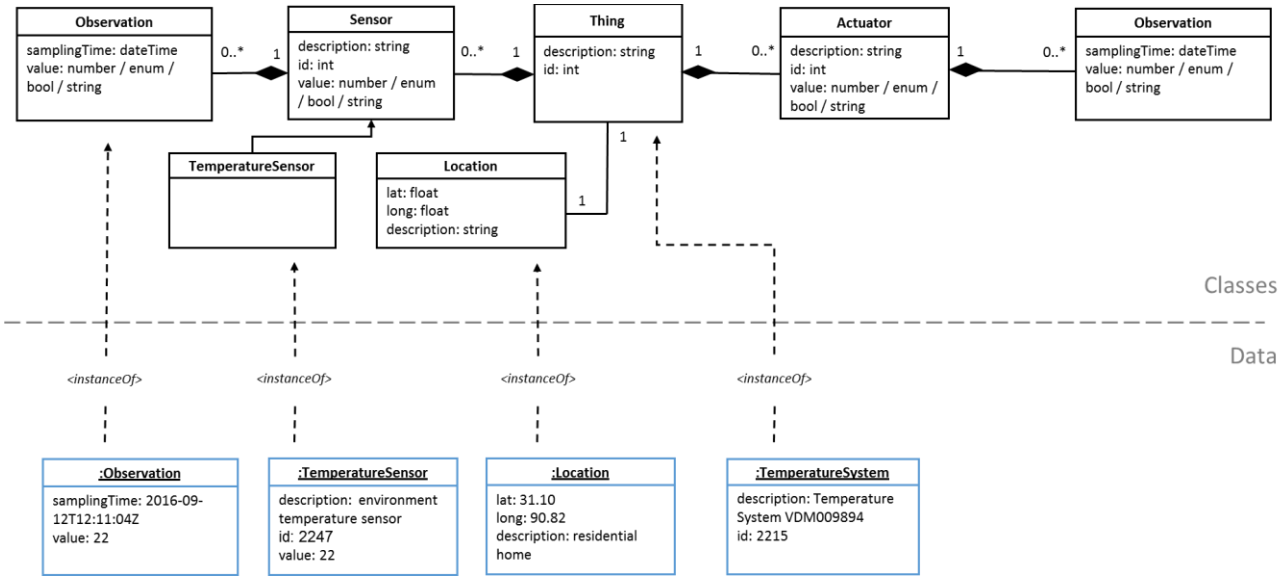


Figure 6 Data model for a temperature sensor in Symphony.

- a linear load actuator (depicted inFigure 7)

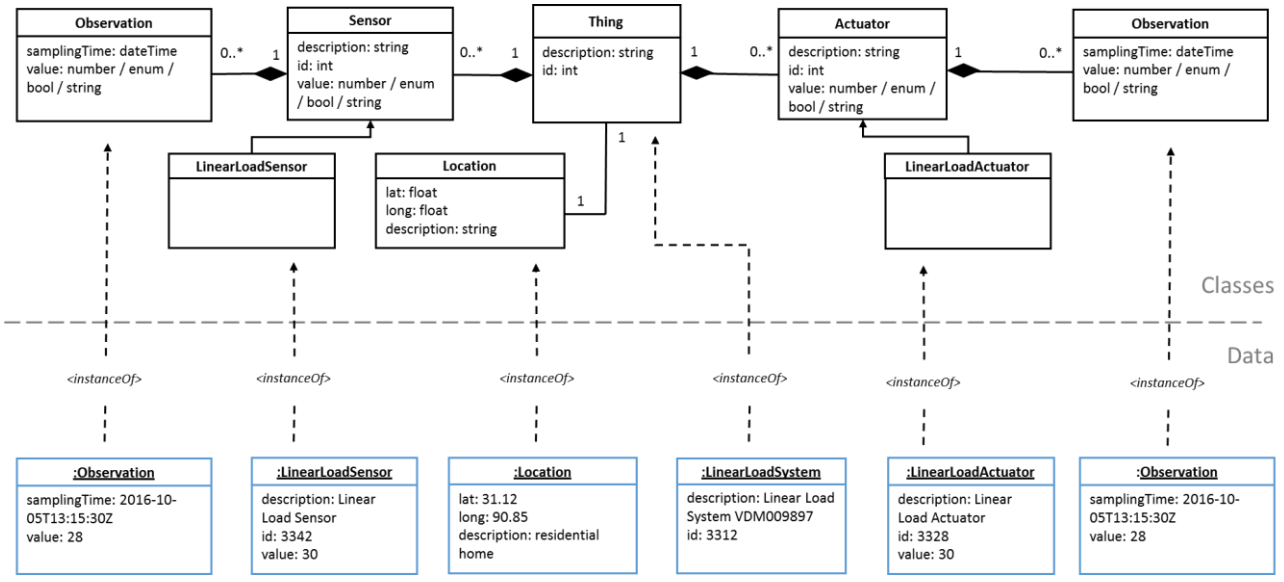


Figure 7 Data model for a linear load actuator in Symphony.

4 Achieving Semantic Interoperability

In this section we introduce the problem of semantic interoperability between multiple IoT platforms. Furthermore, we present multiple possible approaches how semantic interoperability can be achieved on a general level together with their advantages and disadvantages which is the outcome of the analysis on semantic interoperability within the symbIoTe project.

4.1 Problem

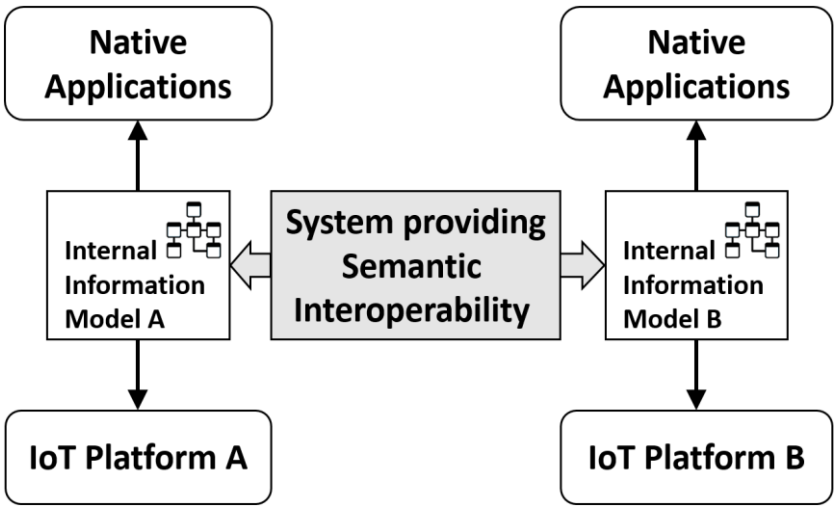


Figure 8 Schematic representation of the problem of semantic interoperability between different IoT platforms.

Figure 8 depicts the general problem of semantic interoperability between multiple IoT Systems. Each IoT platform models its data using an internal information model that is used by native applications of this specific platform. As different IoT platforms are designed by different bodies/communities/companies and often focus on different aspects within IoT they tend to have platform-specific information models that differ in multiple aspects as presented in Section 3.4. Closing this semantic gap means enabling semantic interoperability.

4.2 Possible Approaches

As outcome of our research on the problem domain of how to achieve semantic interoperability we identified a possible solution space which is depicted in Figure 9. It can be thought of as a line between the two most radical and opposed approaches which are, on the one side, using a single core information model that all platforms must use and, on the other side, using completely independent platform-specific information models for each platform which then need to be aligned using semantic mapping techniques. In between, there exists a large, not clearly defined number of intermediate solutions from which three representative ones are chosen and presented in the following together with the two radical ones. These approaches are motivated by and in-line with the concepts presented by Wache et al. [28] and Choi et al. [21]

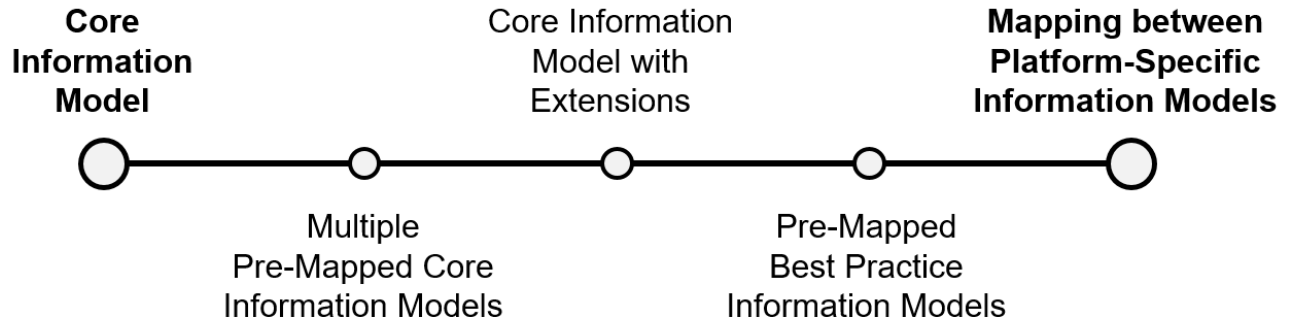


Figure 9 Solution space for possible approaches to semantic interoperability.

4.2.1 Core Information Model

The most widespread approach amongst existing platforms is to use a single core information model that all platforms must comply with. This means that a platform can only expose data that fits into this core information model as custom extensions are not permitted. If a platform needs to expose data that does not fit into the core information model the platform cannot expose this data and cannot inter-operate with others. From our perspective this is rather some form of standardization than an approach to enable true semantic interoperability that will also work without adaption (e.g. changing the “standard”/Core Information Model) when new IoT platforms covering eventually new domains will emerge.

Pros

- easy to implement and use since the data from all platforms follows the same information model
- resulting system easy to use for app developers who only need to know one information model

Cons

- finding/defining an information model all platforms can agree upon may be difficult
- information model tends to become complex as it must comprise all data that should be exchangeable between platforms
- will always exclude some platforms whose internal information model does not fit the core information model
- no way to integrate future platforms with information models not compatible to the core information model without breaking the existing system

4.2.2 Multiple Pre-Mapped Core Information Models

Based on the single core information model approach this one tries to make it more easy and convenient for platform owners to integrate their internal information model by supporting not only a single core information model but multiple ones. To achieve that a large number of existing platforms can easily participate it would be a good idea to choose well-established information models (e.g. the Semantic Sensor Network Ontology [29] (SSN) or the oneM2M ontology [30]) as core information models. To ensure interoperability between platforms using different core information models the supported core information models are already mapped to each other. As it will not always be possible to map two core information models completely there will be some degree of information loss if platforms

conform to different core information models but if they conform to the same one they will be fully interoperable.

Pros

- flexible approach as further core information models and mappings can be added over time
- does not enforce use of one single core information model which excludes less platforms from participating

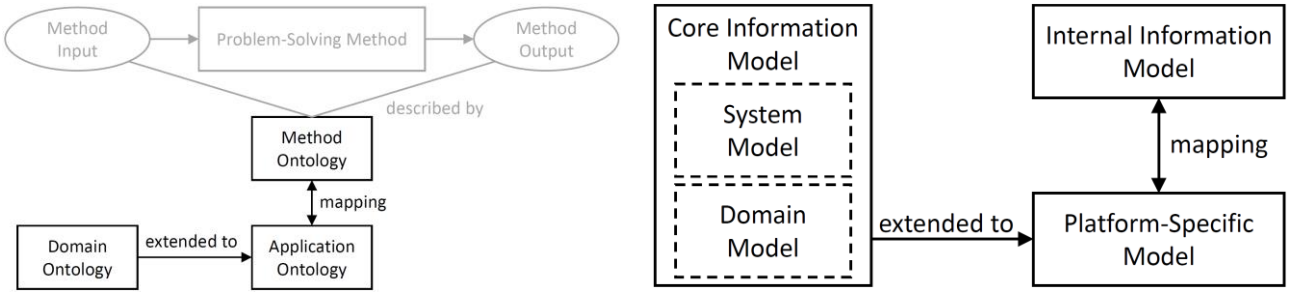
Cons

- may still exclude some platforms whose information model does not match any of the core information models

4.2.3 Core Information Model with Extensions

This approach is based on an information model that is designed to be as abstract as possible but at the same time as detailed as needed. Therefore, the core information model should try to only define high-level classes and their interrelations which act as extension points for platform-specific instantiations of this information model. These platform-specific instantiations either use the provided classes directly or they can define a subclass which can hold any platform-specific extensions to the core information model, e.g. additional properties. Besides the high-level classes the core information model may also contain properties the system needs which will be very general properties like *ID* or *name* in most of the cases.

This approach resembles an approach for a model-driven knowledge engineering system (KES) presented by Studer et al. shown in Figure 10a where a domain ontology is extended to an application ontology which is mapped to a method ontology that is finally used to define in- and output of a method used to solve a problem. The core information model with extensions can be very closely matched to this approach as depicted in Figure 10b. The main difference is, that there exists not only a domain ontology that is extended but rather the
core information model which contains the domain model and the system model (which can be seen as a platform-specific extension of the domain model to the system that provides the interoperability). The application ontology corresponds to the platform-specific model which is a platform-specific extension to the core information model and the method ontology corresponds to the internal information model of the platform as depicted in Figure 10.



(a) Ontology-based approach for a model-driven KES (based on Figure 6 from [11]).

(b) Conceptual structure of the Core Information Model with Extensions approach.

Figure 10 Structural similarity between an ontology-based model-driven KES and the Core Information Model with Extensions.

This results in an information model that has a minimalistic core that all platforms must conform to and extension points to realize custom requirements. Two platforms using different extensions can directly understand each other in terms of the core information model and when they need also to understand the custom extensions they must define a semantic mapping between their extensions.

Pros

- provides basic interoperability between platforms by defining minimalistic core information model
- provides full flexibility by custom extensions, i.e. no platforms are excluded
- high acceptance from adopter-side as it combines basic out-of-the-box interoperability (by the core information model) with support for complex scenarios (through extensions and semantic mapping)

Cons

- requires semantic mapping when custom extensions need to be understood by different platforms
- defining a semantic mapping can be a complex task and requires additional work from developers/platform owners
- design of the core information model is a complex task

4.2.4 Pre-Mapped Best Practice Information Models

Essentially, this is the same approach as *Multiple Core Information Models* but with one small but significant modification: the provided information models are no longer seen as core information models but rather as *best practice* information models. Hence, platforms must not be compliant to any of the provided information models as in the previous approach but can choose their information model freely. If they choose to re-use one of the provided best practice information models they will gain instant interoperability to other platforms also aligned with one of the best practice information models.

Pros

- no limitations on information model, hence does not exclude any platform
- best practice information models make usage for inexperienced platform owner more easy

- better and broader interoperability due to already aligned best practice information models

Cons

- no initial interoperability between platforms as long as no mapping is defined when no pre-mapped information model is used
- defining a semantic mapping can be a complex task and requires additional work from developers/platform owners

4.2.5 Mapping between Platform-Specific Information Models

In this approach, there isn't anything like one or more core information models. Instead, every platform independently provides its own information model. Interoperability is only achieved through mapping between these platform-specific information models.

Pros

- not limited only to a fixed set of information models but rather supports all possible information models
- mappings can be added iteratively increasing the degree of interoperability

Cons

- no initial interoperability between platforms as long as no mapping is defined
- defining a semantic mapping can be a complex task and requires additional work from developers/platform owners
- the system does not understand any of the data it is processing

4.3 Comparison of Approaches

Section 4 presents the analysis of possible approaches for the semantic interoperability. It is required to decide how independent IoT platforms should exchange meta-information and thus create a pool of IoT data sources, resources and services available to the applications. Such an interoperability is a crucial functionality for symbIoTe because it addresses the need of presence of framework across existing and future IoT platforms. The framework will enable discovery and sharing of resources for rapid cross-platform application development. Those applications exploiting multiple data sources and resources will bring new innovative functionalities and lift up the IoT to the next technology level.

The first described approach “Core Information Model” seems to be the most suitable to enforce the interoperability between the IoT platforms. Theoretically, if they speak only one language there is no need to warry about inconsistency, complexity and the performance issue of additional operations like translations (complete or incomplete). Moreover, developers are satisfied because they can stick to only one standard solution so application development is quicker and cheaper from the business point of view. The more applications the more innovative ideas and therefore better IoT based services. This vision is compelling but the reality is different. Heterogeneity is the characteristic that has to be accepted and considered when a service or a product is offered to the market. Nowadays, there are many IoT platforms utilizing different information models to describe their resources, applying policies to share their data and comprising implementation limits preventing smooth integration with other platforms. It would be extremely difficult to convince IoT platform vendors to make deep (and thus expensive) changes in their products. More realistic is to propose the solutions which try to find some compromise and balance. “Core Information Model” is not an optimal approach but also the interoperability with the use of all possible

platform-specific information models (“Mapping between Platform-Specific Information Models”) is not a straightforward direction. In this case semantic mapping and complexity for developers bring disadvantages like translation performance issues, possible slow progress of application development, incomplete translations and the costs of supporting of new emerging information models.

If two aforementioned extreme approaches are not suitable then one can analyse the three others described in the previous subsections. The first one, “Core Information Model with Extensions”, specifies the very abstract representation of information model which may be applicable to any IoT platform. This allows for exchanging at least a set of basic, platform-independent information. Any specific information may be modelled as an extension including required mapping (translation) between extensions of respective platforms. Apart from obvious advantages like flexibility and partial standardization one should emphasize the downsides. It is not clear if the generic abstract representation is enough useful for interoperability and effective in real use cases. Moreover, certain mappings between extensions may still be complex and incomplete. Regarding the last two approaches, “Multiple Pre-Mapped Core Information Models” and “Pre-Mapped Best Practice Information Models”, they are based on mappings of a subset of information models. They require the implementation of translation mechanisms which results in all related difficulties but may be suitable if it is assumed that the number of information models is limited and stable. Moreover, those models are well-known and widely accepted.

Section 4 does not provide the answer, which approach for the semantic interoperability should be chosen by the symbIoTe project. This is an analysis presenting their characteristics and comparing week and strong points. Thus, it should be an input to the further discussion on the final selection decision.

5 symbIoTe's Approach to Semantic Interoperability

We analysed the approaches to semantic interoperability presented in Section 4 regarding their suitability for symbIoTe and decided to follow the Core Information Model with Extensions approach due to mainly two reasons. First, symbIoTe needs to have at least some degree of understanding of the resource descriptions exposed by the platforms to be able to provide additional services like location-based search for sensors. For this, we need platforms to use the same terms to describe all information that is relevant to symbIoTe. This is achieved by having a common, minimalistic Core Information Model covering these symbIoTe-relevant terms. Second, this approach gives almost full flexibility to platforms as it allows platforms to model all non-symbIoTe-relevant information within extensions without any restrictions. Due to the first reason, the approaches Pre-Mapped Best Practice Information Models and Mapping between Platform-Specific Information Models are not suitable as with these approaches symbIoTe would not be able to understand any of the resource descriptions exposed by the platforms. On the other side, the approaches Core Information Model and Multiple Pre-Mapped Core Information Models are not suitable as they do not only enforce usage of a minimalistic core but with these approaches a core information model is a complete model of the IoT domain and thus would massively limit the degree of freedom of the platforms to expose their data as they want or need to.

Nevertheless, this approach is quite challenging and may render too complex for implementation. Hence, aiming also at a solution that is feasible in the next period, together with working on the Core Information Model with Extensions approach, we will also consider adding a Best Practice Information Model on top of the Core Information Model. Such a Best Practice Information Model (BIM) can be seen as a special form of a Platform-Specific Information Model which is provided together with symbIoTe. Its use will be optional and using it will allow platform owners to register their platforms to symbIoTe without having to use semantic technologies but rather using simple clearly defined REST- and JSON-based interfaces. This dual approach allows us to provide simple interfaces and out-of-the-box interoperability where the Best Practice Information Model is applicable and also full support for custom Platform-Specific Information Models aligned with mappings when needed. A more detailed description of the Best Practise Information Model approach is currently under definition and will be documented in the next and final deliverable on semantics. In the remainder of this section we present the Core Information Model with Extensions approach.

Figure 11 shows the concept for realizing the Core Information Model with Extensions approach presented in Section 4.2.3. On the left and right hand side we see two typical vertical IoT silos. Each platform uses its own internal information model to provide a platform-specific API which is then used by native applications. Between those two vertical IoT silos we see the symbIoTe interoperability framework depicted in light grey providing interoperability between the two IoT platforms. As proposed in Section 4.2.3 and shown in Figure 10b, symbIoTe uses two central information models: the Core Information Model (CIM) describing domain specific information (matches the Domain Model in Figure 10b) and the Meta Information Model (MIM) describing symbIoTe internal meta information about platforms and resources (matching the System Model in Figure 10b). For a platform to become symbIoTe-compliant, it must expose its data using a Platform-Specific Information Model (PIM) which is basically the CIM with platform-specific extensions to it. The main part of the actual interoperability happens via semantic mapping as shown by the arrow connection between the two platform-specific information models. This allows to define how the platform-specific extension of one platform can be translated into the platform-specific

extensions of another platform and therefore allows to define an arbitrary degree of interoperability between two platforms. When an app or a platform queries symbIoTe to find resources of interest on all available platforms, symbIoTe uses these mappings to re-write the query to fit the platform-specific information model of each platform when needed and execute it against the metadata it has stored about each of them. Details on the symbIoTe Information Model (SIM) as well as semantic mapping and SPARQL query re-writing are provided in the following sections.

To enable syntactic interoperability, which is a prerequisite for semantic interoperability, we need platforms to expose their data in a unified way. At the moment, symbIoTe is using the Open Data Protocol[15] (OData) for this purpose but since it has no native support for subscription-based access to entities this decision might be revised. Furthermore, this issue has been identified within the IoT-EPI Task Force “Platforms Interoperability” (TF02) as a potential collaboration and standardization subject between projects.

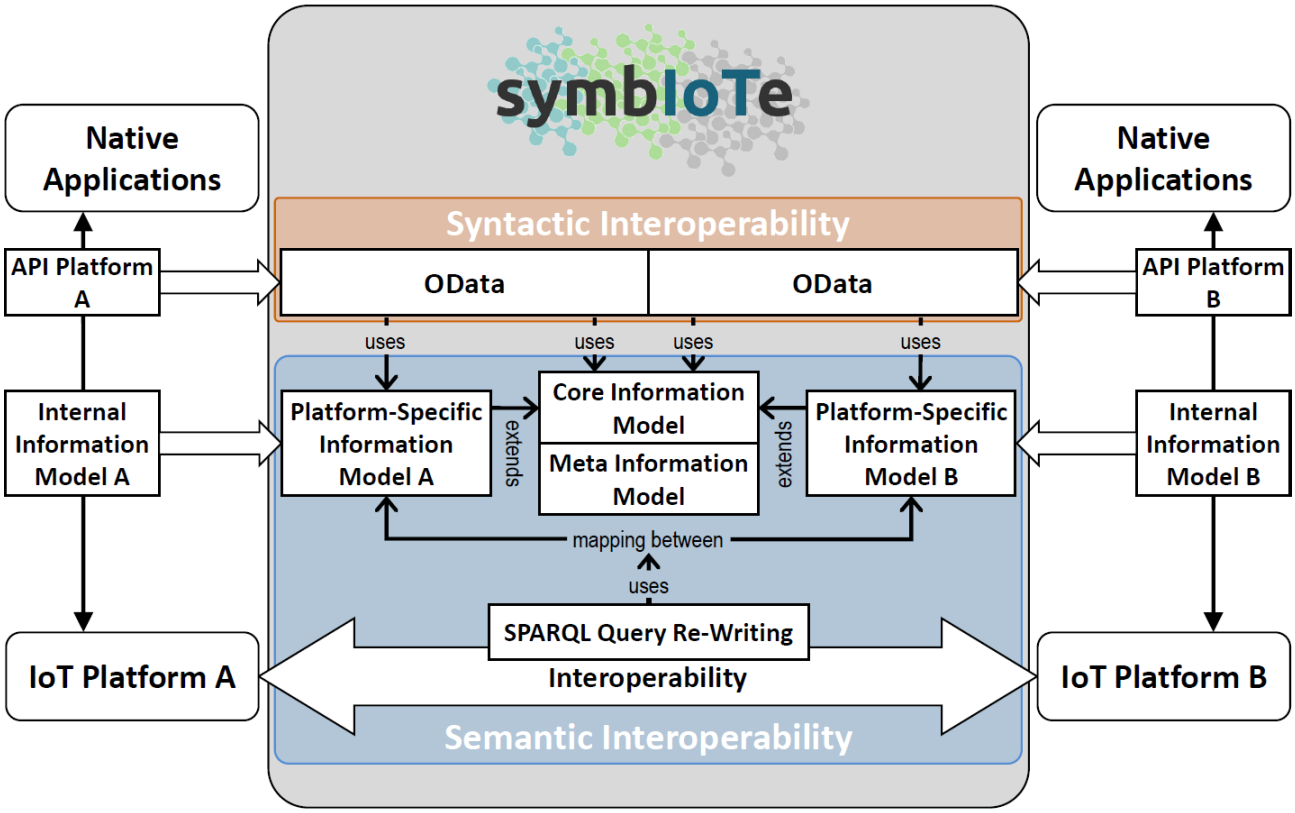


Figure 11 High-level diagram showing how symbIoTe approaches syntactic and semantic interoperability.

[15] http://www.odata.org/

5.1 symbIoTe Information Model (SIM)

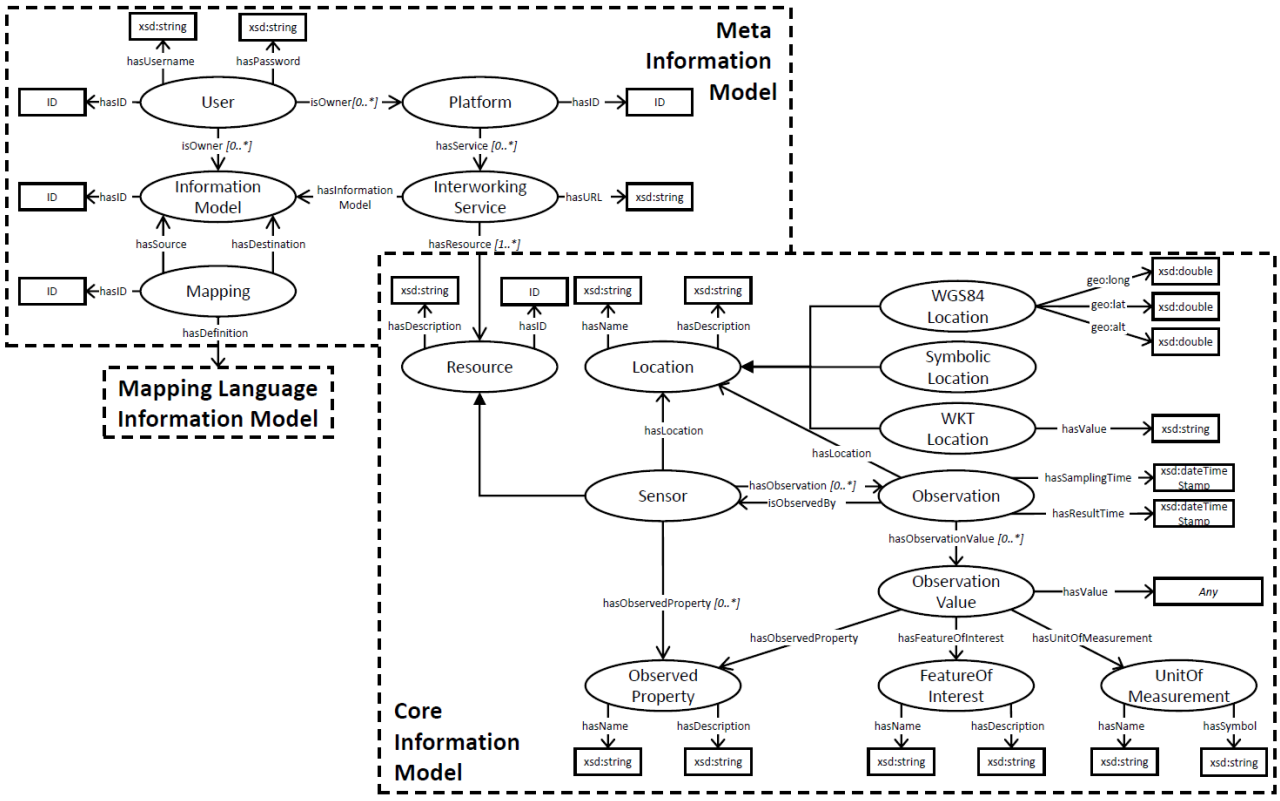


Figure 12 The symbIoTe Information Model (SIM).

The symbIoTe Information Model (SIM) as shown in Figure 12 comprises all classes (also called concepts) and their relations (also called predicates) which symbIoTe can understand. It is composed of two parts, the Meta Information Model (MIM) and the Core Information Model (CIM) which are explained in detail in the following sections. All information models are realized as an OWL ontology. Furthermore, they are developed in an iterative process as it is considered the best practice for ontology design [31] and currently in version 0.2.

5.1.1 Survey on Domains

To find out which domains need to be part of the SIM, we conducted a project-internal survey. We started out collecting requirements about relevant domains and came up with the following twelve domains which are divided into four groups:

- Observation & Measurement
  - Time: Concepts of time instant, time interval, time zones, etc.
  - Location: Basic geo-location like long/lat/alt, polygon, symbolic location.
  - Sensors: Definition of an abstract concept of a sensor.
  - Units of Measurement: Abstract concept, may include some model of units.
- Access & Control
  - Actuators: Definition of an abstract concept of an actuator.

- Services: Services offered by symbIoTe-compliant IoT platforms.
- Protocols: Protocols via which these platforms can be accessed.

- Interoperability
  - symbIoTe infrastructure: Description of symbIoTe-compliant platforms for discovery purpose.
  - Quality of Service / SLA: Concepts describing QoS parameters of interoperability e.g. reliability, availability, response time, etc.
  - Trading & Bartering: Concepts like cost and utility functions, algorithms, etc.
- Security
  - Identity & Access Management: Concepts needed for IAM like roles, access rights, etc.
  - Encryption: Abstract concept, may include concrete encryption algorithms.

We then asked the project partners to state their opinion if these domains must, should or could be part of a core ontology. The outcome is shown in Figure 13 which shows that location and time are believed to be the most important concepts together with units of measurement and sensors. For identity & access management, actuators and symbIoTe infrastructure we have still a rather strong opinion that they should be part of the CIM. Services and Protocols are up for discussion as at most 50% of partners think that this should be an essential element of the CIM. The same goes for Bartering & Trading. Encryption & Quality of Service / SLA is not considered to be part of the CIM.

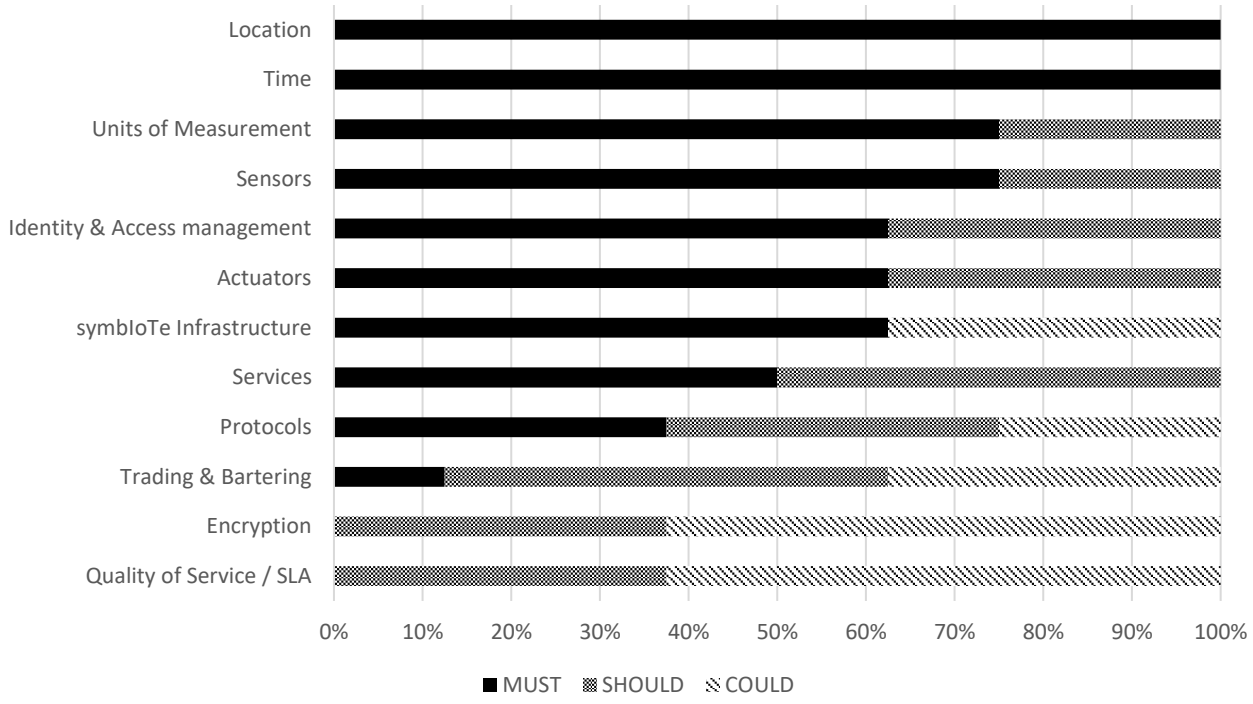


Figure 13 Outcome of the internal survey on which domains must/should/could be modelled within symbIoTe.

5.1.2 Core Information Model (CIM)

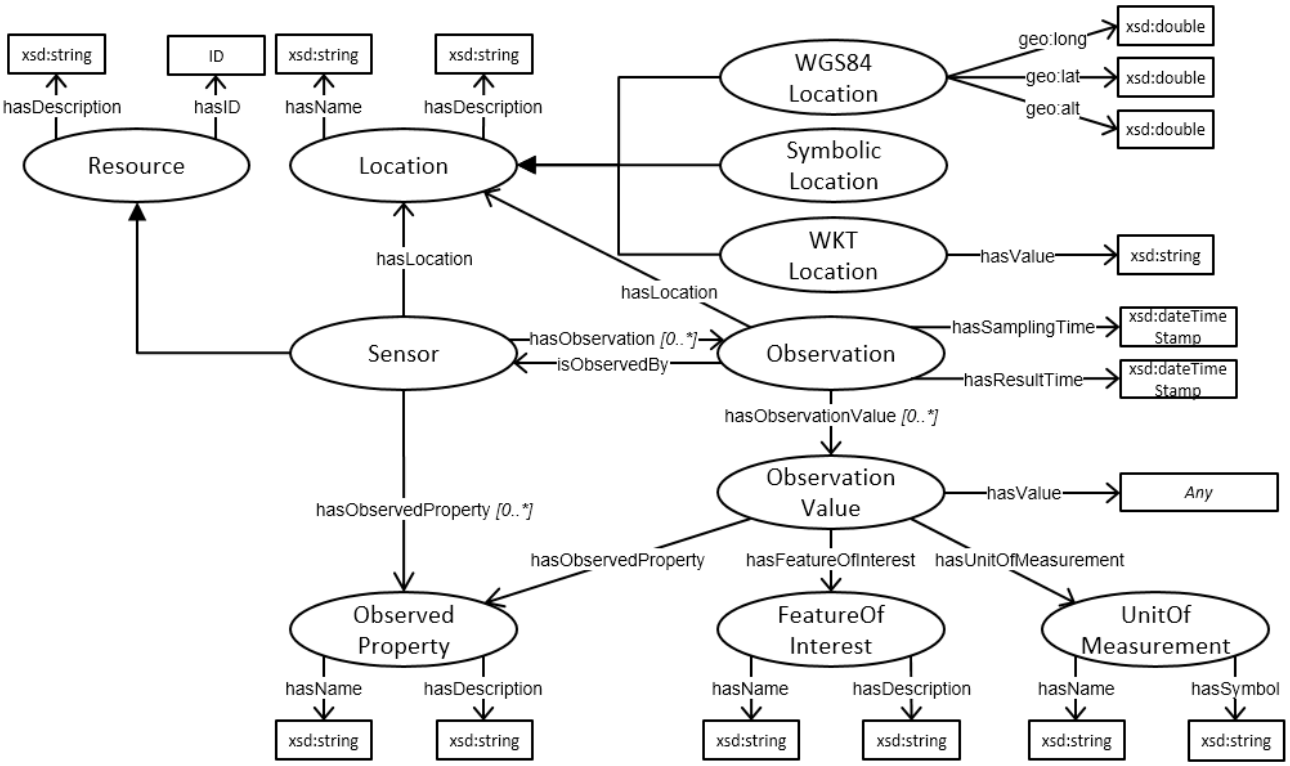


Figure 14 The symbIoTe Core Information Model (v0.2).

The Core Information Model (CIM) comprises all the metadata about resources exposed by platforms connected to symbIoTe that symbIoTe can understand. It is designed to be as abstract as possible and as specific as needed at the same time. The design principles of the CIM and why we are not re-using any existing ontology for that is explained in the following section. The concepts and relations which are selected to be included in the proposed CIM use as input the conducted survey on domains presented in the previous section.

The idea of having such a CIM is that all platforms are forced to use a very minimalistic information model so that symbIoTe can understand at least the essential descriptions of platform resources that it needs to understand to work properly. We assume that a platform which is symbIoTe-enabled must provide its own PIM to expose its resources and associated data to the outside world and symbIoTe. This PIM must be fully compliant and aligned to the CIM. For details on the relation between PIM and CIM see Section 5.2.

It is to notice that the CIM is developed in an iterative process and Figure 14 depicts the current version (v0.2) and not the final one. There are a number of open issues under discussion, e.g. how to properly model mobile and stationary sensors as they have different requirements regarding the relations between Sensor and Location as well as Observation and Location. Furthermore, the CIM will be extended with concepts regarding actuators in a later version.

5.1.2.1 Why we need a new information model

The design of the CIM was driven by two principles: on the one hand we wanted to be as explicit as needed to at least include all the concepts symbIoTe needs to understand to

provide its functionality and services. In other words, we need the concept of a resource which has an identifier or the concept of a sensor and observations which must be associated to a location so that symbIoTe is able to support location-based search for sensors and observations. On the other hand, we need to be as abstract as possible, i.e., we must be careful not to introduce any concept and relation that is not really essential, as each introduced concept and relation reflects a design/modelling decision which needs to be shared by platforms that want to use symbIoTe. Thus adding any concept or relation that is not essential to the CIM can potentially exclude some platforms from using symbIoTe because they won't be able to fit their internal information model with the CIM.

Based on these two partially contradictory design principles we figured out that directly re-using any existing ontology is not suitable for our needs. In general, there are mainly two types of IoT ontologies that could be considered for re-use. First, the more abstract type, e.g., the SSN ontology which resemble the CIM very closely. In fact, abstract ontologies are almost what we need (that is why they are so similar to the CIM) but existing ontologies are either too broad, i.e., containing concepts that are not relevant for symbIoTe or, which is worse, do not fully cover the needs of symbIoTe (e.g., support of moving sensors). However, the CIM is strongly influenced by the existing ontologies, especially the SSN ontology and the SensorThings API information model [32]. The second type of IoT ontologies are those which are on a more practical level. They often extend abstract ontologies with additional concepts for practical use, e.g., the IoT-Lite[16] ontology which is based on the SSN ontology. Such ontologies are not suitable for symbIoTe as they are far too application specific and not abstract enough to be used as CIM.

5.1.3 Meta Information Model (MIM)

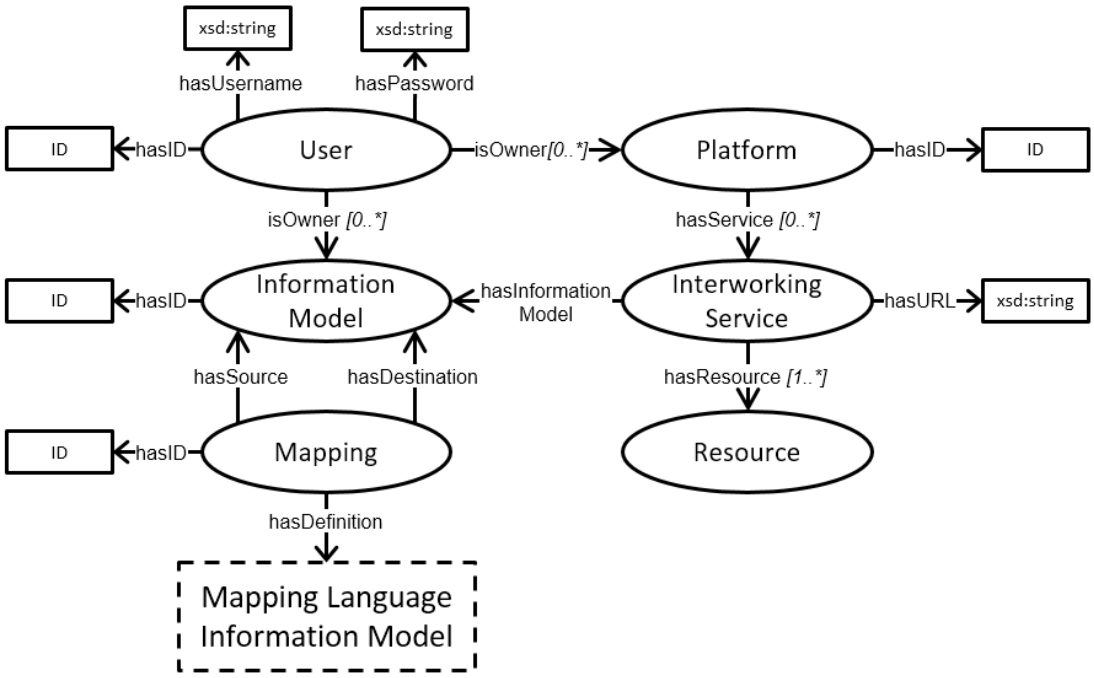


Figure 15 The symbIoTe Meta Information Model (v0.2).

[16] https://www.w3.org/Submission/iot-lite/

The Meta Information Model (MIM) is used to store meta information about platforms in symbIoTe for internal use. It provides the concepts describing the platforms which are registered to symbIoTe: platform owner, endpoints for data access (URL of the Interworking Interface implementation/Resource Access Proxy), information model used by endpoints and definitions of the mappings between information models. The last concept is linked to an external ontology used to describe these mappings. We are currently considering to use the Expressive and Declarative Ontology Alignment Language (EDOAL) [33] [16] for this purpose.

5.2 Platform-Specific Information Models / Extending the Core Information Model

To make an IoT platform symbIoTe compliant, the platform must expose its resource metadata to the outside world based on a Platform-Specific Information Model (PIM). The PIM needs to be compliant with the CIM, which means that it must contain all concepts and relations that are defined in the CIM and is not allowed to alter them. Basically, any PIM can be seen as an extension to the CIM adding all concepts and relations that are necessary to completely describe the IoT domain and also platform-specific domains.

As it may be a hard task for platform owners to define a PIM using RDF, especially if they are no experts in semantic technologies, to define such a PIM using RDF symbIoTe is considering to merge the concept of a best practice information model with the Core Information Model with Extensions approach. This means that symbIoTe would provide a Best Practice Information Model (BIM) which is compliant to the CIM and can be used by platforms as their PIM if it suits their needs. Additionally, platforms will automatically gain semantic interoperability to all other platforms also using the BIM without the need to define any mappings. This is expected to massively lower the entrance barrier for platform owners who are not that familiar with semantic technologies to make their platform symbIoTe-compliant and enable interoperability to other platforms.

However, using the symbIoTe-specific BIM should be optional so that platforms with special information models that are incompatible with BIM are not excluded from the symbIoTe ecosystem. Such platforms will rather have the possibility to define their own PIM and manually align it with other PIMs (as well as the BIM).

5.3 symbIoTe Architecture from the Semantic Interoperability Perspective

In this section, we provide details on how the chosen approach to semantic interoperability influences the symbIoTe architecture as defined in deliverable D1.2 “Initial Report on System Requirements and Architecture” [34]. We start from the component diagram defined for symbIoTe Level 1 compliance presented in D1.2 in Figure 16, and highlight all components in green that are affected by the proposed solution for semantic interoperability. The previously defined information models (MIM, CIM, PIM) have the strongest impact on the symbIoTe architecture.

The most affected parts of the architecture are those that enable the interaction between the symbIoTe Core Services and symbIoTe-enabled IoT platforms, as well as between applications/enablers and IoT platforms. We can identify three main functionalities that directly depend on the information models: 1) registration of platform/resources, 2) search

for resources and 3) data acquisition from a platform. Further details on how each component is affected by the integration of semantic interoperability concepts are given in the following tables and they should be considered as expanded description of the components introduced in deliverable D1.2. Note that these descriptions are only a proposal on how to integrate semantic interoperability concepts within the next version of the symbIoTe architecture.

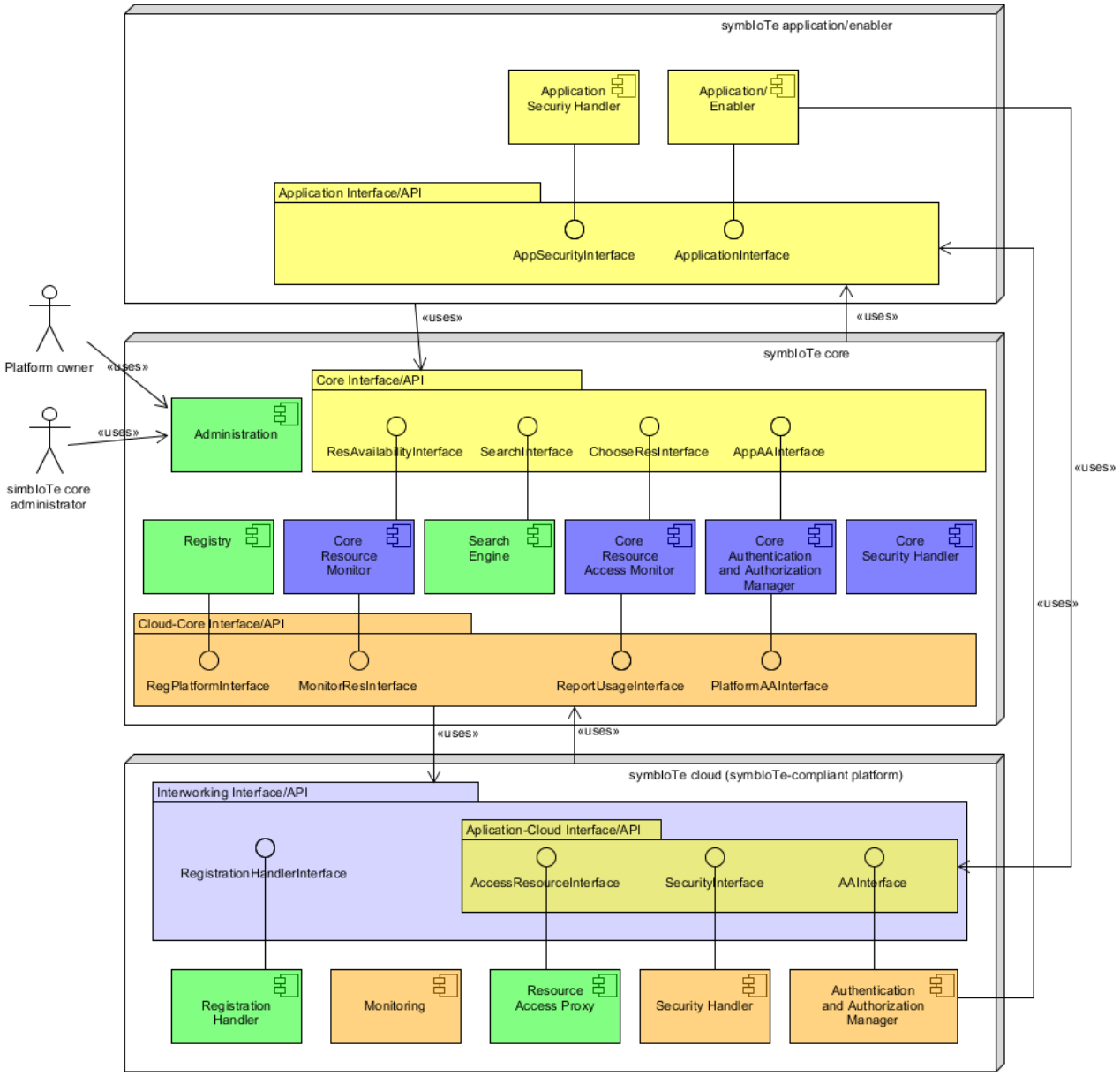


Figure 16 symbIoTe component diagram for Level 1 compliance (taken from D1.2) with changes regarding semantic interoperability highlighted in green.

Table 4 Changes of the Administration component introduced by semantic interoperability.

Component	Administration
symbIoTe Domain	APP

Description	This component enables registration of a PIM through its interface. The PIM should be provided in the RDF format which can be verified to check if the PIM is aligned with the CIM. If the two models are aligned, the PIM can be stored in Registry. This component will also provide an interface to define a mapping between two PIMs. It allows a user to create a mapping between two PIMs to indicate relations between PIM entities by using an alignment language (e.g. EDOAL). The mappings defined through the Administration interface are stored in Registry and are later on used by Search Engine to retrieve results across multiple platforms even if they don't use the same PIM.
Provided functionalities	• Provides an interface for registration of a PIM • Provides an interface for definition of the mapping between two PIMs
Relation to other components	Registry: stores PIMs and defined mappings between two PIMs.

Table 5 Changes of the Registry component introduced by semantic interoperability.

Component	**Registry**
symbIoTe Domain	APP
Description	The component must provide unique symbIoTe identifiers to all resources registered within the symbIoTe Core Services. Uniqueness must be enforced both within and across IoT platform boundaries, which is critical, e.g., in the case of roaming IoT devices. The Id is also assigned to all metadata resources, such as: symbIoTe users, PIMs or mappings between PIMs. A core information model with extensions (i.e. PIM) must be supported by the Registry for the description of available resources across IoT platforms. The symbIoTe Core Information Model should be compatible with existing standards, prominent ontologies in the IoT space and should be aligned (if possible) with the data model of other IoT-EPI projects. The model must support geo-referenced information. It must distinguish IoT devices which are fixed (their geo-location does not change over time) and mobile (their geo-location continuously changes). The information generated by IoT devices must be annotated by standard units. The information model of an IoT platforms and enablers (i.e., PIM) registering their resources to symbIoTe should be aligned to the symbIoTe core information model (i.e., CIM). All interfaces should communicate using the Semantic Web technologies. In case of data manipulation (insertion, deletion, update), resources and resource metadata should be communicated in the RDF format (e.g., as payload of REST message). To execute a search query issued by the Search Engine, the Registry should offer an interface that accepts the SPARQL query.

Provided functionalities	• Handles requests for resource registration and resource. • Stores resource metadata. All resources are described using the appropriate symbIoTe information model (platforms and devices using the PIM, PIM mappings using the MIM).
Relation to other components	Search Engine: uses platform, enabler and resource-related information (PIM) managed by the Registry and uses mappings between PIMs to execute re-written search queries.

Table 6 Changes of the Search Engine component introduced by semantic interoperability.

Component	**Search Engine**
symbIoTe Domain	APP
Description	This component allows to search for registered resources across platforms registered to symbIoTe in a unified way. A query must be formulated against an information model known to symbIoTe, which can either be the Core Information Model (CIM), any Platform-Specific Information Model (PIM) or the Best Practice Information Model (BIM), i.e., a special case of PIM that is used by multiple platforms. The goal is to return cross-platform results that satisfy the conditions of the query. When the query is formulated against the CIM than it can be answered by straightforward execution of the query against the Registry. Otherwise, if the query is formulated against a PIM (or, as special case, the BIM), the query is translated based on the mappings between the used PIM and other PIMs stored in the Registry using SPARQL query re-writing techniques to provide results from multiple platforms. The component's primary interface accepts SPARQL queries, but to simplify its usage, it may also offer multiple pre-defined parameterisable search queries (such as: search for a property, location, owner, etc.), which will be subsequently transformed to the SPARQL query using pre-prepared templates before executing it in Registry.
Provided functionalities	• Searches for resources which match a specific query across registered platforms and their resources.
Relation to other components	Registry: provides platform, enabler and resource-related information (PIM) needed to perform the search operation, provides all defined mappings for the used PIM and performs a SPARQL execution on the storage.

Table 7 Description of the Registration Handler component regarding the symbIoTe Information Model.

Component	**Registration Handler (RH)**
symbIoTe Domain	CLD
Description	This component will drive an IoT platform owner through the step of registering resources into the symbIoTe Core. Registration Handler

	should monitor platform resources so that it can register them into the symbIoTe (eco)system. These resources can be an IoT Device or a Composite IoT Service and must be described using a PIM which is aligned with the CIM. Registration Handler at least needs to provide all information from the CIM: IoT Device or Composite IoT Service description, Location and its properties and Observed Property description and name. Registration using the PIM relies on the Semantic Web technologies (PIM and resource instances description should be provided in the RDF format for the general version of the API of this component but for ease of usability additional interfaces hiding the Semantic Web technologies can be added later on) so it can become an obstacle for platform owner to join the symbIoTe ecosystem. For that purpose, the Registration Handler will provide the BIM, i.e. Best-Practice Information Model, which is a Platform-Specific Information Model (PIM) aligned with the CIM and provided by symbIoTe for making it more easy for platform owners to make their platforms symbIoTe-compliant. The purpose of the BIM is to facilitate the registration of IoT platforms that do not use Semantic Web technologies, so that platforms can only provide raw data necessary to fill the BIM template (i.e. to describe its resources). The data provided for BIM needs to include the mandatory elements of CIM, to enable the basic functionalities of the symbIoTe Core Services (search, check availability). Platforms that use BIM are automatically interoperable with other platforms using the BIM without the need to define any mapping.
Provided functionalities	• Registers resources to the symbIoTe core, virtual and physical, using the PIM • Updates resource status and unregistered resources
Relation to other components	Registry (within symbIoTe Core Services): stores data about resource in the PIM, assigns unique symbIoTe IDs and maintains information about current resource status.

Table 8 Description of the Resource Access Proxy component regarding the symbIoTe Information Model.

Component	**Resource Access Proxy (RAP)**
symbIoTe Domain	CLD
Description	This component enables symbIoTe-compliant access to resources within an IoT platform or (enabler acting as a platform). The data generated by IoT Services must be returned in accordance with the PIM. Since the PIM is an extension of the CIM, all requests made in-line with the CIM will return valid results (also valid for the platforms that are using the BIM). Also, the component will return results if the query is in-line with the used PIM, but it does not guarantee to return results if a request is made using some other PIM.
Provided functionalities	• Ensures formatting of data generated by resources in accordance with the PIM

Relation to other components	Application/Enabler: provides requested data

5.4 Semantic Mapping and SPARQL Query Re-Writing

As shown in Section 3.4, semantic mapping is not only about defining mapping between ontologies but also about using these mappings at runtime to mediate between them. This functionality is provided as an execution framework or a mediator as depicted in Figure 2. In symbIoTe this will be implemented using SPARQL query re-writing techniques [35] [36] [37]. How this works in detail is shown in Figure 17 which is an enhanced version of the sequence diagram describing the symbIoTe Core search functionality taken from D1.2 “Initial Report on System Requirements and Architecture” (Figure 17). The implications on the existing sequence diagram and changes to it are based on the introduction of SPARQL query re-writing as the main approach to address semantic interoperability. These are described in detail in hereafter.

Message 1 to 4 stay exactly the same, whereas for Message 5 we now know that the payload (referred to as *search* in the diagram) should be in the form of a SPARQL query under the assumption that all information models inside symbIoTe are formulated using RDF/OWL. We are aware that not all people are familiar with semantic technologies like SPARQL and that enforcing them to use semantic technologies can be quite an entry barrier. As SPARQL is only the direct interface to the Search Engine this doesn’t mean it must be the only one. To facilitate access to the Search Engine we are planning to also provide a simpler interface with a limited subset of functionality, e.g., a REST-based interface supporting multiple pre-defined parameterisable SPARQL queries. The rest of the messages stays the same except Message 11 which is now decomposed into multiple sub-messages and operations. First of all, the Search Engine asks the Registry for all platforms that use an information model, called $IM_{Platform}$ for which a mapping exists to from the information model the search SPARQL query Q is formulated in, called IM_{Query} (Message 11a). As a result, the Registry will provide a list of platforms $P_{1..n}$ together with the mapping definition $M_{1..n}$ mapping between $IM_{Platform\ 1..n}$ and IM_{Query}. Now for all found i = 1..n platforms the Search Engine executes the following steps:

- Re-write SPARQL query Q based on mapping M_i to query Q_i (Message 11b)
- Execute query Q_i against Registry (Message 11c) and receive query results QR_i (Message 11d)
- Re-write query result QR_i based on mapping M_i to query result $QR_{i,Query}$ which is compliant with the information model of the original query IM_{Query}

These steps are executed in parallel for each found platform. After all parallel tasks have finished, the re-written results are merged ($\bigcup_{i=1}^{n} QR_{i,Query}$) and returned (Message 12).

Following this approach, symbIoTe is able to answer queries formulated against any (registered) information model with data from all platforms using an information model which is mapped to the one used in the query which essentially enables semantic interoperability. It is to notice that there exist multiple different types of ontology mismatches as described in Section 3.4 of which only a subset will be supported for SPARQL query re-writing by symbIoTe. This means that PIMs which extended the CIM in such a different way that they

have serious types of mismatches will probably not be able to interoperate in practice. As it is still an open research question how to overcome those serious types of ontology mismatches symbIoTe envisions to define clear interfaces for the components related to solving this issue to provide a framework to encourage researches to address these problems and to make it easy for them to integrate their new solutions into symbIoTe. This will also allow to clearly separate the part of symbIoTe that is guaranteed to be successfully and completely implemented and the research part of symbIoTe which outcome is open to some degree as it is the case with every scientific research question. Furthermore, this separation with a clearly defined interface allows to be open to collaboration, especially with other projects inside the IoT-EPI which will also need to somehow address the problem of semantic interoperability.

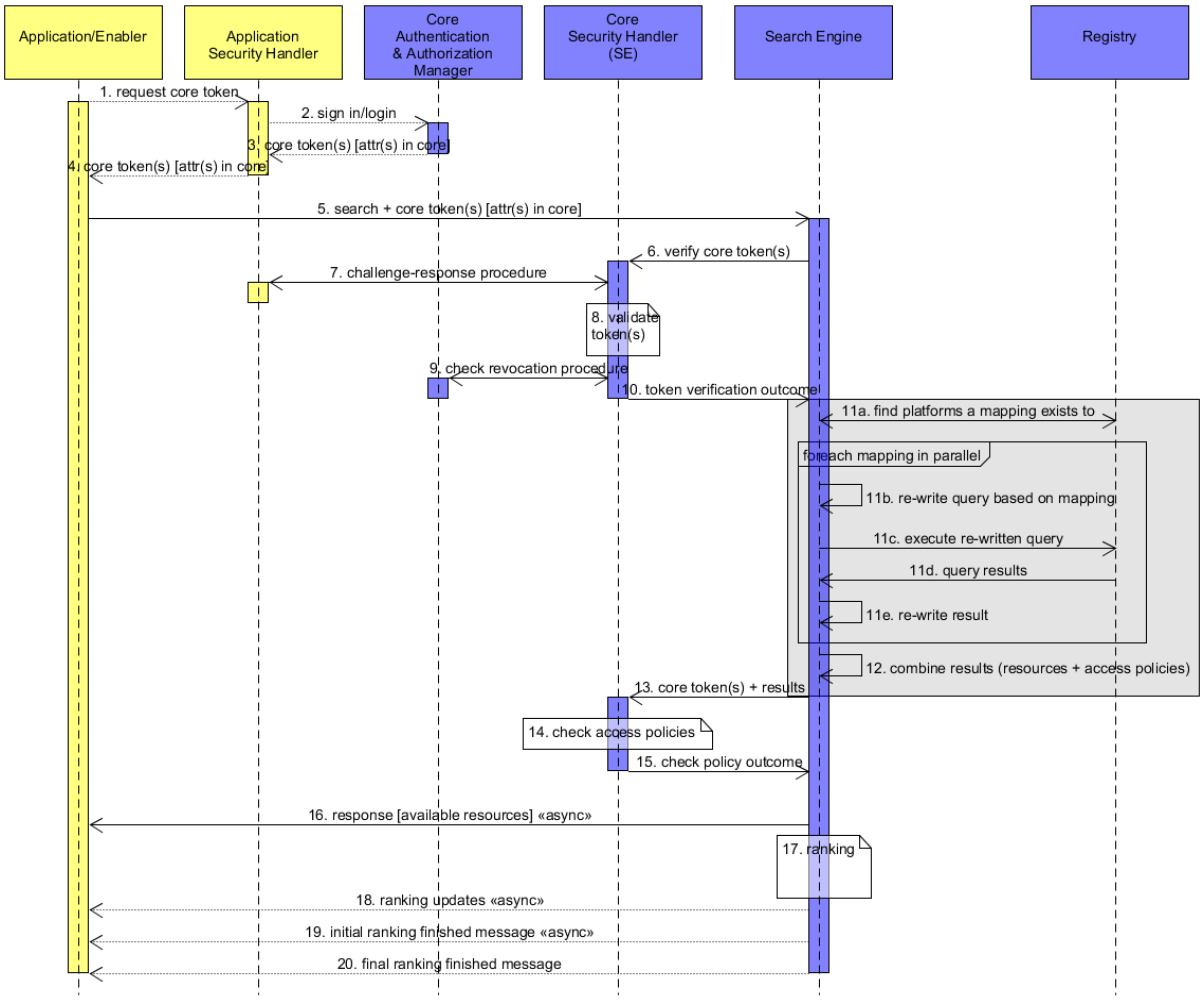


Figure 17 Enhanced sequence diagram from D1.2 “Initial Report on System Requirements and Architecture” (Figure 18) describing the search functionality with respect to SPARQL query re-writing.

5.5 Registering resources to the symbIoTe ecosystem

Hereafter we will describe the registration process from an alternative point of view, depicting communication between components and transmitted data. A basic sequence

diagram for resource registration is taken from deliverable D1.2 [34] (Figure 14). Figure 18 presents an enhanced sequence diagram for resource registration that presents two possible approaches of resource registration. Compared to the original diagram presented in deliverable D1.2 “Initial Report on System Requirements and Architecture”, only message 12 is presented in more details because it is the only one that is affected by specification of the symbIoTe information model. A platform owner has two possibilities to register resources to the symbIoTe ecosystem, one using the semantic resource description (depicted with message 12.1) and second describing resources in plain text and using that description to fill the Best Practice Information Model (depicted with the messages 12.2). The first approach assumes that a platform is using Semantic Web technologies, and a PIM needs to be aligned with the CIM. In this case, a platform owner needs to expose its used data model via its PIM extending and aligned with the CIM (by adding appropriate relations), and such extended PIM (i.e. platform and resource description in the RDF format) is provided to Registration Handler which will forward it to the Registry and perform registration. If a platform owner does not want to use Semantic Web technologies and/or thinks that the BIM fits his/her needs well, he can instead just re-use the BIM as his/her PIM (this is very close to the approach of a Core Information Model as introduced in Section 4.2.1). As the used information model is completely defined by the BIM it is then possible to register resources via any kind of description (e.g. JSON, XML or even plain-text) via the Registration Handler. In both cases, Registration Handler uses the semantic description of resources in RDF format when it is communicating with Registry to perform registration.

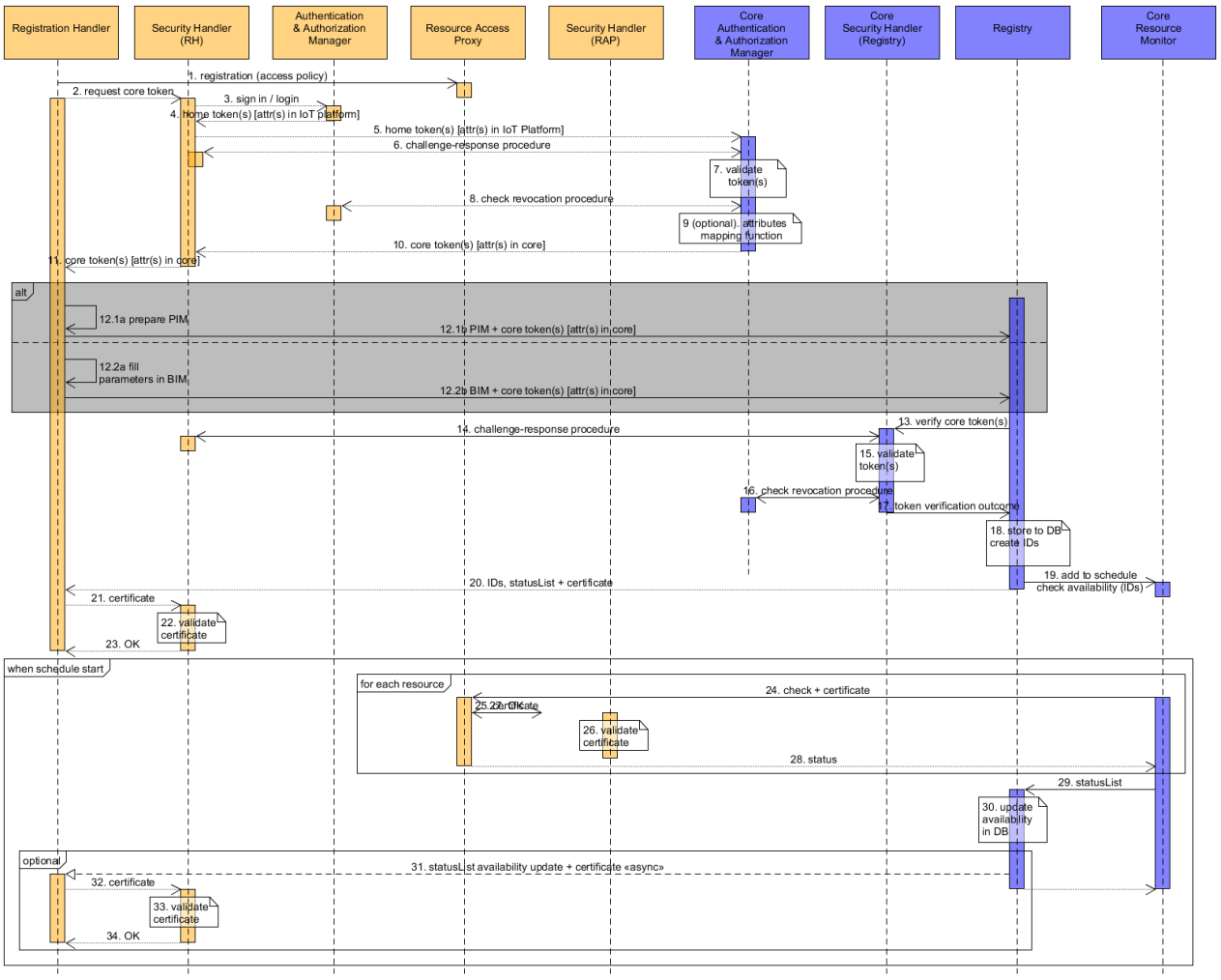


Figure 18 Enhanced sequence diagram from D1.2 “Initial Report on System Requirements and Architecture” (Figure 14) describing the resource registration.

6 Conclusions and Next Steps

In this deliverable we presented the problem of semantic interoperability between multiple heterogeneous IoT platforms together with a formal analysis of the solution domain to this problem. We highlighted five different possible approaches to achieve semantic interoperability ranging from a “monolithic” approach where all platforms use a single completely agreed upon information model to a “distributed” approach where each platform defines their own information model and they can only exchange information via mappings between those information models. In addition, we have also presented the advantages and disadvantages of the proposed approached. We presented that symbIoTe is aware that the best suitable approach would be the Core Information Model with Extensions approach but as it may render too complex for implementation, symbIoTe will consider the Best Practice Information Model approach with a Best Practice Information Model aligned with a Core Information Model as a next feasible step.

As semantics is essential for IoT platform interoperability (as explained in Section 3.1) we decided to build upon semantic technologies, e.g., RDF to describe information models. In Section 5.1 and its subsections we presented the symbIoTe Information Model (SIM) consisting of the Core Information Model (CIM) and the Meta Information Model (MIM) together with their design rationales. In addition, considerations on how this impacts the symbIoTe architecture is presented in D1.2 “Initial Report on System Requirements and Architecture” and also details on how SPARQL query re-writing can be used to solve the issue of semantic interoperability are provided in Section 5.3 and Section 5.4.

We have identified multiple domains where future work is needed. As a next step, we will keep up with the iterative development of the CIM which will focus on the different modelling needs of mobile vs. stationary sensors as well as in-situ vs. remote observations and also actuators will be part of a next version of the CIM.

Regarding SPARQL query re-writing as an approach to solve the problem of semantic interoperability we will focus on clearly separating the core functionality of symbIoTe from the research questions so that the success of the project is not dependant on the outcome of the research questions as real scientific research questions are never guaranteed to be completely solvable. This will also enable us to provide a partial solution to the research question of SPARQL query re-writing which will probably look like we’re providing this functionality to a limited subset of identified ontology mismatches. Therefore, a part of the research challenge will be to do an analysis of existing ontology mismatch classifications and to decide which of these can and will be supported within symbIoTe.

7 References

[1] Network Centric Operations Industry Consortium, „NCOIC, "SCOPE",“ 2008.

[2] M. Jacoby, A. Antonic, K. Kreiner, R. Lapacz und J. Pielorz, „Semantic Interoperability as Key to IoT Platform Federation,“ *Interoperability and Open-Source Solutions for the Internet of Things,* Forthcoming 2017.

[3] R. Herzog, M. Jacoby und I. Podnar Zarko, „Semantic interoperability in IoT-based automation infrastructures,“ *at-Automatisierungstechnik,* Bd. 64, Nr. 9, pp. 742-749, 2016.

[4] „Merriam-Webster,“ 06 04 2016. [Online]. Available: http://www.merriam-webster.com/dictionary/semantics.

[5] T. R. Gruber, „A translation approach to portable ontology specifications.,“ in *Knowledge acquisition 5.2*, 1933, pp. 199-200.

[6] A. Tolk und J. A. Muguira, „The levels of conceptual interoperability model,“ in *Proceedings of the 2003 Fall Simulation Interoperability Workshop*, Citeseer, 2003, pp. 1-11.

[7] „Wikipedia,“ [Online]. Available: https://en.wikipedia.org/wiki/ Conceptual_interoperability#Levels_of_conceptual_interoperability. [Zugriff am 06 04 2016].

[8] T. Berners-Lee und L. Ora, „The semantic web,“ *Scientific american,* pp. 28-37, 2001.

[9] W3C, „Vocabularies,“ [Online]. Available: https://www.w3.org/standards/semanticweb/ontology. [Zugriff am 06 10 2016].

[10] P. Visser, D. M. Jones, T. J. Bench-Capon und M. J. Shave, „Assessing heterogeneity by classifying ontology mismatches,“ in *Proceedings of the FOIS*, 1998.

[11] P. Visser, D. M. Jones, T. J. Bench-Capon und M. J. Shave, „An analysis of ontology mismatches; heterogeneity versus interoperability,“ in *AAAI Spring Symposium on Ontological Engineering*, Stanford CA., USA, 1997.

[12] M. Klein, „Combining and relating ontologies: an analysis of problems and solutions,“ in *IJCAI-2001 Workshop on ontologies and information sharing*, 2001.

[13] F. Scharffe, O. Zamazal und D. Fensel, „Ontology alignment design patterns,“ *Knowledge and Information Systems,* 2014.

[14] M. Rebstock, J. Fengel und H. Paulheim, Ontologies-based business integration, Springer Science & Business Media, 2008.

[15] J. Euzenat, „An API for ontology alignment,“ in *International Semantic Web Conference*, Springer, 2004, pp. 698-712.

[16] J. Euzenat, F. Scharffe und A. Zimmermann, „Expressive alignment language and implementation,“ 2007.

[17] P. Bouquet, F. Giunchiglia, F. Van Harmelen, L. Serafini und H. Stuckenschmidt, „C-OWL: Contextualizing ontologies,“ in *International Semantic Web Conference*, Springer, 2003, pp. 164-179.

[18] I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof und M. Dean, „SWRL: A semantic web rule language combining OWL and RuleML,“ *W3C Member submission,* Bd. 21, p. 79, 2004.

[19] A. Maedche, B. Motik, N. Silva und R. Volz, „MAFRA—An Ontology MApping FRAmework in the Context of the Semantic Web," in *Workshop on Ontology Transformation at ECAI-2002*, 2002.

[20] P. Shvaiko und J. Euzenat, „A survey of schema-based matching approaches," in *Journal on data semantics IV*, Springer, 2005, pp. 146-171.

[21] N. Choi, I.-Y. Song und H. Han, „A survey on ontology mapping," *ACM Sigmod Record,* 2006.

[22] E. Rahm und P. A. Bernstein, „A survey of approaches to automatic schema matching," *the VLDB Journal,* Bd. 10, Nr. 4, pp. 334-350, 2001.

[23] B. T. Le, R. Dieng-Kuntz und F. Gandon, „On ontology matching problems," *ICEIS (4),* pp. 236-243, 2004.

[24] M. Granitzer, V. Sabol, K. W. Onn, D. Lukose und K. Tochtermann, „Ontology alignment—a survey with focus on visually supported semi-automatic techniques," *Future Internet,* Bd. 2, Nr. 3, pp. 238-258, 2010.

[25] Á. Siciliaa, G. Nemirovskib und A. Nolleb, *Map-On: A web-based editor for visual ontology mapping.*

[26] S. Massmann, S. Raunich, D. Aumüller, P. Arnold und E. Rahm, „Evolution of the COMA match system," in *Proceedings of the 6th International Conference on Ontology Matching-Volume 814*, 2011, pp. 49-60.

[27] M. Kerrigan und A. Mocan, „The web service modeling toolkit," in *European Semantic Web Conference*, Springer, 2008, pp. 812-816.

[28] H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann und S. Hübner, „Ontology-based integration of information-a survey of existing approaches," in *IJCAI-01 workshop: ontologies and information sharing*, 2001.

[29] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson und A. Herzog, „The SSN ontology of the W3C semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web,* 2012.

[30] oneM2M Partners Type 1, „oneM2M Base Ontology," 2016.

[31] N. F. Noy, D. L. McGuinness und others, *Ontology development 101: A guide to creating your first ontology,* Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880, Stanford, CA, 2001.

[32] Open Geospatial Consortium, „OGC SensorThings API Part 1: Sensing," Version 1.0, 2016.

[33] J. David, J. Euzenat, S. Francois und C. Trojahn dos Santos, „The alignment API 4.0," *Semantic web,* Bd. 2, Nr. 1, pp. 3-10, 2011.

[34] s. Project, „D1.2: Initial Report on System Requirements and Architecture," 2016.

[35] G. Correndo, M. Salvadores, I. Millard, H. Glaser und N. Shadbolt, „SPARQL query rewriting for implementing data integration over linked data," in *Proceedings of the 2010 EDBT/ICDT Workshops*, ACM, 2010, p. 4.

[36] M. Konstantinos, N. Bikakis, N. Gioldasis und S. Christodoulakis, „SPARQL-RW: transparent query access over mapped RDF data sources," in *Proceedings of the 15th International Conference on Extending Database Technology*, ACM, 2012, pp. 610-613.

[37] B. Quilitz und U. Leser, „Querying distributed RDF data sources with SPARQL,“ in *European Semantic Web Conference*, Springer, 2008, pp. 524-538.

[38] R. Studer, V. R. Benjamins und D. Fensel, „Knowledge engineering: principles and methods,“ *Data & knowledge engineering,* 1998.

8 Acronyms

API	Application Programming Interface
BIM	Best Practice Information Model
C-OWL	Context OWL
CIM	Core Information Model
EDOAL	Expressive and Declarative Ontology Alignment Language
IAM	Identity & Access Management
ID	Identifier
IM	Information Model
IoT	Internet of Things
IoT-EPI	IoT-European Platforms Initiative
JSON	JavaScript Object Notation
JSON-LD	JSON-based Serialization for Linked Data
KES	Knowledge Engineering System
LCIM	Levels of Conceptual Interoperability Model
MAFRA	Ontology MApping FRAmework
MIM	Meta Information Model
N3	Notation3
OData	Open Data Protocol
OWL	Web Ontology Language
PIM	Platform-Specific Information Model
QoS	Quality of Service
QR	Query Results
RDF	Resource Description Framework
RDFa	Resource Description Framework in Attributes
RDFS	RDF Schema
REST	Representational State Transfer
SE	Core Security Handler
SIM	symbIoTe Information Model
SLA	Service Level Agreement
SPARQL	SPARQL Protocol and RDF Query Language
SPIN	SPARQL Inferencing Notation
SQL	Structured Query Language
SSN	Semantic Sensor Network

SWRL	Semantic Web Rule Language
symbIoTe	Symbiosis of Smart Objects across IoT Environments
Turtle	Terse RDF Triple Language
UoM	Units of Measurements
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WGS	World Geodetic System
WKT	Well-Known Text
XML	Extensible Markup Language
XSD	XML Schema Definition