Enabling the business-based
Internet of Things and Services

(FP7 257852)

D4.2 Knowledge representation formalism analysis

Published by the ebbits Consortium

Dissemination Level: Public

Project co-funded by the European Commission within the 7th Framework Programme
Objective ICT-2009.1.3: Internet of Things and Enterprise environments

Document version: 1.0

Submission date: 1.4.2011

Document control page

Document file:	D4.2 Knowledge representation formalism analysis.doc
Document version:	1.0
Document owner:	Jan Hreno (TUK)
Work package:	WP4 – Semantic Knowledge Infrastructure
Task:	T4.5 – Knowledge creation analysis
Deliverable type:	P
Document status:	☒ approved by the document owner for internal review ☒ approved for submission to the EC

Document history:

Version	**Author(s)**	**Date**	**Summary of changes made**
0.0	Jan Hreno(TUK)	2011-02-01	Initial TOC
0.1	Premchand Nutakki (SAP), Martin Knechtel (SAP)	2011-03-17	Sections 3.2 - 3.6
0.2	Karol Furdik (IS)	2011-03-20	Section 3.9 on semantic web services
0.3	Jan Hreno	2011-03-20	Section 3.7,3.8, 4
0.4	Karol Furdik	2011-03-23	Section 3.9 finished, Section 4.3
0.5	Jan Hreno	2011-03-23	Introduction, Conclusion, Executive summary
0.6	Tomas Sabol (TUK)	2011-03-23	Overall corrections
1.0	Jan Hreno, Premchand Nutakki, Karol Furdik	2011-03-31	Final corrections
		2011-04-01	Final version submitted to the European Commission

Internal review history:

Reviewed by	**Date**	**Summary of comments**
Riccardo Tomasi (ISMB)	2011-03-28	Approved with comments
Matts Ahlsén (CNET)	2011-03-27	Approved with comments

Index:

1. Executive summary

This deliverable contributes to the T4.5 task of the project.
The deliverable analyses the existing knowledge representation formalisms from different aspects, including:

- Device and service modelling capability
- Language complexity
- Human usability and readability
- Machine usability

In the first section, an overview of the knowledge representation formalisms in historical context is presented, along with an introduction to the most used formalisms. The following section is the core section of the document and contains: an analysis of formalisms for semantic web services; a description of human and machine optimized syntaxes and formalisms; an introduction to simple knowledge representation methods suitable for devices; a comparison of several formalisms used for knowledge representation and for querying the model. In the final section, good practises for knowledge modelling in relation to a choice of proper formalism(s) are proposed for the ebbits project.

The development of this deliverable was led by TUK with contribution from SAP and IS. These partners were previously involved in various development and R&D projects, where semantic technologies, including knowledge storing, sharing and reusing, were employed.

2. Introduction

2.1 Purpose, context and scope of this deliverable

The purpose of this deliverable is to provide an overview and critical evaluation of existing knowledge formalisms and to analyse how a decision to select a proper formalism can influence the ebbits platform and the use cases. For that purpose different formalisms will be analysed, from basic knowledge representation formalisms like RDF, OWL to specific semantic web service representation formalisms and formalisms used in knowledge reasoning and querying. Rather than going too deep into particular formalisms specifications, which are described in several freely available documents and tutorials[1,2,3,4,5,6], this document summarizes only the main properties and characteristics of these formalisms in the state of the art chapter. In the following chapter a relation to the ebbits use cases is analysed. Some examples, related to a preliminary ebbits automotive use case, are presented. These examples use different formalisms to represent knowledge about a chosen device. Finally the summarization chapter shows consequences of formalisms usage on the ebbits system.
The deliverable is delivered within the T4.5 Task ”Knowledge creation analysis” and this deliverable will be followed (after two months) by the deliverable D4.3 "Coverage and scope definition of a semantic knowledge model”. The task is intended to form a common understanding base of semantic technologies between partners for the future tasks where knowledge models will be developed (T4.6, T4.1-T4.4).
The deliverable is addressing problems that have to be understood by the whole consortium and thus is intended to be used by developers as well as the user partners. As it is delivered as a public deliverable, wider professional community can also learn about the project approach to the use of the knowledge representation standards.

2.2 Background

The goal of Task 4.5 according to the DOW states as follows "*This task will analyse the required scope and coverage of the semantic model, specifically for the use cases in ebbits. Semantic interoperability of devices and information systems’ resources needs a common defined terminology. One way to provide this is to adhere to standard interaction protocols and data formats. In fields, where such a standardisation does not exist since interaction mechanisms and architecture is in its innovative structure not yet covered by existing standards, a shared semantic model helps out.*
Most often, an ontology is used to store a formal representation of a shared conceptualization. ebbits needs a semantic model in order to allow for semantic interoperability. In this task we will analyse the required scope and size of the semantic model in order to prepare its creation systematically. Based on the of knowledge representation formalism analysis carried out, IS will propose coverage and scope definition of the semantic knowledge model, inputs and comments will be provided by TUK. ISMB will contribute to the definition of solutions enabling semantic interoperability between physical devices and information systems.”

[1] XML tutorials http://www.w3schools.com/xml/
[2] RDF tutorials http://www.w3schools.com/rdf/
[3] Protégé OWL tutorial http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/
[4] TopBraid Getting Started Guide http://www.topquadrant.com/composer/docs/TBC-Getting-Started-Guide.pdf
[5] SPARQL tutorial http://jena.sourceforge.net/ARQ/Tutorial/
[6] Describing Web services using OWL-S and http://www.ai.sri.com/daml/services/owl-s/1.1/owl-s-wsdl.html

3. State of the art of knowledge representation formalisms

3.1 Introduction

formalism (Philosophy / Logic)
the notation, and its structure,
in which information is expressed
(Collins English Dictionary
HarperCollins Publishers 2003)

This document attempts to describe and analyse notations and structures that are used in areas of relevance to ebbits. These formalisms can be used for storing, retrieving, transforming, reasoning, querying or presenting the knowledge.

3.2 Semantic Networks and Frames:

Description Logics (DLs) evolved from the early knowledge representation (KR) formalisms like *Semantic Networks* and *Frames*, both of them introducing the notion of classes of individuals and relations between the classes.

In Semantic Networks (Quillian 1967), a class or an individual is represented by a vertex and a relation by a labelled edge (Figure 1). The special *is-a* relation relates two classes or an individual and a class. Relations are inherited along *is-a* edges.

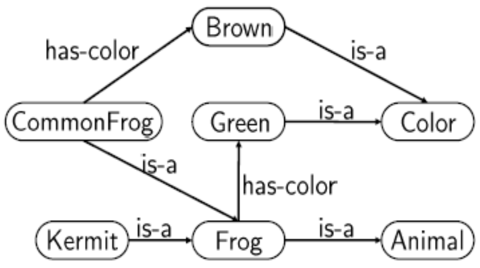


Figure 1 An example Semantic Network

In the Frame systems (Minski 1981), classes are implemented as Frames, where each Frame has a name, a collection of more general Frames, and a collection of slots. The slots specify relations to other classes, similar to the edges in the Semantic Networks.

The problem of both formalisms, Semantic Networks as well as Frames, is a lack of formally defined semantics. Meaning of a given knowledge representation is left to the intuition of the programmer who builds a reasoner. Consequently for the *is-a* relation there are at least six different meanings for the *is-a* relation between two classes, and at least four different meanings for the *is-a* relation between an individual and a class (Brachman 1983).

For Semantic Networks and Frames, the semantics introduced in (Schubert, Goebel and Cercone 1979; Hayes 1979) employed a relatively small fragment of the first-order logic. Based on these KR formalisms, *logic-based concept languages* were developed, which are known as *Description Logics.*

The Description Logics (Baader et al. 2003) are today embodied in many knowledge-based systems and are used for development of various real-life applications. The most popular application is Semantic Web. The W3C developed and recommended OWL as a standard ontology language for the Semantic Web (Motik, Patel-Schneider and Parsia 2009), and DLs provide its basis.

3.3 DLs in Semantic Web

Already some time ago it has been realized that the Web could benefit from making its content available in a machine processable form, which would enable computers to interpret the data. While the Web language HTML is focused on presentation and text formatting rather than content, languages such as XML do add some support for capturing the meaning of the Web content. The Semantic Web has been envisioned as an evolution from a linked document repository to a platform where "information is given well-defined meaning, better enabling computers and people to work in cooperation" (Berners-Lee, Hendler and Lassila 2001) and, to limit the scope, which "will enable machines to COMPREHEND (original emphasis) semantic documents and data, not human speech and writings" (Berners-Lee, Hendler and Lassila 2001).

This is to be achieved by augmenting the existing layout information with semantic annotations that add descriptive terms to the Web content, with the meaning of such terms being defined in ontologies. The DARPA Agent Markup Language (DAML) and Ontology Inference Layer (OIL) ontology languages for the Semantic Web are syntactic variants of DLs (Horrocks 2002) and have been the starting point for the W3C Web Ontology Working Group. The Working Group finished its work in 2004 with the publication of the OWL standard (Bechhofer et al. 2004). In 2007 the W3C OWL Working Group began to work on refinements and extensions to OWL, and finished with the publication of the OWL2 standard (Motik, Patel-Schneider and Parsia 2009) in 2009.

3.4 Explanation and Debugging of Inferences

Similarly to writing large software systems, building large-scale ontologies is error-prone. An ontology might imply unexpected or even undesired consequences. A real-world example of an unintended consequence is the subsumption relationship "amputation of finger is an amputation of arm", which follows from the SNOMED CT (Systematized Nomenclature of Medicine Clinical Terms) ontology (Suntisrivaraporn 2008). However, finding a reason, i.e. a set of related axioms, by just looking at 400,000 axioms manually, is not realistic. Humans are usually not good in seeing implications from large sets of axioms. So we can define an *explanation* for a consequence following from the ontology as a minimal set of axioms (**MinA)** from the ontology from which the consequence still follows. The dual notion of a MinA is that of a **diagnosis**, a minimal set of axioms, which need to be removed so that a consequence does not follow anymore. By making use of the two concepts above we can perform debugging of an inference by finding explanations and removing the cause.

3.5 Most used knowledge representation formalisms

Most of the content below is from (Dau et al. 2009). Various knowledge representation formalisms are used today, while each system has different requirements like expressivity, performance etc. So, choosing an appropriate formalism is also an important task. For sake of example, a simple ebbits-oriented case study has been described using some of the knowledge representation formalisms in Section 4. Thus, we provide a brief introduction to the commonly used KR formalisms:

3.5.1 RDFS

RDFS or RDF Schema (Resource Description Framework Schema - Brickley and Guha 2004) is an extensible knowledge representation language, providing basic elements for the description of ontologies, called also RDF vocabularies, intended to structure RDF resources. The RDF is a family of W3C specifications originally designed as a metadata data model which slowly became a general method for conceptual description or modeling of information that is implemented in web resources and the RDFS vocabulary builds on the limited vocabulary of RDF. RDFS also provides mechanisms for describing related resources and the relationships between these resources. These resources are used to determine characteristics of other resources, such as the domains and ranges of properties. RDF Schema vocabulary descriptions are written in RDF. (Manola and Miller 2004) describes some actual deployed RDF applications, showing how RDF supports various real-world requirements to represent and manipulate information about a wide variety of things.

3.5.2 OWL 1.0 DL

The Web Ontology Language OWL (Dean and Schreiber 2004) is a semantic markup language for publishing and sharing ontologies on the World Wide Web. The OWL is developed as a vocabulary extension of RDF and is derived from the DAML+OIL Web Ontology Language and is designed to facilitate ontology development and sharing via the Web, with the ultimate goal of making Web content more accessible to machines. OWL DL (where DL stands for "Description Logic") was designed to support the existing Description Logic business segment and to provide a language subset that has desirable computational properties for reasoning systems. The OWL DL is obtained by placing some constraints on the use of the OWL language constructs so that a decidable reasoning procedure can exist for an OWL reasoner. There are numerous tools to generate an ontology (Protege, TopBraid, Swoop, OntoStudio, Neon) and several inference algorithms (Fact++, RacerPro, OntoBroker (KAON), pellet) were implemented. Since OWL is a W3C standard, it is particularly suitable for the integration of multiple, distributed resources on the web.

3.5.3 OWL 2

The W3C OWL 2 Web Ontology Language (OWL) is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things (OWL 2009, Hitzler et al. 2009). OWL 2 ontology documents describe information in terms of classes, properties, individuals, and data values the relationships of which can be described by a number of features. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents. The OWL 2 is compatible with the prior OWL 1 ontology language. The main enhancements in comparison to the OWL 1 are (Hitzler et al. 2009):

- Property chains
- Asymmetric, reflexive, disjoint properties
- Richer data types, data ranges
- Qualified cardinality restrictions
- Enhanced annotation capabilities
- New profiles and a new syntax
- Keys

Also, some of the constraints applicable to OWL DL have been relaxed.

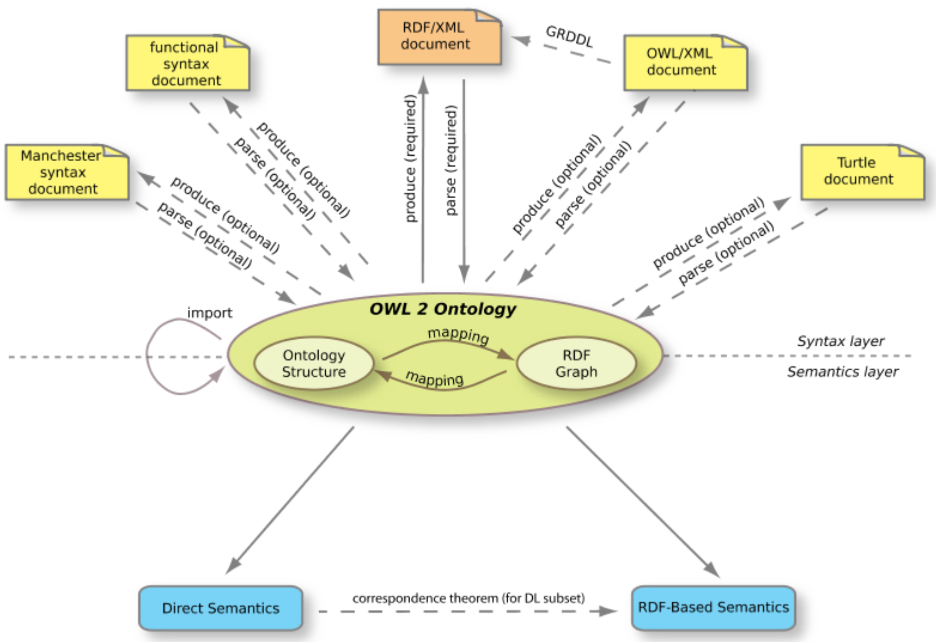


Figure 2 Structure of OWL 2

Figure 2 (Hitzler et al. 2009) gives an overview of the OWL 2 language, showing its main building blocks and how they relate to each other. The ellipse in the center represents the abstract notion of an ontology, which can be thought of either as an abstract structure or as an RDF graph. At the top there are various concrete syntaxes that can be used to serialize and exchange ontologies. At the bottom there are the two semantic specifications that define the meaning of the OWL 2 ontologies.

3.5.4 EL++ (OWL 2 EL)

The OWL 2 EL is a profile of OWL 2 (Motik et al. 2009), i.e. a sub-language, characterized by lower expressiveness that makes it more comprehensible for humans and enables more efficient inference algorithms (Dau et al. 2009). The class constructors are restricted to conjunction, existential restriction and nominal (singleton class containing one individual). Property can be declared as transitive or reflexive and limitations on the range and domain of a property are possible, as is the use of data types. Classes and properties can be declared as sub-classes (or properties) of other classes and properties. The OWL 2 EL is a formalism that

- is particularly suitable for applications employing ontologies that define very large numbers of classes and/or properties,
- captures the expressive power used by many such ontologies, and
- for which the ontology consistency, class expression subsumption, and instance checking can be decided in polynomial time.

For example, the OWL 2 EL provides class constructors that are sufficient to express very large biomedical ontology SNOMED CT.

3.5.5 ELP

The ELP is a description logic based on the OWL 2, which has been developed only recently in (Krötzsch, Rudolph and Hitzler 2008) and special attention is paid to it due to the fact that most powerful rules can be expressed, but the logic is still decidable in polynomial time. In particular, the ELP is a decidable fragment of the (undecidable) Semantic Web Rule Language (SWRL). The ELP includes:

- Description Logic Programs (DLP - Grosof et al. 2003), also OWL 2 RL (see below).

- EL++, also OWL 2 EL.
- DL-safe Datalog.

In simple words, the ELP is the union of EL++ and DLP. While the ELP can be viewed as an extension of both formalisms, however, it limits interactions between the expressive features of either language and thus preserves polynomial time reasoning complexity (Dau et al. 2009). Since the ELP is a very new language proposal, there does not exist a sufficient support for the ELP with tools for modeling ontologies or for drawing conclusions in ontologies.

3.5.6 OWL 2 RL

The OWL 2 RL profile (Motik et al. 2009) is aimed at applications that require scalable reasoning without sacrificing too much expressive power. It is designed to accommodate both OWL 2 applications that can trade the full expressivity of the language for efficiency and RDF(S) applications that need some added expressivity from the OWL 2 (Dau et al. 2009). It enables implementation of polynomial time reasoning algorithms using rule-extended database technologies operating directly on RDF triples; it is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to operate directly on data in the form of RDF triples. The design of OWL 2 RL was inspired by the Description Logic Programs (DLP - Grosof et al. 2003) and pD* (Herman and Horst 2005). *T*(s,p,o) represents a generalized RDF triples with subject *s*, the predicate *p* and object *o*, in which empty nodes (bnode) and literals (literals) are allowed in all positions. Variables are denoted by a preceding question mark. The symbol *false* represents a contradiction: if it is derived, it means that the initial RDF graph contains inconsistencies. Rules are specified as universally quantified implications of the first order logic to a ternary predicate *T*.

3.5.7 F-Logic 1

The F-Logic is a deductive, object-oriented database language (Dau et al. 2009). Explicit factual knowledge is represented in the F-Logic by logic programs. This includes knowledge about objects, relations between objects and classes to which objects belong. Additionally, it allows the modeling of implicit, intentional knowledge in the form of rules and queries (Kifer, Lausen and Wu 1995). The basic building blocks of knowledge representation with the F-Logic are terms and predicates. A term is a constant, a function, or variable. Predicates represent atomic elements of knowledge and can be either *true* or *false*. Due to the object-oriented character of the F-logic, epistemological primitives for object-oriented modeling such as definition of subclass- and instance-of- relations and signature specifications for methods are also provided. It is used in a range of applications for information integration, question answering and semantic search.

3.5.8 F-Logic 2

The F-Logic 2 has been obtained by extending F-Logic 1 with many additional features (Dau et al. 2009). The syntax for attributes and relations is added and the F-Logic 2 also allows specification of cardinalities, quantification of variables in queries, rules and the syntax of aggregations has been changed as well. Compared with the original F-Logic more data types are supported. To use this in an appropriate manner, there are a variety of built-in features. This supports, for example, arithmetic or string operations. For a given attribute or a relation we can specify minimum and maximum values of an instance. Also, the F-Logic-2 syntax allows to specify for each attribute and each relation, whether it can be inherited by subclasses or not. This property is important in the context of meta-modeling.

3.6 Tabular comparison of knowledge representation formalisms that are mainly used

Most of the following content below is taken from (Dau et al. 2009). We will compare the following categories of properties amongst the above mentioned KR formalisms.

1. *Expressiveness*: This category describes various types of statements that are possible in the language.
2. *Modelling*: This category deals with modelling issues.
3. *Semantics*: This category deals with the details of the semantics of the language
4. *Infer*: This category deals with the derivation of new knowledge
5. *Query*: How are ontology queries supported?
6. *Web compliance*: Does the ontology language (and its query languages) follow the W3C recommendations?

The comparison is shown by ranking the abilities of the DLs in the range [0...4].
(Note: '0' specifies that the property can't be expressed by the formalism and value '4' specifies that the property can be very easily expressed by the formalism. '1' specifies that the property can be expressed but not in an effective way. Similarly, value '2' and '3' specifies that the properties are fairly and quite easily and efficiently expressible by the formalism respectively.)

	RDFS	OWL 1.0 DL	OWL 2	EL++	ELP	OWL 2 RL	F-LOGIC 1	F-LOGIC 2
EXPRESSIVENESS								
Binary relations	4	4	4	4	4	4	4	4
Higher order relations	2	1	1	1	1	1	4	4
Hierarchy of classes	4	4	4	4	4	4	4	4
Hierarchy of relations	4	4	4	4	4	4	3	3
Disjointness of classes	0	4	4	4	4	4	2	2
Negation of classes	0	4	4	0	0	4	2	2
Complete coverage by subclasses	0	4	4	0	0	3	2	2
Existential quantification	0	4	4	4	4	4	2	2
Universal quantification	1	4	4	0	0	4	2	2
Domain/Range restrictions	4	4	4	4	4	3	4	4
Qualifying number restrictions	0	3	4	0	0	2	4	4
Unsafe/stochastic facts	0	0	0	0	0	0	1	1
Rules	0	0	2	2	3	1	4	4
Operational definitions	0	0	0	0	0	2	4	4
MODELLING								
Understandability	4	2	2	3	3	2	3	3
Visualization	4	2	2	2	2	2	3	3
Support Tools	4	4	4	1	0	2	3	3
Existing Ontologies	4	2	2	2	0	0	2	0
Mappings	2	2	2	2	2	2	4	4
Built-ins	0	1	2	2	0	0	4	4
Documentation	2	4	4	4	4	4	4	4
Modularization	4	4	4	4	4	2	4	4
Meta-modelling	4	0	2	2	0	4	4	4
Relations with parameter	1	1	2	2	2	0	4	4
Context transformations	0	0	0	0	0	0	4	4
SEMANTICS								
Semantics based on FOL	2	4	4	4	4	4	4	4
Semantics is formal	4	4	4	4	4	4	3	3
Open/Closed world semantics	o	o	o	o	o	o	c	c

INFER								
Correctness	4	4	4	4	4	4	4	4
Completeness	2	4	4	4	4	3	4	4
Terminating	4	4	4	4	4	3	2	2
Works with uncertain facts	0	0	0	0	0	0	1	1
Consistency check	0	4	4	4	4	3	4	4
Generation of class hierarchy	4	4	4	4	4	4	4	4
Support Tools	4	4	4	1	1	2	4	4
QUERY								
Query languages	4	3	3	2	0	2	4	4
Expressiveness	4	4	4	4	0	2	4	4
Scalability for simple queries	4	4	4	4	4	2	3	3
Scalability for complex queries	4	2	2	4	4	2	3	3
Support of closing rules	4	2	2	4	4	2	4	4
Database access	4	2	2	2	2	2	4	4
Access to other sources	4	2	2	2	2	2	4	4
WEB COMPLIANCE	4	4	4	4	2	3	2	2

Table 3.1: Comparison of various KR formalisms that are mainly used (Dau et al. 2009)

From Table 3.1 we can select the required KR formalism based on our requirements.

3.7 Supporting formalisms

3.7.1 eXtensible Markup Language

eXtensible Markup Language (XML) is used for encoding documents in a form readable by machines. It is defined in the XML 1.0 Specification (XML 2008) produced by the W3C. XML was chosen as a main formalism to be used in semantic web initiative by using XML based RDF, RDFs, OWL formalisms. The use of XML as a base for the majority of currently popular knowledge representation formalisms (or possibility to transform these into an XML form) has very practical reasons. The XML is a universally accepted standard for structuring data especially if it has to be sent over the HTTP protocol. A number of Web services technologies (e.g. SOAP) are in fact based on manipulation and transfer of XML messages over the HTTP protocol. This also means that there is a plenty of tools available for XML manipulation and many developers are capable of working with these tools.

3.7.2 Resource identifiers

A Uniform Resource Identifier (URI) is a string of characters used to identify a name or a resource. The URI syntax is defined in a IETF Network Working Group RFC3986 document (Berners-Lee et al. 2005). URIs can be URLs (Locators) or URNs (Names). While URL is a name for location on the Internet, URN has similar syntax, but it can be used for naming of anything. The Internationalized Resource Identifier (IRI) is a generalization of the Uniform Resource Identifier (URI). While URIs are limited to a subset of the ASCII character set, IRIs may contain characters from the Universal Character Set. XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references. In the knowledge representations, mainly based on some subset of OWL or RDF, which are XML based, IRIs are used for namespace identification.

3.8 Query formalisms

Obtaining the particular knowledge from the knowledge representation is the elementary requirement for using the semantic technologies. Together with the RDF for knowledge storing, a query language SPARQL was developed, which is currently the W3C recommendation from 2008 (SPARQL 2008). The SPARQL is a query language for RDF, like SQL for databases and XQuery and XPath for XML.

The SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. The SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. The SPARQL also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be result sets or RDF graphs. With the growing need for integration of widely spread relational database systems with semantic technologies, a question of integrated querying of combined RDF+RDB+XML resources has been raised.

In (Elliot et al. 2009) an efficient not nested SQL is generated from the full SPARQL to be processed with database engine. In (Chebotko, Lu and Fotouhi 2009) a similar approach is proposed. The SQL produced there is semantically equivalent to the SPARQL input. A number of optimizations in order to produce simpler and more efficient SQL are presented. Benchmarks show that performance is comparable to native RDF storage systems. Triple stores using relational databases for storing the data translate SPARQL queries into SQL queries. On the other hand, the W3C RDB2RDF Working Group[7] attempts to provide a specification for a language to map relational data and relational schemas to RDF and OWL (called R2RML). In (Bikakis et al. 2009) the SPARQL2XQuery Framework is described, providing a formal mapping from OWL ontology to XML Schema and translating SPARQL to semantically equivalent XQuery. In (Perez, Arenas and Gutierrez 2009) authors study complexity of the evaluation of several fragments of the SPARQL language. According to them the SPARQL query evaluation is a PSPACE-complete problem (any problem that can be solved in a polynomial amount of space on a touring machine can be transformed into it in a polynomial time).

Some advantages of SPARQL are:

- SPARQL has strong support for querying with an unpredictable and unreliable structure. Variables may be used instead of the predicate position to query unknown relationships, OPTIONAL keyword enables querying relationships that may not occur in the data.
- SPARQL is built to support queries in a networked, web environment.

There are also disadvantages of using the SPARQL in comparison to SQL or XQuery:

- SPARQL is a rather recent language and has not so far a wide tool and system support, as for example SQL OR XQuery.
- It is more difficult to query transitive or hierarchical relations in the SPARQL.

3.9 Machine friendly syntaxes of knowledge serialization

For formalisms to be machine-readable different syntaxes are used. The XML based syntaxes of formalisms have the biggest set of available tools to be used to read from or write into these syntaxes. The XML based OWL/RDF syntaxes are recommended syntaxes for machines. Namespaces are used to distinguish individual knowledge bases. So-called SW Parsers are used to read from textual documents containing knowledge models serialized in different formalisms. These are used to import knowledge data into knowledge storages (triple stores). Serializers are SW tools used to export data from the knowledge store to a textual representation. By using the recommended standards for serialization and parsing (mostly based on W3C recommendations) users ensure that their knowledge stores are compatible and interoperable with others. The same is valid also for exchange of data inside semantically enhanced intranet systems, where using open standards ensures easy upgradeability and independence on particular products or solutions. Recommended formalisms are used not only for storing, importing or exporting of data within isolated system, but also for exchange of information between different systems connected on same network (usually Internet).

[7] http://www.w3.org/2001/sw/rdb2rdf/

3.10 Human friendly knowledge representation formalisms

Semantic technologies are intended not only to help computers to understand the meaning of data, but also to help humans to share the same understanding of the data. Formalisms for representation of the knowledge were developed to be understandable not only for computers, but also for human. By using parsers and serializers, we can translate between different formalisms, using one formalism for the exchange of information between machine and human and different one for communicating only between machines.

3.10.1 RDF Notation 3, Turtle, N-Triples

The Notation 3 (N3)[8] is a language, which is a compact and readable alternative to RDF's XML syntax. It has subsets, one of which is RDF 1.0 equivalent (Terse RDF Triple Language - Turtle[9]), and another one is RDF plus a form of RDF rules. Another subset of the N3 is the N-Triples[10] which is an RDF syntax for expressing RDF test cases and defining the correspondence between RDF/XML and the RDF abstract syntax.

3.10.2 OWL Syntaxes

The most known OWL syntaxes focused on user friendliness are the Functional and the Manchester OWL syntaxes. The Functional OWL syntax is a simple text base syntax. It is not intended to be used as a syntax for exchanging data, but rather used for transformation from structural specification into some concrete syntax. The Manchester OWL syntax is a user-friendly compact syntax for the OWL 2 ontologies; it is frame-based in contrast to the axiom-based syntaxes for the OWL 2.

3.10.3 OBO Format

The OBO Format[11], originally used for biomedical ontologies only, can express a subset of the description logic language OWL-DL 2.0, but in addition to that it has standard syntax for representing classes of meta-data like synonyms and references to publications. It is designed to be human readable and editable, easy to parse, easy to extend and to have minimal redundancy.

3.10.4 Graphical representation of knowledge

Human in comparison to machine is capable of very efficient visual perception and understanding. This is why there exists one very important way of presenting knowledge representation to a human - graphical representation. Even if there is currently no generally accepted standardization effort behind different graphical representations for ontologies, there are some similarities between existing solutions. They are mostly based on some kind of oriented graph visualisation (usually a tree), where nodes are representing objects or groups of objects and connections between nodes represent relation of connected objects. These graphs are based on an idea of semantic net (Richens 1956). Visual tools are used mainly for design of knowledge models, but also for presenting these to non knowledge worker persons. In (Katifori et al. 2006, 2007) authors try to determine advantages and disadvantages of different ontology visualisation methods in the Protégé[12] ontology editor and their suitability for various ontologies and user groups.

[8] http://www.w3.org/DesignIssues/Notation3
[9] http://www.w3.org/TeamSubmission/turtle/
[10] http://www.w3.org/TR/rdf-testcases/
[11] http://www.geneontology.org/GO.format.obo-1_2.shtml
[12] http://protege.stanford.edu/

3.11 Semantic web service formalisms

Semantic web service technologies build on the notion of the basic web service, which could be considered as one of the main building blocks of the Service Oriented Architecture (SOA). This progressive method of developing distributed information systems enables loose coupling of system elements, i.e. various functional modules that provide and/or consume shared or private information resources, in a transparent way, by means of standardised web service interfaces. Software systems adhering to the SOA paradigm provide several important functionalities achieved by the web services (Papazoglou 2003), namely:

- Service publication – service descriptions are created in a suitable format and are published according to pre-defined standards in well-known locations;
- Service discovery – information retrieval techniques are employed on the published service descriptions;
- Service selection – results of the discovery process are filtered according to the specified query parameters;
- Service binding – the interface and transport protocol of a service is specified and the service is ready to be executed.

According to (Cerami 2002), the definition of a Web Service is rather general: "A web service is any service that is available over the Internet, uses a standardised XML messaging system, and is not tied to any operating system or programming language". Additionally the authors postulate that web services shall be self-descriptive and discoverable. These features drove streams of research efforts in this area as can be seen, for example, in (Curbera et al. 2002) or (Alonso et al. 2008). Nowadays, web services are considered a well-established, advanced and effective technological foundation for achieving interoperability between the elements of distributed information systems. The specifications and parameters of web services are standardised on the levels of service protocols, frameworks, and XML-based markup languages – see, for example, in (WSA 2007), ISO/IEC 24824-2:2006, ISO/IEC 29361-3:2008, etc. In general, three basic aspects of web services are fundamentally important:

- XML messaging system – most widely used implementations of XML messaging are SOAP (Simple Object Access Protocol), XML-RPC (XMLRCP 2003) and REST (REpresentational State Transfer) (Fielding 2000). SOAP is a lightweight protocol intended for exchanging structured information in a decentralised, distributed environment (Gudgin et al. 2007). SOAP basically works by tunnelling XML-formatted messages via Internet protocols (SMTP, HTTP(S)) and is easy for implementation in existing infrastructures. XML-RPC simplifies SOAP approach by a restriction to HTTP(S), where the XML content is transferred in a POST message. REST further simplifies the process by usage of intuitive request format directly based on the HTTP methods of GET, POST, PUT and DELETE. Besides the XML-based data, REST can rely on different languages for content representation (e.g. JSON or YAML). Nowadays, REST becomes very popular solution with SOA and many technologies start to support this standard.
- Self-description of services – important for description of services in terms of available functions with expected input. Various standards have been created during time which can be grouped into two categories: a) Fundamental web service descriptions, which are based on WSDL (currently in revision 2.0) (Chinnici et al. 2007), and b) Semantic web service descriptions (Farrell and Lausen 2007) that additionally annotate service descriptions in a semantic manner. Semantic web languages such as OWL-S (Semantic markup for Web Services – Ontology Web Language) or WSML (Web Service Modelling Language) are available for semantic annotations of web services. This semantic enhancement enables automatic discovery, invocation, composition and interoperation of heterogeneous web services.
- Discoverability – process of searching for services and retrieving information about them. The UDDI (Universal Description, Discovery and Integration) standard (Clement et al.

2004) is typically used for the discovery of "general" web services (i.e. without an additional semantic information). Implementers of UDDI can either be clients or servers, so called registries, which store various information on web services - business entity (publisher information), business service (descriptive information about service), binding template (technical information about service), tModel (generic container to summarise all technical information on the services). In practice, however, crawling via common search engines is used for service discovery more frequently than by registries (Al-Masri and Mahmoud 2008).

Web services can be composed into chains by means of pre-defined or ad-hoc calculated workflow sequences, based on IOPE (i.e. inputs, outputs, preconditions, effects) characteristics of elementary services. Related to the composition of web services, the notions of service choreography and orchestration (Reynolds 2006) can be specified as follows:

- Orchestration relates to the (order of) execution of web services in a scope of specific business processes. WS-BPEL (Jordan and Evdemon 2007) is a language for defining workflow sequences in processes that can be executed on an orchestration engine.
- Choreography is related to a description of externally observable interactions between web services. WS-CDL (Kavantzas et al. 2005) is a formal language for describing multi-party contracts and can be seen as an extension of WSDL: WSDL describes web services interfaces, WS-CDL describes collaborations between web services.

The ability to compose web services into complex workflow chains is enabled by technical standards and specifications such as WSDL, SOAP, UDDI, etc. However, meaningful exchange of inputs and outputs between chained services needs to be supported on the semantic level as well. The semantic interoperability between possibly heterogeneous web services is achieved by enhancing the WSDL descriptions of web services with additional information. A survey of some of formal languages, which are most commonly used for semantic annotation of web services, is provided in following subsections.

3.11.1 SAWSDL

The Semantic Annotations for WSDL and XML Schema (SAWSDL) recommendation (Farrell and Lausen 2007) defines a set of extension attributes for WSDL, which allows an insertion of semantic descriptions for web services. While the syntactic descriptions of WSDL provide information about the structure of input and output messages of an interface and about how to invoke the service, semantic extension is needed to describe what a web service actually does. The SAWSDL specification defines how semantic annotation is accomplished using references to semantic models, e.g. ontologies. It provides mechanisms by which ontology concepts, typically defined outside the WSDL document, can be referenced from within WSDL and XML Schema components using semantic annotations.

The annotation mechanism of SAWSDL uses the abstract definition of servicers, which is represented in WSDL by Element Declaration, Type Definition, and Interface components. Such a semantic annotation of abstract part of the service definition consequently enables dynamic discovery, composition and invocation of services. The extension attributes defined by SAWSDL are as follows:

- the *modelReference* attribute specifies the association between a WSDL or XML Schema component and a concept in some semantic model;
- the *liftingSchemaMapping* and *loweringSchemaMapping* extension attributes are added to XML Schema element declarations and type definitions for specifying mappings between semantic data and XML.

Multiple semantic annotations are allowed for a single WSDL element in service descriptions. Both schema mappings and model references can contain multiple pointers - URIs that typically refer to concepts described in an external ontology. Multiple schema mappings are interpreted as alternatives whereas multiple model references are all applied in parallel. SAWSDL does not specify any other relationship between them.

3.11.2 OWL-S

The Semantic Markup for Web Services (OWL-S) is the OWL ontology for semantic description of web services (Martin et al. 2004). The structure of the OWL-S consists of a service profile for service discovery, a process model which supports composition of services, and a service grounding that associates profile and process concepts with the underlying service interfaces. Currently, the OWL-S is available in version 1.2 (Martin et al. 2008).

The class *ServiceProfile* of OWL-S ontology provides a superclass of every type of high-level description of the service. It defines functional properties that describe IOPEs of a service, as well as non-functional properties that describe semi-structured human-readable information for service discovery, e.g. service name, description and parameters which incorporates further requirements on the service capabilities (e.g. security, quality-of-service, geographical scope, etc.).

The class *ServiceModel* specifies ways of operating the service in a workflow structure with other services. The service is viewed as a process (represented by the Process sub-class of the ServiceModel), which defines the functional properties of the service (IOPEs) together with details of its constituent processes (if the service is a composite service). Functional properties of the service model can be shared with the service profile.

Interactions between services are represented by the class *ServiceGrounding*. It enables execution of the Web Service by binding the abstract concepts of the OWL-S profile and process model to concrete message formats and communication protocols. Although different message specifications are supported by OWL-S, the widely accepted WSDL is preferred as an initial grounding mechanism.

3.11.3 WSMO

The Web Service Modeling Ontology (WSMO) is a conceptual model that was specifically developed for describing semantic web services (deBruijn et al. 2005). The underlying ontological specification of WSMO consists of four major components - ontologies, goals, web services, and mediators:

```
Class wsmoTopLevelElement
      hasNonFunctionalProperties type nonFunctionalProperties
Class ontology sub-Class wsmoTopLevelElement
Class webService sub-Class wsmoTopLevelElement
Class goal sub-Class wsmoTopLevelElement
Class mediator sub-Class wsmoTopLevelElement
```

Ontologies provide an agreed common terminology, a formal semantics that can be used by all other components. WSMO specifies the following constituents as a part of the description of ontology: concepts, relations, functions, axioms, together with instances of concepts and relations, as well as non-functional properties, imported ontologies, and used mediators.

Goals specify objectives that a client might have when consulting a web service, i.e. functionalities that a web service should provide from the user perspective. The Goal element is characterized by a set of non-functional properties, imported ontologies, used mediators, the requested capability and the requested WSDL interface.

The *Web Service* elements are described by non-functional properties, references to imported ontologies, used mediators, and the behavioral aspects of web services that are represented by the capability and interface properties. The capability of a web service defines its functionality in terms of preconditions, postconditions, assumptions and effects, which are expressed by a set of axioms and shared variables. By means of the capability property, a web service may be linked to certain goals that are solved by the web service by means of referenced mediators. The interface of a web service provides further information on how the service functionality is achieved. It describes the behavior of the service for the client's point of view (i.e. service choreography) as well as the means of achieving overall functionality of the service in terms of cooperation with other services (service orchestration).

Mediators represent the elements that enable overcoming structural, semantic or conceptual mismatches that appear between the components that build up a WSMO description. Depending of the type of mediated components, four types of mediators are distinguished: 1) OOMediators (for resolving semantic mismatches between the source and the target ontologies); 2) GGMediators (for connecting goals into sub-goal hierarchies and resolving mismatches between goals); 3) WGMediators (for linking a goal to a web service); and 4) WWMediators (for connecting several web services into a collaboration structure).

All WSMO components are formalized using the Web Service Modeling Language (WSML), which is based on the description logic, first-order logic and logic programming formalisms (deBruijn et al. 2008). The WSMO framework is supported by the Web Service Modelling eXecution environment (WSMX), which serves as a reference implementation for WSMO (WSMX 2008).

4. Ebbits use cases analysis

4.1 Knowledge representation formalisms example

In order to clarify the differences between the knowledge representation formalisms, a simple device ontology describing the ebbits manufacturing scenario demo developed within the WP5 can be created. Only one scenario from two was selected, as the proposed simple model is generic enough to be used with a small modification for the other scenario as well. The ontology describes a water pump device, which is capable of being monitored for its current water and energy consumptions. Based on the Hydra project ontology (Kostelnik, Sarnovsky and Hreno 2009) a small fragment of the ontology modelling the device was prepared. Modelling has started with describing known facts by a few subject-predicate-object (triple) statements:

```
Pump is a Device
Pump has an ActualEnergyConsumption
Pump has an ActualWaterFlow
```

As it was described in the D7.2 deliverable, measuring values of the device can be better modelled as events based measurements. That enables splitting of the continuous measuring into reasonable event based discrete measurements. One real device named WaterPump1 was also added into the ontology, which is an instance of the generic object Pump. Just to clarify, what would be the intention of such a model, we should say that this model would not store any of real values measured on the device. It is considered that there is another subsystem (let's call it for example a Measurement manager), which would do that instead. The model presented here will be just the model, from which human or machine will know, which one is the device that can measure observable values. The Measurement manager can use the model for example for a decision, which particular device to check, if we need the particular measure.
Thus the triple statements describing the situation can be modified to the following:

Generic objects (classes) and properties:

```
Pump is a Device
ActualWaterFlow is an ObservedProperty
ActualEnergyConsumption is an ObservedProperty
Device hasEvent Event
Event observesProperty ObservedProperty
ObservedProperty hasUnit Unit
```

Real objects (instances, individuals):

```
WaterPump1 isInstanceOf Pump
WaterPump1 hasEvent EnergyConsumptionEvent1
EnergyConsumptionEvent1 isInstanceOf Event
EnergyConsumptionEvent1 observesProperty ActualEnergyConsumption1
ActualEnergyConsumption1 instanceOf ActualEnergyConsumption
ActualEnergyConsumption1 hasUnit WattPerHour1
WattPerHour1 instanceOf Unit
WaterPump1 hasEvent WaterFlowEvent1
WaterFlowEvent1 instanceOf Event
WaterFlowEvent1 observesProperty ActualWaterFlow1
ActualWaterFlow1 instanceOf ActualWaterFlow
ActualWaterFlow1 hasUnit LiterPerHour1
LiterPerHour1 instanceOf Unit
```

What is above is still not a knowledge model in any formalism, although it is close to that. Before these statements can be transformed into formalism for knowledge representation, IRIs have to be created. Resource identificators are needed, to differentiate the model from other models of similar devices developed elsewhere in the world.

4.1.1 Resorce identifiers

Several objects are named in the proposed example already. There are for example the Pump, the Device and the Event. These are describing classes. IRIs will be used to create unique names of objects. In our case that will be done by using the IRI:

```
http://ebbits.eu/Device.owl#
```

This first part of the IRI is called a namespace identifier. This has to be used in front of the each new object name we create, to get fully qualified IRI of such an object. Sometimes it is expected, that the model will be too complex even inside one application, like it is in our example, so the user can create different prefixes for particular groups of objects. We will use the

```
http://ebbits.eu/Event.owl#
```

for the event related objects and the

```
http://ebbits.eu/Unit.owl#
```

for the unit related objects.
The usage of IRIs is usually further simplified by using prefixes. Thus we can for example define, that the

```
event:
```

prefix will be used instead of

```
http://ebbits.eu/Event.owl#
```

Usage of IIRs and corresponding prefixes can be seen in the following examples and in the Annex of this deliverable.

4.1.2 Textual representations comparison

Let’s see now, how the model is represented in various formalisms using the proposed names and IRIs. For comparison, we will show only a small fragment describing the Device in different formalisms. We will define, that a Pump is a subclass of a Device, the WaterPump1 is an instance of the Pump and the observesProperty is a property of Event with ObservedProperty as its range. In RDF/XML the definition looks like:

```
        <owl:Class rdf:ID="Pump">
          <rdfs:subClassOf rdf:resource="#Device"/>
        </owl:Class>
        <owl:ObjectProperty rdf:about="Event.owl#observesProperty">
          <rdfs:domain rdf:resource="Event.owl#Event"/>
          <rdfs:range rdf:resource="Event.owl#ObservedProperty"/>
        </owl:ObjectProperty>
          <owl:NamedIndividual
rdf:about="http://ebbits.eu/Device.owl#WaterPump1">
              <rdf:type rdf:resource="http://ebbits.eu/Device.owl#Pump"/>
              <hasEvent
rdf:resource="http://ebbits.eu/Event.owl#EnergyConsumptionEvent1"/>
              <hasEvent
rdf:resource="http://ebbits.eu/Event.owl#WaterFlowEvent1"/>
          </owl:NamedIndividual>
```

In OWL/XML we should write:

```
        <Declaration>
            <Class IRI="#Pump"/>
        </Declaration>
        <SubClassOf>
            <Class IRI="#Pump"/>
            <Class IRI="#Device"/>
        </SubClassOf>
        <ObjectPropertyDomain>
            <ObjectProperty IRI="#hasEvent"/>
            <Class IRI="#Device"/>
        </ObjectPropertyDomain>
        <ClassAssertion>
            <Class IRI="#Pump"/>
            <NamedIndividual IRI="#WaterPump1"/>
        </ClassAssertion>
        <ObjectPropertyAssertion>
            <ObjectProperty IRI="#hasEvent"/>
            <NamedIndividual IRI="#WaterPump1"/>
            <NamedIndividual abbreviatedIRI="event:WaterFlowEvent1"/>
        </ObjectPropertyAssertion>
        <ObjectPropertyAssertion>
            <ObjectProperty IRI="#hasEvent"/>
            <NamedIndividual IRI="#WaterPump1"/>
            <NamedIndividual abbreviatedIRI="event:EnergyConsumptionEvent1"/>
        </ObjectPropertyAssertion>
```

In the Functional OWL the definition is much shorter:

```
      Declaration(Class(:Pump))
      SubClassOf(:Pump :Device)
      Declaration(ObjectProperty(:hasEvent))
      ObjectPropertyDomain(:hasEvent :Device)
      ObjectPropertyRange(:hasEvent event:Event)
      ClassAssertion(:Pump :WaterPump1)
      ObjectPropertyAssertion(:hasEvent :WaterPump1
event:EnergyConsumptionEvent1)
      ObjectPropertyAssertion(:hasEvent :WaterPump1 event:WaterFlowEvent1)
```

In the Manchester OWL Syntax the definition is also quite short:

```
      Class: Pump
          SubClassOf:
              Device
      ObjectProperty: hasEvent
          Domain:
              Device
          Range:
              event:Event
      Individual: WaterPump1
          Types:
              Pump
          Facts:
           hasEvent  event:WaterFlowEvent1,
           hasEvent  event:EnergyConsumptionEvent1
```

Turtle notation of the same element will look like:

```
      :Pump a owl:Class ;
           rdfs:subClassOf :Device .
      :hasEvent rdf:type owl:ObjectProperty ;
           rdfs:domain :Device ;
           rdfs:range <http://ebbits.eu/Event.owl#Event> .
      :WaterPump1 rdf:type :Pump , owl:NamedIndividual ;
           hasEvent <http://ebbits.eu/Event.owl#EnergyConsumptionEvent1> ,
                            <http://ebbits.eu/Event.owl#WaterFlowEvent1> .
```

The N-Triple notation to express the same is:

```
      <http://ebbits.eu/Device.owl#Pump>
         <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
         <http://www.w3.org/2002/07/owl#Class> .
```

```
<http://ebbits.eu/Device.owl#Pump>
   <http://www.w3.org/2000/01/rdf-schema#subClassOf>
   <http://ebbits.eu/Device.owl#Device> .
<http://ebbits.eu/Device.owl#hasEvent>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://www.w3.org/2002/07/owl#ObjectProperty> .
<http://ebbits.eu/Device.owl#hasEvent>
   <http://www.w3.org/2000/01/rdf-schema#range>
   <http://ebbits.eu/Event.owl#Event> .
<http://ebbits.eu/Device.owl#hasEvent>
   <http://www.w3.org/2000/01/rdf-schema#domain>
   <http://ebbits.eu/Device.owl#Device> .
<http://ebbits.eu/Device.owl#WaterPump1>
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <http://ebbits.eu/Device.owl#Pump> .
<http://ebbits.eu/Device.owl#WaterPump1>
   <http://ebbits.eu/Device.owl#hasEvent>
   <http://ebbits.eu/Event.owl#WaterFlowEvent1> .
<http://ebbits.eu/Device.owl#WaterPump1>
   <http://ebbits.eu/Device.owl#hasEvent>
   <http://ebbits.eu/Event.owl#EnergyConsumptionEvent1> .
```

OBO 1.2 OWL can describe the same in the following:

```
[Term]
id: Pump
name: Pump
is_a: Device
[Typedef]
id: hasEvent
name: hasEvent
domain: Device
range: Event
is_metadata_tag: false
[Instance]
id: WaterPump1
name: WaterPump1
instance_of: Pump
property_value: hasEvent WaterFlowEvent1
property_value: hasEvent EnergyConsumptionEvent1
```

As we can see, the closer the syntax of ontologies to the XML, less human user friendly it is. It does not necessarily mean that the XML is a not human friendly formalism (it also was developed to be a human-also readable formalism, but not so much human understandable). It only is used for ontology representation in a quite complicated way. Nevertheless, what we can say is that we can use any of these for describing a simple knowledge model. For simple ontologies, or for an early development stage of more complex models, notations optimised for human are best suitable. Several syntaxes are subsets of each other and can be used for expressing only particular subset of possible relations. More about expressivity of these has been written in previous chapters. If knowledge designer encounters an expressivity limitation of some formalism, it is possible to upgrade it to a higher-level language quite easily.

When it comes to a growing complexity and amount of represented knowledge, even the user-friendly formalisms become hard to read and understand by a human. To have a clearer idea, what we mean by complexity, let's see what our ontology, with all the relations from the beginning of this chapter, will look like in one of the user-friendly notations – in the Manchester OWL Syntax. In the ANNEX 1 of this deliverable we can see two-page listing of our very simple ontology. So everyone can imagine how long such a listing can be for only a few tens of different devices. Ontology editor tools with graphical ontology visualisation can help human in a complex knowledge model manipulation.

4.1.3 Visualisation of ontology

In Figure 3 and Figure 4 we can see some popular ontology editors and their approach to the visualisation of the example ontology.

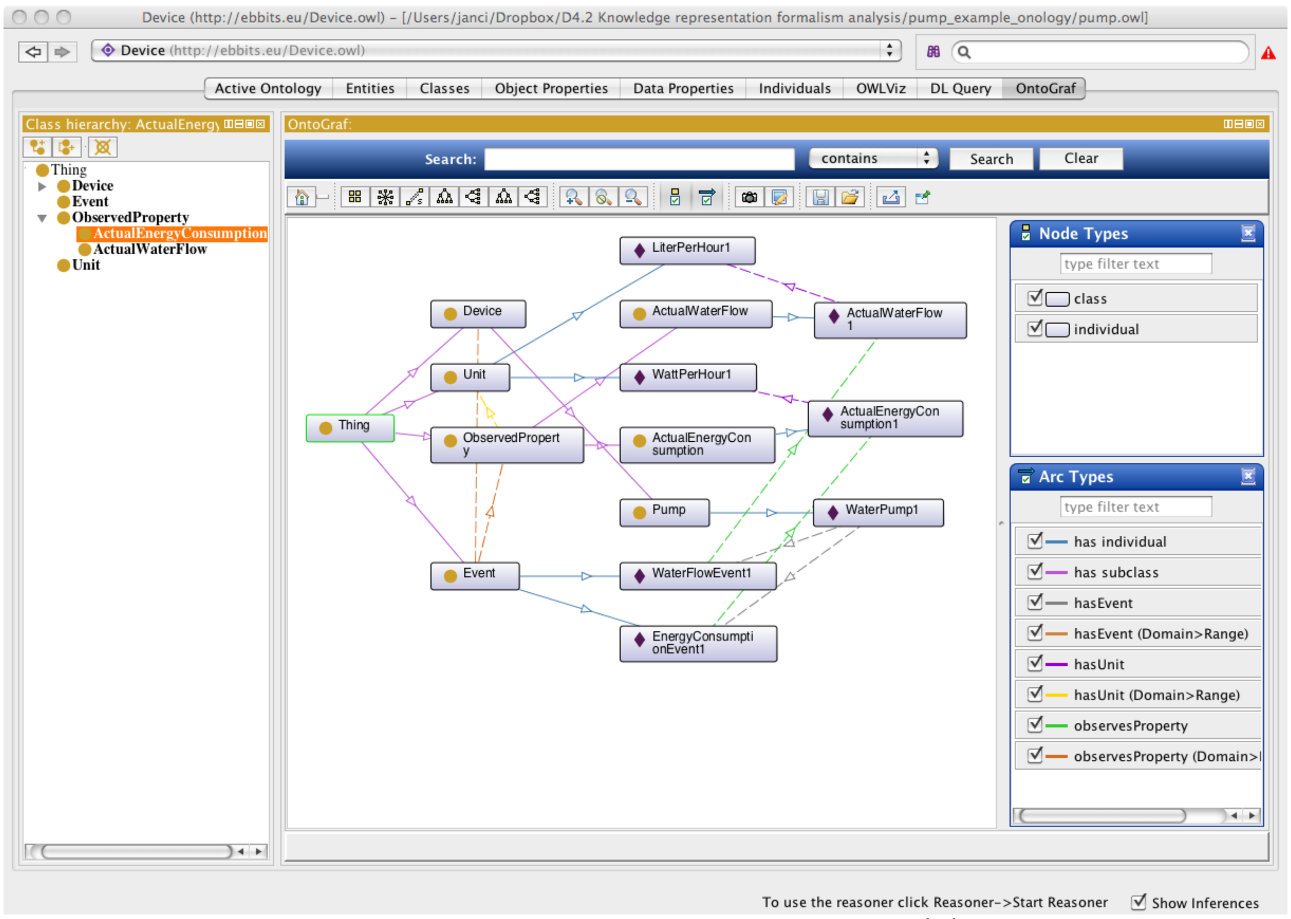


Figure 3 Device Ontology example visualisation in Protégé

In both cases the visualisation is a graph-like representation, with nodes representing classes and edges or lines between them representing properties. This helps user to overview and browse the ontology. Another commonly used way of browsing the ontology quickly can be seen on both images on the left side of the interface. It is a tree representation of so called "is-a relation", where sub nodes represent more specific versions of parent nodes. The "is-a relation" has a special purpose in ontologies, as it is used for inheritance between objects. It is a very common error to mix the "is-a" relation with a "part of" relation. The easiest way to overcome such a mistake is to say, "leaf-node is a parent-node" for any new nodes when constructing a model. In our ontology, for example, we can say, "Pump is a Device". A pump valve for example cannot be a sub-node of the Pump node in the is-a tree, as it is not true, that "Pump valve is a Pump". However, we can of course have the "part-of" relation among other properties in the ontology.

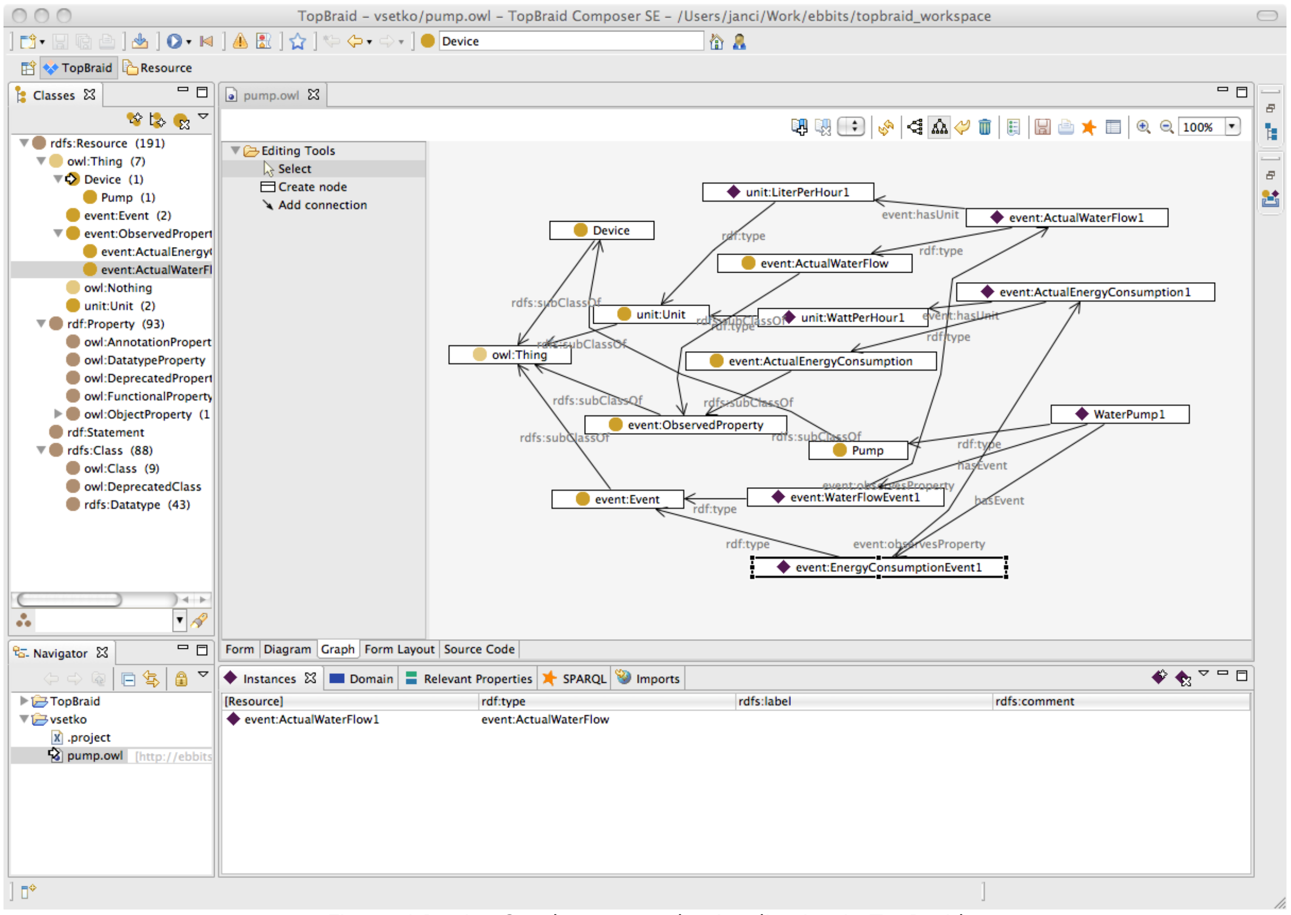


Figure 4 Device Ontology example visualisation in TopBraid

When editing ontologies, user uses another common ontology tool. It is a more or less standardized form structure to edit object properties. To modify the ObservedProperty concept in the example ontology, user uses very similar forms in different ontology tools (Figure 5).

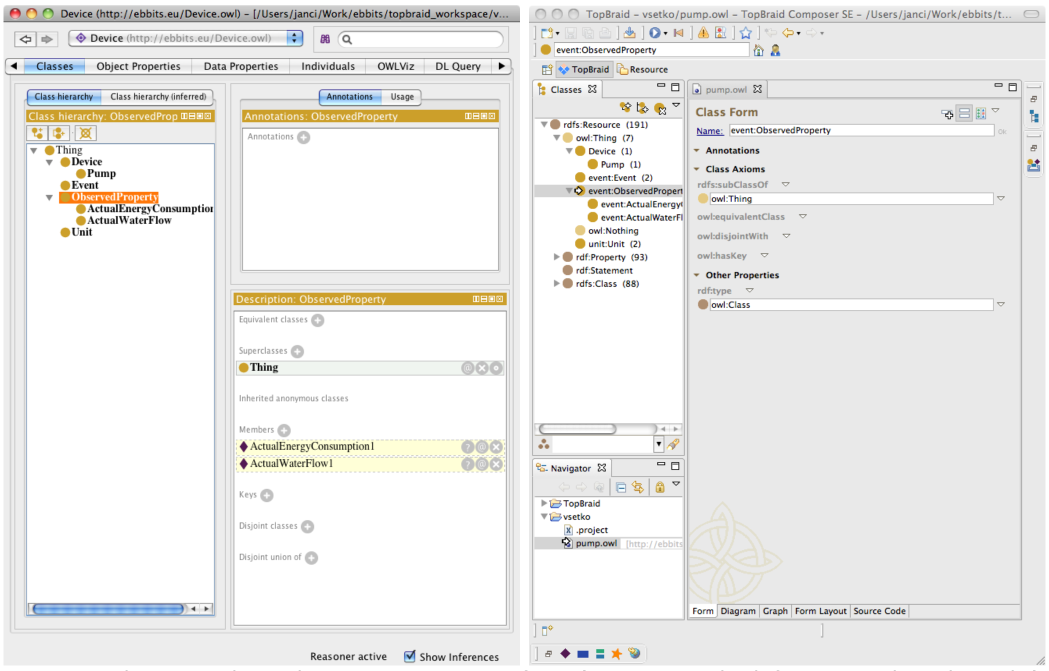


Figure 5 Editing ontology object properties in a form (Protege on the left, TopBraid on the right)

4.1.4 Query formalisms

Now, when we have the example ontology prepared, where the device description is stored, knowledge can be retrieved from it. Let's assume, there is a large set of devices already in the ontology and we need to address a particular device in some application. Here are some example queries in SPARQL language to do that:

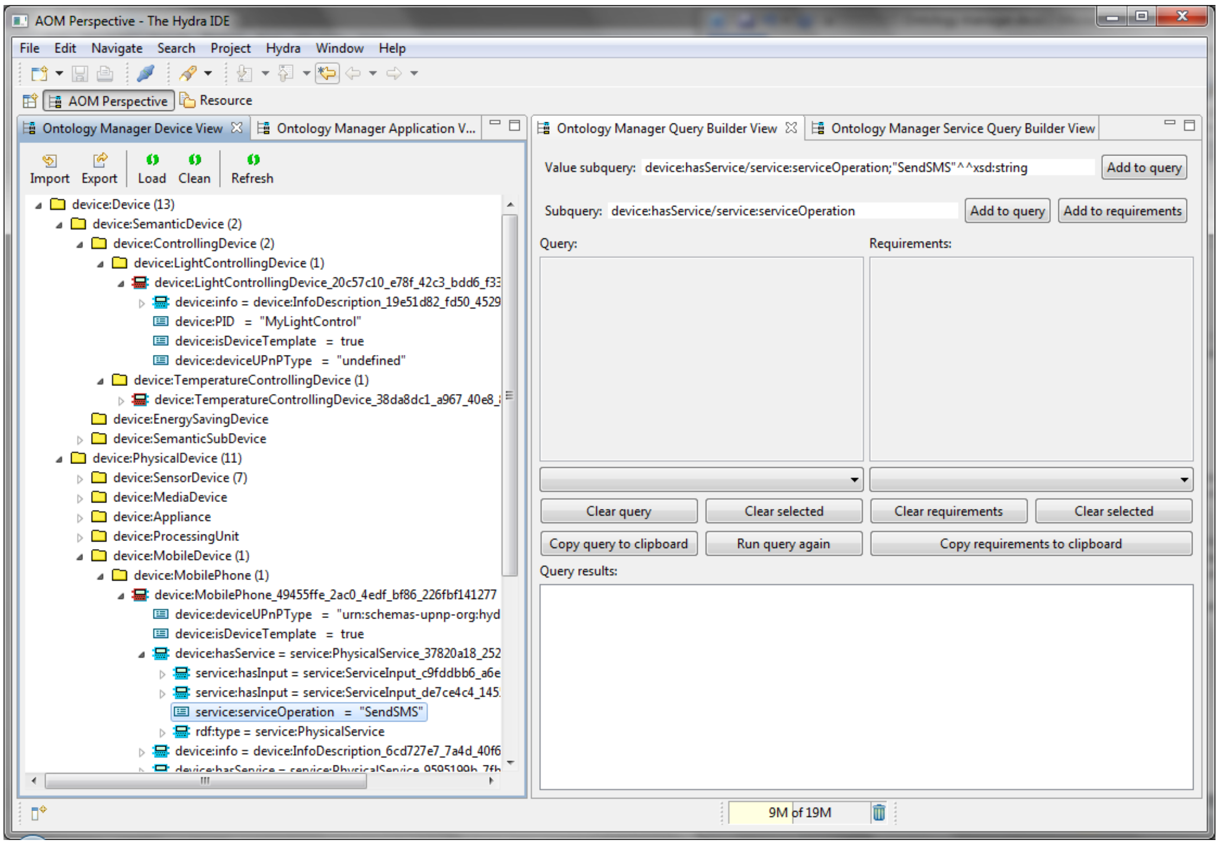
List all devices, on which an event can be triggered to observe the ActualWaterConsumption property:

```
select ?device where {
?device device:hasEvent ?event.
?event event:observesProperty event:ActualWaterConsumption.
}
```

List al events of Pumps measuring Property in l/h

```
select ?event where {
?device device:hasEvent ?event.
?device rdf:type device:Pump.
?event event:observesProperty ?result.
?result event:hasUnit unit:LiterPerHour1.
}
```

We can see, that these queries are very similar to the original RDF representation. The RDF is close to human understanding as it is based on Semantic networks, which can be used for natural language processing. Thus queries (questions) in the SPARQL are easy to formulate for a human user. However, even with SPARQL, queries (and especially the process of its construction) can become a complex task, if ontology is huge. For that reason, graphical tools for query formulation comes handy. These use the same approach as editors for ontologies. The user can easily browse ontologies to find corresponding objects, combine them into a query and then try to execute constructed queries to continuously check results on the go. An example of simplified QUERY editor from the Hydra project is in Figure 6.


AOM Perspective - The Hydra IDE
File Edit Navigate Search Project Hydra Window Help
AOM Perspective
Resource
Ontology Manager Device View
Ontology Manager Application V...
Import Export Load Clean Refresh
Ontology Manager Query Builder View
Ontology Manager Service Query Builder View
Add to query
Add to requirements
Clear query
Clear selected
Clear requirements
Clear selected
Copy query to clipboard
Run query again
Copy requirements to clipboard
9M of 19M

```
device:Device (13)
  device:SemanticDevice (2)
    device:ControllingDevice (2)
      device:LightControllingDevice (1)
        device:LightControllingDevice_20c57c10_e78f_42c3_bdd6_f33
          device:info = device:InfoDescription_19e51d82_fd50_4529
          device:PID  =  "MyLightControl"
          device:isDeviceTemplate  =  true
          device:deviceUPnPType  =  "undefined"
      device:TemperatureControllingDevice (1)
        device:TemperatureControllingDevice_38da8dc1_a967_40e8_
    device:EnergySavingDevice
    device:SemanticSubDevice
  device:PhysicalDevice (11)
    device:SensorDevice (7)
    device:MediaDevice
    device:Appliance
    device:ProcessingUnit
    device:MobileDevice (1)
      device:MobilePhone (1)
        device:MobilePhone_49455ffe_2ac0_4edf_bf86_226fbf141277
          device:deviceUPnPType  =  "urn:schemas-upnp-org:hyd
          device:isDeviceTemplate  =  true
          device:hasService = service:PhysicalService_37820a18_252
            service:hasInput = service:ServiceInput_c9fddbb6_a6e
            service:hasInput = service:ServiceInput_de7ce4c4_145
            service:serviceOperation  =  "SendSMS"
            rdf:type = service:PhysicalService
          device:info = device:InfoDescription_6cd727e7_7a4d_40f6
```

Value subquery: device:hasService/service:serviceOperation;"SendSMS"^^xsd:string

Subquery: device:hasService/service:serviceOperation

Query:

Requirements:

Query results:

Figure 6 Query editor for the LinkSmart device ontology

If we accept the fact that the SPARQL is equally powerful as the SQL for solving problems, we can say that with semantic technology we have the tool by means of which it is easier to formulate queries, but still of the same power as any of the database systems. Here comes the question of scalability of the triple stores in comparison to very advanced commercial database systems. We tried to answer this question in the D4.1 report. We have shown there that there are several solutions available for a scalable knowledge stores that can be used via predefined interfaces; however all of these are still in the development phase. We should build our system in such a way that it will be modular enough to be able to switch to a different triple store with a minimal effort if needed.

4.1.5 Semantic web services formalisms

To demonstrate differences among the above-mentioned formalisms for semantic descriptions of web services (cf. section 3.9), a sample web service will be annotated by the concepts taken from the Device.owl ontology, as it was designed in the previous subsections. Assume that there exists a web service that provides an analysis of key performance indexes. For the sake of simplicity, let us assume that the service is atomic one and has three input parameters – observed property, due date and unit.

The SAWSDL representation of the service is presented in the following listing. The original WSDL description is enhanced by the sawsdl:modelReference elements (marked in bold font), which provide a reference to respective ontology concepts for specified input parameters of the web service:

```
<wsdl:description
  targetNamespace="http://ebbits.eu/wsdl/AnalysisRequestService/"
  xmlns="http://ebbits.eu/wsdl/AnalysisRequestService/"
  xmlns:wsdl="http://www.w3.org/ns/wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sawsdl="http://www.w3.org/ns/sawsdl">

  <wsdl:types>
    <xsd:schema targetNamespace="http://ebbits.eu/wsdl/AnalysisRequestService">
    <xsd:element name="AnalysisRequestServiceRequest">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="observedProperty" type="xsd:string"
sawsdl:modelReference="http://ebbits.eu/onto/DeviceOntology#ObservedProperty"/>
          <xsd:element name="date" type="xsd:dateTime"
sawsdl:modelReference="http://ebbits.eu/onto/DeviceOntology#DueDate"/>
          <xsd:element name="unit" type="xsd:string"
sawsdl:modelReference="http://ebbits.eu/onto/DeviceOntology#Unit"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="AnalysisRequestServiceResponse" type="analysisResult"/>
    <xsd:simpleType name="analysisResult"
sawsdl:modelReference="http://ebbits.eu/onto/DeviceOntology#AnalysisResult">
      <xsd:restriction base="xsd:string"/>
    </xsd:simpleType>
    </xsd:schema>
  </wsdl:types>

  <wsdl:interface name="AnalysisRequestService">
      <wsdl:operation name="AnalysisRequestOperation"
pattern="http://www.w3.org/ns/wsdl/in-out">
```

```
      <wsdl:input element="AnalysisRequestServiceRequest"/>
      <wsdl:output element="AnalysisRequestServiceResponse"/>
    </wsdl:operation>
  </wsdl:interface>
</wsdl:description>
```

The OWL-S representation is provided in a form of OWL ontology, which models the service instance, service profile, process model with data flow, grounding instances, and WSDL definitions for grounding. The process model example for the web service of requesting the analysis of key performance indexes could be as follows:

```
<process:AtomicProcess rdf:ID="AnalysisRequestService">
  <process:hasInput>
    <process:Input rdf:ID="ObservedProperty">
      <process:parameterType rdf:resource="&xsd;#string"/>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
   <process:Input rdf:ID="DueDate">
      <process:parameterType rdf:resource="&xsd;#dateTime"/>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
   <process:Input rdf:ID="Unit">
      <process:parameterType rdf:resource="&xsd;#string"/>
    </process:Input>
  </process:hasInput>
  <process:hasPrecondition rdf:resource="#OPisActEnConsumption"/>
  <process:hasPrecondition rdf:resource="#UnitisWPH"/>
```

The WSMO framework provides native WSML format for semantic description of web services, which enables introducing conditional statements, variables, and other specific elements into preconditions, post-conditions, effects, and other parts of the service representation. The sample web service, presented below in WSML, contains precondition constraints on input parameters in the capability specification:

```
namespace {_"http://ebbits.eu/wsdl/AnalysisRequestService#",
   do       _"http://ebbits.eu/onto/DeviceOntology#",
   dc       _"http://purl.org/dc/elements/1.1#"}
webService _"http://ebbits.eu/wsdl/AnalysisRequestService"
importsOntology _"http://ebbits.eu/onto/DeviceOntology"
capability AnalysisRequestCapability
  sharedVariables {?observedProperty, ?unit}
  precondition
      nonFunctionalProperties
           dc#description hasValue "A textual description of the web service for
the analysis request."
      endNonFunctionalProperties
      definedBy
          ?observedProperty memberOf do#ActualEnergyConsumption
          and
          ?unit hasValue do#WattPerHour1
interface
   choreography AnalysisRequestChoreography
   orchestration AnalysisRequestOrchestration
```

The formats for semantic annotation of web services differ in the level of expressiveness and means of combining processes into complex workflow structures. The WSMO framework was designed in years 2004-05 specifically for modelling and maintaining semantic web services (deBruijn et al. 2005). It is, however, still in a pre-mature state, currently available only in version 0.8. The SAWSDL and OWL-S technologies are more stable and proven by numerous applications in practice. Both of them are constructed as semantic extensions on an existing WSDL; however, the support for workflow structures and complex service orchestration/choreography constructs is less advanced as it is in WSMO. Selection of a proper formalism and framework for semantic annotation of web services, which will be most suitable for ebbits purposes, should be driven by its compatibility with the LinkSmart environment, which will be investigated later in the WP4 in more details.

5. Conclusion

Based on the knowledge representations state of the art analysis and the ebbits use cases analysis, we can postulate that the following recommendations will be used in the process of development of knowledge models in the ebbits project:

1. RDF/OWL knowledge representation is the choice of the ebbits project, as it is the most widely accepted format for a knowledge sharing.
2. As it is possible to migrate from a simple to more complex formalism we will use as simple formalisms as possible in the process of the knowledge modelling. We will start with user-friendly notations of the RDF and move to the OWL if needed.
3. Ontologies will use predefined IRI namespaces. These will be based on the project prefix:
   `http://www.ebbits-project.eu/ontologies`
   or on user partners company prefixes, if requested.
4. Ontologies will be shared among partners via SVN in XML based RDF/OWL syntaxes
5. If ontologies are shared among users, visual explanations will be used as well using ontology graphical visualisation tools of a common choice.
6. Semantic web service formalisms are incompatible between each other and cannot be easily transferred from one to another. Web service formalisms have to be further investigated in WP4 and the most suitable formalism needs to be selected.
7. The need for development of simplified ontology and query manipulation tools will be considered within the project. The existing open source tools will be used, if possible.

6. References

(Al-Masri and Mahmoud 2008) Al-Masri, E., Mahmoud, Q. H.: Investigating Web Services on the World Wide Web. In: Proceedings of the 16th international conference on World Wide Web, ACM New York, 2008, pages 795-804.

(Alonso et al. 2008) Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures and Applications. Springer, 2004, 354 p.

(Baader et al. 2003) Baader, F., Calvanese, D., McGuinness D., Nardi D., and Patel-Schneider P.F., editors (2003). The Description Logic Handbook: Theory, Implementation and Applications, 2003.

(Bechhofer et al. 2004) Bechhofer, S., Harmelen, S. V., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., and Stein, L.A.(2004). OWL Web Ontology Language Reference. W3C Recommendation. Available at http://www.w3.org/TR/owl-ref/. World Wide Web Consortium, Feb. 2004

(Berners-Lee, Hendler and Lassila 2001) Berners-Lee, T., Hendler, J., and Lassila, O.(2001). "The Semantic Web." In: Scientific American 284 (May 2001), pages 29–37.

(Berners-Lee et al. 2005) Berners-Lee, et al., Standards Track, RFC 3986, URI Generic Syntax, January 2005, Available at http://tools.ietf.org/html/rfc3986.

(Bikakis et al. 2009) Bikakis N., Tsinaraki C., Gioldasis N., Christodoulakis S.: "Querying XML Data with SPARQL". 20th International Conference on Database and Expert Systems Applications (DEXA'09).

(Brachman 1983) Brachman, R. J.(1983). "What IS-A Is and Isn't: An Analysis of Taxonomic Linksin Semantic Networks." In: IEEE Computer 16.10 (1983), pages 30–36. issn: 0018-9162.

(Brachman and Levesque 1985) Brachman, R.J. and Levesque, H.J., editors. Readings in Knowledge Representation. Kaufmann, M., Los Altos, 1985. isbn: 978-0934613019.

(Brickley and Guha 2004) Brickley, D., Guha, R.V. (Editors), RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation, 10 February 2004. http://www.w3.org/TR/rdf-schema/.

(Cerami 2002) Cerami, E.: Web Services Essentials. O'Reilly Media, Inc., 2002, 304 p.

(Chebotko, Lu and Fotouhi 2009) Chebotko, A., Lu, S., Fotouhi, F. : Semantics preserving SPARQL-to-SQL translation. Data Knowl. Eng. 68, 10 (October 2009), 973-1000.

(Chinnici et al. 2007) Chinnici, R., Moreau, J-J., Ryman, A., Weerawarana, S.: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Recommendation. Available at http://www.w3.org/TR/wsdl20/, World Wide Web Consortium, June 2007.

(Clement et al. 2004) Clement, L., Hately, A., von Riegen, C., Rogers, R. (eds): UDDI Version 3.0.2. Available at http://uddi.org/pubs/uddi-v3.0.2-20041019.htm. OASIS, Oct. 2004.

(Curbera et al. 2002) Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. In: IEEE Internet Computing, Vol. 6, Issue 2, 2002, pages 86-93.

(Dau et al. 2009) Dau, F., Hladik, J., Becker, A., Brockmans, S., Korf, R., Erdmann, M. and Niemann, M. "D.G4.1 Modellierungsmethodik und globales semantisches Modell", Technical report, SAP AG, ontoprise, FUB, 2009.

(Dean and Schreiber 2004) Dean, M., and Schreiber, G.(2004), Editors, OWL Web Ontology Language Reference, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/owl-ref/.

(deBruijn et al. 2005) de Bruijn, J. et al: Web Service Modeling Ontology (WSMO). W3C Member Submission. Available at http://www.w3.org/Submission/WSMO/, World Wide Web Consortium, June 2005.

(deBruijn et al. 2008) de Bruijn, J. et al: The Web Service Modeling Language WSML. Available at http://www.wsmo.org/wsml/wsml-syntax, ESSI WSML working group, 2008.

(DLP - Grosof et al. 2003) Grosof, B. N., Horrocks I., Volz R., and Decker S. (2003). Description Logic Programs: Combining Logic Programs with Description Logic. in Proc. of the 12th Int. World Wide Web Conference (WWW 2003), Budapest, Hungary, 2003. pp.: 48–57:

(Elliot et al. 2009) Elliott, B., Cheng, E., Thomas-Ogbuji, Ch., and Ozsoyoglu, Z., M.: 2009. A complete translation from SPARQL into efficient SQL. In Proceedings of the 2009 International Database Engineering. Applications Symposium (IDEAS '09). ACM, New York, NY, USA, 31-42.

(Farrell and Lausen 2007) Farrell, J., Lausen, H.: Semantic Annotations for WSDL and XML Schema. W3C Recommendation. Available at http://www.w3.org/TR/sawsdl/, World Wide Web Consortium, Aug. 2007.

(Fielding 2000) Fielding, R. T.: Representational state transfer (REST). Chapter 5 in Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.

(Gudgin et al. 2007) Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen, H., Karmarkar, A., Lafon, Y.: Simple Object Access Protocol (SOAP) 1.2. Part 1: Messaging Framework (Second Edition). W3C Recommendation. Available at http://www.w3.org/TR/soap12-part1/, World Wide Web Consortium, April 2007.

(Hayes 1979) Hayes P. J.(1979). "The Logic of Frames." In: Frame Conceptions and Text Understanding. Edited by D. Metzing. Republished in (Brachman and Levesque 1985). Walter de Gruyter and Co., 1979, pages 46–61.

(Herman and Horst 2005) Herman J. and Horst. J. (2005). Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary, of Web Semantics 3(2–3):79–115, 2005.

(Hitzler et al. 2009) Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (editors): OWL 2 Web Ontology Language: Primer. W3C Recommendation, October 27 2009. Available at http://www.w3.org/TR/owl2-primer/.

(Horrocks 2002) Horrocks, I.(2002). "DAML+OIL: a Reason-able Web Ontology Language." In: Proceedings of 8th Conference on Extending Database Technology. Volume 2287. Lecture Notes in Computer Science. Springer-Verlag, 2002, pages 2–13.

(Jordan and Evdemon 2007) Jordan, D., Evdemon, J. (TC chairs): Web Services Business Process Execution Language. Version 2.0. Available at http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, OASIS Standard, April 2007.

(Katifori et al. 2006) Katifori, A., Torou, E., Halatsis, C., Lepouras, G. and Vassilakis, C.: 2006. A Comparative Study of Four Ontology Visualization Techniques in Protege: Experiment Setup and Preliminary Results. In Proceedings of the conference on Information Visualization (IV '06). IEEE Computer Society, Washington, DC, USA, 417-423.

(Katifori et al. 2007) Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C. and Giannopoulou, E.: 2007. Ontology visualization methods\—a survey. ACM Comput. Surv. 39, 4, Article 10 (November 2007).

(Kavantzas et al. 2005) Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., Barreto, C. (eds.): Web Services Choreography Description Language Version 1.0. W3C Candidate Recommendation. Available at http://www.w3.org/TR/ws-cdl-10/, World Wide Web Consortium, Nov. 2005.

(Kifer, Lausen and Wu 1995) Kifer, M., Lausen, G., Wu, J., Logical Foundations of Object-Oriented and Frame-Based Languages, *Journal of the ACM,* 1995, 42, 741-843.

(Kostelnik, Sarnovsky and Hreno 2009) Kostelník, P., Sarnovský, M., Hreňo, J.:Ontologies in HYDRA – Middleware for Ambient Intelligent Devices, Ambient Intelligence and Smart Environments Volume 5, 2009, Ambient Intelligence Perspectives II - Selected Papers from the Second International Ambient Intelligence Forum 2009, Edited by Pavel Čech, Vladimír Bureš, Ludmila Nerudová.

(Krötzsch, Rudolph and Hitzler 2008) Krötzsch, M., Rudolph, S., Hitzler, P. ELP: Tractable Rules for OWL 2. In Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K.(2008). eds.: Proceedings of the 7th International Semantic Web Conference (ISWC-08), pp. 649–664. Springer 2008.

(Manola and Miller 2004) Manola, F. and Miller, E., Editors, RDF Primer, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/rdf-primer/#applications.

(Martin et al. 2004) Martin, D. et al: OWL-S: Semantic Markup for Web Services. W3C Member Submission. Available at http://www.w3.org/Submission/OWL-S/, World Wide Web Consortium, Nov. 2004.

(Martin et al. 2008) Martin, D. et al: OWL-S 1.2 Release. Available at http://www.ai.sri.com/daml/services/owl-s/1.2/, OWL-S (formerly DAML-S) Coalition, Dec. 2008.

(Minski 1981) Minsky, M.(1975). "A Framework for Representing Knowledge." In: Mind Design. Edited by Haugeland J.(1975). A longer version appeared in The Psychology of Computer Vision (1975). Republished in (Brachman and Levesque 1985). The MIT Press, 1981.

(Motik, Patel-Schneider and Parsia 2009) Motik, B., Patel-Schneider, P.F., and Parsia, B.(2009). OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation. Available at http://www.w3.org/TR/owl2-syntax/, World Wide Web Consortium, Oct. 2009.

(Motik et al. 2009) Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. eds. OWL 2 Web Ontology Language: Profiles, W3C Recommendation, 27 October 2009, http://www.w3.org/TR/owl2-profiles/.

(OWL 2009) W3C OWL Working Group, Editors, OWL 2 Web Ontology Language, Document Overview , W3C Recommendation, 11 June 2009, http://www.w3.org/TR/owl2-overview/.

(Papazoglou 2003) Papazoglou, M.: Service-oriented computing: concepts, characteristics and directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003, pages 3-12.

(Perez, Arenas and Gutierrez 2009)Perez, J., Arenas, M. and Gutierrez, C.: 2009. Semantics and complexity of SPARQL. ACM Trans. Database Syst. 34, 3, Article 16 (September 2009), 45 pages.

(Quillian 1967) Quillian, M. R.(1967). "Word concepts: A theory and simulation of some basic capabilities." In: Behavioral Science 12 (1967). Republished in (Brachman and Levesque 1985), pages 410–430.

(Reynolds 2006) Reynolds, J.: Service Orchestration vs. Service Choreography. Available at http://weblogs.java.net/blog/johnreynolds/archive/2006/01/service_orchest.html, Blog posted at Java.net, Jan. 2006.

(Richens 1956) Richens, R.H.: Preprogramming for Mechanical Translation, Mechanical Translation, Volume 3, Number 1, July 1956; p.20-25.

(Schubert, Goebel and Cercone 1979) Schubert, L. K., Goebel, R. G., and Cercone, N. J.(1979). "The Structure and Organization of a Semantic Net for Comprehension and Inference." In: Associative Networks: Representation and Use of Knowledge by Computers. Edited by Findler N.V. Academic Press, 1979, pages 121–175.

(SPARQL 2008) SPARQL Query Language for RDF, W3C Recommendation 15 January 2008, Available at http://www.w3.org/TR/rdf-sparql-query/

(Suntisrivaraporn 2008)	Suntisrivaraporn, B.: 2008, “Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies.” Available at http://nbn-resolving.de/urn:nbn:de:bsz:14-ds-1233830966436-59282. PhD thesis. Technische Universität Dresden, 2008.
(XML 2008)	Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation 26 November 2008, Available at http://www.w3.org/TR/REC-xml/.
(XMLRCP 2003)	XML-RPC Home Page. Available at http://www.xmlrpc.com, Scripting News, Inc., 2003.
(WSA 2007)	Web Services Activity. W3C Technology and Society domain. Available at http://www.w3.org/2002/ws/, World Wide Web Consortium, 2007.
(WSMX 2008)	Web Service Modelling eXecution environment. Available at http://www.wsmx.org, DERI Galway and STI Innsbruck, 2008.

7. ANNEX

An ebbits example ontology in the Manchester OWL Syntax:

```
Prefix: xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix: unit: <http://ebbits.eu/Unit.owl#>
Prefix: owl: <http://www.w3.org/2002/07/owl#>
Prefix: event: <http://ebbits.eu/Event.owl#>
Prefix: : <http://ebbits.eu/Device.owl#>
Prefix: xml: <http://www.w3.org/XML/1998/namespace>
Prefix: rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
Prefix: rdfs: <http://www.w3.org/2000/01/rdf-schema#>
Ontology: <http://ebbits.eu/Device.owl>

ObjectProperty: hasEvent

    Domain:
        Device
    Range:
        event:Event

ObjectProperty: event:observesProperty

    Domain:
        event:Event
    Range:
        event:ObservedProperty

ObjectProperty: event:hasUnit

    Domain:
        event:ObservedProperty
    Range:
        unit:Unit

Class: event:ObservedProperty

    SubClassOf:
        owl:Thing

Class: owl:Thing


Class: unit:Unit

    SubClassOf:
        owl:Thing

Class: Device

    SubClassOf:
        owl:Thing

Class: Pump

    SubClassOf:
        Device

Class: event:ActualWaterFlow
```

```
    SubClassOf:
        event:ObservedProperty

Class: event:Event

    SubClassOf:
        owl:Thing

Class: event:ActualEnergyConsumption

    SubClassOf:
        event:ObservedProperty

Individual: event:ActualWaterFlow1

    Types:
        event:ActualWaterFlow
    Facts:
     event:hasUnit  unit:LiterPerHour1

Individual: WaterPump1

    Types:
        Pump
    Facts:
     hasEvent  event:WaterFlowEvent1,
     hasEvent  event:EnergyConsumptionEvent1

Individual: event:ActualEnergyConsumption1

    Types:
        event:ActualEnergyConsumption
    Facts:
     event:hasUnit  unit:WattPerHour1

Individual: unit:LiterPerHour1

    Types:
        unit:Unit

Individual: unit:WattPerHour1

    Types:
        unit:Unit

Individual: event:WaterFlowEvent1

    Types:
        event:Event
    Facts:
     event:observesProperty  event:ActualWaterFlow1

Individual: event:EnergyConsumptionEvent1

    Types:
        event:Event
    Facts:
     event:observesProperty  event:ActualEnergyConsumption1
```