FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Master Information Technology	Spring semester, 2008. Open / ~~Confidential~~
Author: Odd Frode Torbjørnsen	…… (signature author)
Instructor: Chunming Rong Supervisor(s): Henning Janssen, PCA SIG Drilling & Completion	
Title of Master's Thesis: SAWSDL in Oil & Gas Norwegian title: SAWSDL i Olje & Gass	
ECTS: 30	
Subject headings:	Pages: …… + attachments/other: …… Stavanger, …….. Date/year

Abstract

SAWSDL, The Semantic Annotations for WSDL and XML Schema, is a W3C recommendation that defines mechanisms to link semantics to the description of web services. These semantics when expressed in formal languages can help disambiguate the description of Web services during automatic discovery and composition of the Web services. This will help the industry to compose new web services based on the discovery of existing web services and based on the request made by a user even if it is expressed in an informal language.

The Oil & Gas industry has stated high ambitions in designing remotely controlled installations in the High North. This will require that a large set of real time data is acquired and furthermore that the challenging task of interpreting all these data must be dealt with. OLF visions that Semantic Web technology might be one solution to provide automatic reasoning of large sets of data.

As a first step towards the semantic web the PCA SIG for Drilling & Completions wanted to evaluate the SAWSDL specification to find out how to deploy this specification within the Oil & Gas IT infrastructure. This Master thesis is aiming to provide guidelines on how to implement SAWSDL based on a use case provided by the SIG.

Table of contents

Acknowledgements

Henning Janssen, Chairman of the PCA SIG for Drilling & Completion, for coming up with a very interesting subject for this Master Thesis. This area has proven so interesting that it will probably influence my future professional career.

Kari Anne Haaland Thorsen, research fellow at the Department of Electrical Engineering and Computer Science UiS, for revealing the secrets of OWL and providing useful feedback on the thesis.

Jan Einar Gravdal, researcher at IRIS, for providing me with important OPC documentation.

Anne Berit, my wife, for pushing me into doing this thesis.

Stakeholders

The following groups are stakeholders in this project:

POSC Caesar SIG (Special Interest Group)

PCA (POSC Caesar), the leading global, not-for-profit, standardization organization for the process industry including oil and gas, has through ISO TC 184/SC4, developed a methodology for data integration across disciplines and phases and this work has been documented in ISO 15926 "Integration of lifecycle data for process plants including oil and gas production facilities".

PCA has nominated a SIG – Special Interest Group) to create and maintain Ontology for the Oil & Gas Drilling & Completions domain. The PCA SIG is chaired by UiS (Kari Anne Haaland Thorsen), IRIS (Nejm Saadallah) and NOV (Henning Jansen)

The PCA SIG is orchestrating a session at the Semantic Days Conference in Stavanger April 2008 http://www.posccaesar.com/ The SIG aims at presenting a conclusion on the use of SAWSDL in conjunction with OPC UA, WITSML and ISO-15926

"AutoConRig", Petromaks - Research Council of Norway project:

National Oilwell Varco is in cooperation with StatoilHydro, Computas AS, IRIS and two more major operating companies starting a RCN project which aims at creating a next-generation quantum leap technology for offshore Drilling Control Systems. In conjunction with this we are investigating the SAWSDL proposal from W3C and recommends that a proper Master Thesis is used as a conclusion and recommendation to the RCN project. Responsible for AutoConRig is Henning Jansen, National Oilwell Varco Stavanger.

The Capgemini EpiCentre

The Capgemini EPiCentre provides the oil and gas industry with subject matter expertise, experience and lessons learned from business critical upstream and midstream projects.

Integrated Operations with a focus on information management and process improvements is one of the main business areas where the centre wants to maintain leadership and provide expertise and services to the upstream industry. In this respect the EpiCentre wants to join and contribute to this project.

Section 1: Introduction

Web services add a new level of functionality to the Web, an environment of distributed applications. The SOAP protocol and the Web service description language enables systems to communicate over the internet.

But the XML descriptions of the web services still demand human interpretations in order to set up the workflows needed to exchange data between the services offered. Given the nature of an autonomic distributed environment this means that the industry needs to face the challenges of resolving a larger degree of automation in searching for available and adequate web services, integrating them without the need of human interactions.

The World Wide Web Consortium (W3C) has made a standard for annotating semantics to the WSDL, SAWSDL. This standard adds hooks that let the WSDL components point to their semantics (Kopecky, Vitvar et al. 2007). This standard enables us to annotate WSDL with pointers to semantic concepts. Applications will be able to interpret the references enabling them to automate tasks as service discovery, composition of new services and invocation of these.

The web is an enormous data repository and web services are growing without borders. In order to understand all these data and services, the Semantic web defines the logic and knowledge representation to help computers find the right information for their users.

Computers must have access to structured sets of information and rules that they can use to do automated reasoning for the semantic web to function. Such rules have been developed by researchers within Artificial-Intelligence systems long time before the web was developed (Berners-Lee, Hendler et al. 2001).

Computers may discover and even combine services to create new services by using semantics and in that sense enable the web services to grow together with the data on the web.

SAWSDL is the W3C's first step toward standardizing the Semantic Web Services, SWS. SAWSDL extends WSDL with pointers to semantics defined in an ontology.

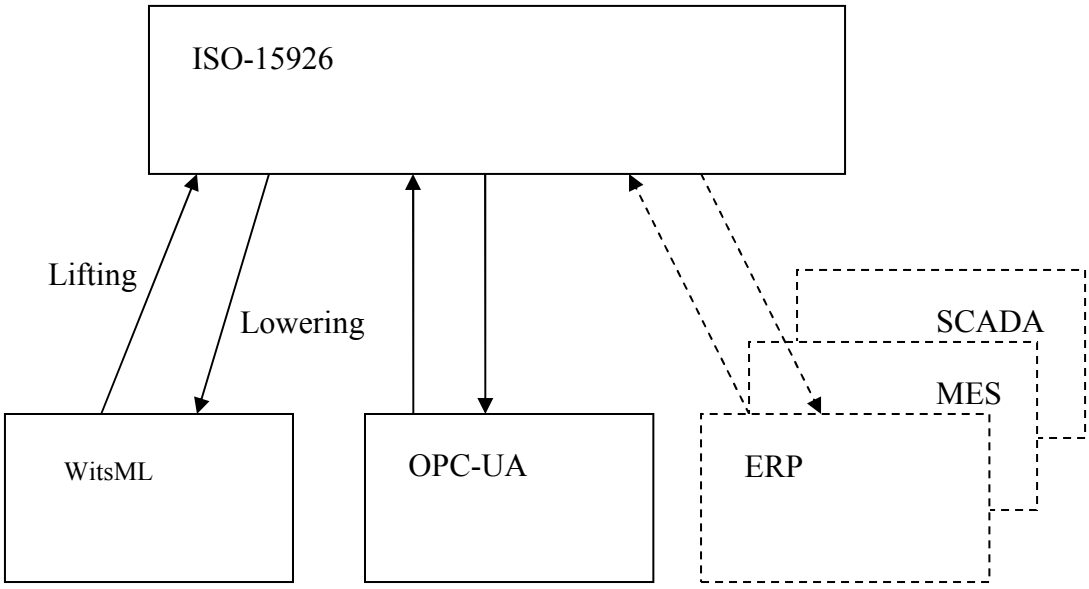
Ambition

As this technology is relatively new, it has not been widely accepted and adapted by the industry.

PCA SIG Drilling & Completion has expressed needs to integrate web services based on the widely accepted WSDL Protocol WitsML used by the Oil & Gas upstream industry and the OPC-UA used by the processing industry.

This could easily be conducted by using traditional point-to-point integration methodology. But by elaborating Semantic annotated web services, annotating different WSDL domains to a common ontology, we could (at least in theory) do integration to any given WSDL protocol that could be annotated to the same ontology.

This Master Degree thesis is aiming to prove that it is possible to integrate WitsML and OPC-UA by using the Ontology defined in ISO-15296. This ontology shall be annotated to the web services by following the recommendation from W3C, SAWSDL. If the thesis concludes by stating that this is possible, we could assume that it will be possible to integrate web services in the ERP, MES and SCADA domains as well.



Other industries that have done similar evaluations of SAWSDL are within life sciences like glycoproteomics. Glycoproteomics is a branch of proteomics that identifies, catalogs, and characterizes proteins containing carbohydrates as a post-translational modification. (Baker and Cheung 2005)

In addition a lot of tools like Radiant and WSMO is supporting this way of annotating web services. But to our knowledge there has not been done any similar evaluations within the Oil & Gas industry earlier.

How to read this document

Citations in this document are referenced in the following format *(<author> <year>)*. The referenced documents are listed in section References at the end of this document.

The following is a short description of the different sections of this document.

Section 2: Semantic Technologies, describes the basics of the semantic technologies and how it relates to SOA and the rest of the IT stack.

Section 3: Core Technologies describes the core technologies of the semantic layer also referred to as the knowledge plane.

Section 4: The Semantic Web is a short introduction to the semantic web envisioned by Tim Berners Lee and some of the W3C Semantic Web Standards.

Section 5: Selected Capabilities describes the capabilities and functionalities of the semantic web that is relevant to this thesis.

Section 6: Oil and Gas Applications of Semantic Technologies aims to envision how to apply these capabilities within the Oil & Gas industry.

Section 7: Tests defines the tests that were elaborated.

Section 8: Results, Findings and Possibilities present the results and findings from the test cases. In addition some possibilities envisioned when running these test cases are also presented.

Section 9: Summary contains a summary of the tests and findings.

Section 10: Conclusion and further work contains the conclusions made in this thesis. It also contains some suggestions for tasks that could be performed in order to fulfill a complete semantic web implementation of the described pilot.

Section 2: Semantic Technologies

Characterization of Semantic Technologies

This a list of some definitions of the semantic technologies found in the literature:

A semantic technology is a software technology that allows the meaning of and associations between information to be known and processed at execution time. For a semantic technology to be truly at work within a system there must be a knowledge model of some part of the world that is used by one or more applications at execution time.(Polikoff and Allemang 2003)

Semantic technologies include software standards and methodologies that are aimed at providing more explicit meaning for the information that's at our disposal. (McComb 2005)

Semantic technologies are functional capabilities that enable both people and computers to create, discover, represent, organize process, manage, reason with, present, share, and utilize meanings and knowledge to accomplish business, personal, and societal purposes. (Davis 2006)

One definition of semantic interoperability tells us how semantic technology may help us to overcome the ever increasing flow of information:

Semantic interoperability is a dynamic enterprise capability derived from the application of special software technologies (such as reasoners, inference engines, ontologies and models) that infer, relate, interpret, and classify the implicit meanings of digital content without human involvement – which in turn drive adaptive business processes, enterprise knowledge, business rules, and software application interoperability. (Pollock and Hodgson 2007)

Summarized the following characterization of semantic technology will be used throughout this document:

Semantic technologies are software technologies that exploit the meaning of the information at hand and involve the use of an explicit knowledge model. (Hansen, Gagnes et al. 2007)

Examples of semantic technology may be ontologies, rule engines, integrators and smart agents.

Functions of Semantic Technologies

The semantic technologies are an extension to the existing information technologies. XML, databases and other technologies where the semantics are not represented explicitly are not considered to be semantic.

In semantic technology the meaning (semantics) of the data is modelled and declared separately in addition to the data itself and the program logic. These declarations are understandable by both humans and machines.

This ability to detach the semantics from the application code is following a trend of loose coupling more and more elements from the main program.

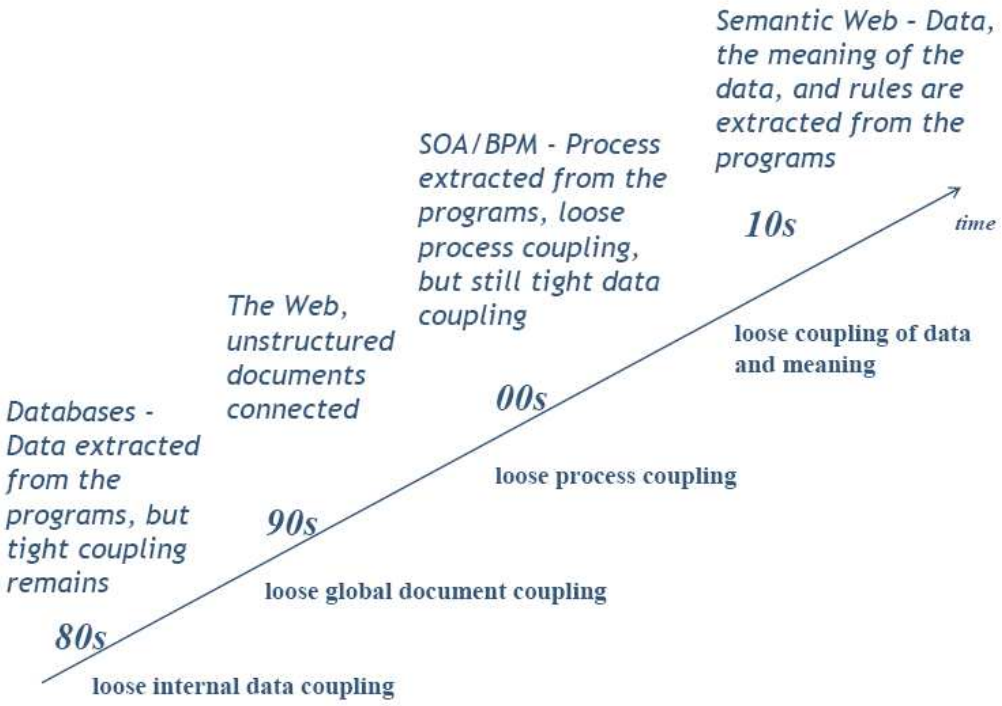


Figure 1 Elements of the ICT stack being loose coupled (Norheim 2007)

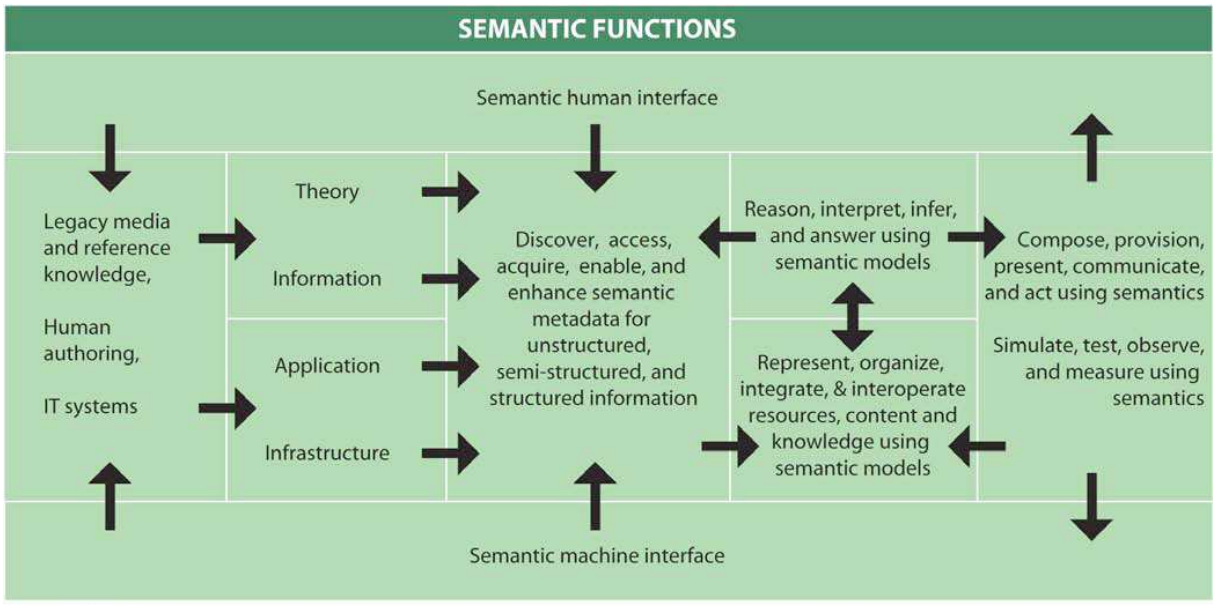
This trend started by inventing the two-layered software architecture extracting data from the applications into databases, giving a loose internal data coupling.

This trend continued when the World Wide Web enabled loose coupling to globally distributed documents.

And currently service-oriented architecture (SOA) and business process modelling (BPM) have removed processes from the programs providing a loose process coupling.

So, what is next?

What will the semantic technologies provide?



Source: MILLS•DAVIS

Figure 2 Semantic Functions or Capabilities

This figure extracted from a report called "Executive guide to Billion Dollar Markets" (Davis 2006) visualizes the functions filling the layer between humans and machines needed to make knowledge and meanings understandable by both humans and machines. If this is made understandable to the machines it will also become executable by the machines.

Capabilities to discover, extract, and model knowledge as well as enhance information with semantic metadata are needed to do automatic reasoning.

Information sources may be legacy media and reference knowledge and other IT systems. It is important that this is authored by humans.

Tools are used for recognizing patterns, syntax and structures within different data and language formats. Semantic tools provide capabilities for automatic discovery of topics and concepts. These tools may also extract the meaning of information provided, categorize, correlate and map between different sources.

The capabilities to reason, interpret and give answers are based on semantic models. Semantic models may be considered as organization of meanings. These models make use of taxonomies, ontologies and knowledge bases.

By using ontologies, semantic technologies can automatically discover and deploy semantic annotated web services and make use of the functionalities they provide. The sequence of making web service requests may be orchestrated and the responses may be reasoned and put together into composites that deliver a more comprehensive view of the data. This may result

in information in a context that may be new in relation to the context of the services providing the data.

Semantic technologies reason by declared associations, constraints, rules, conditions, and axioms that are represented in the ontology. The same source of information can be used to answer questions about how, why, and what-if as well as to provide pure facts.

And semantic technologies can directly search topics, concepts, associations in a number of sources, providing results that are more relevant than doing string based and index based searching as is more commonly used today.

Further, semantic technologies can deliver intelligence and answers to questions, not just provide lists of sources.

It is easy to modify ontologies by creating new concepts, relationships, properties, constraints and instances.

This means that applications built using semantic technologies are much more flexible because this technology is able to integrate data, content, applications, and processes by the means of a shared ontology. No software needs to be modified. This has a potential of minimizing development and maintenance costs drastically. Developers can spend less time on coding applications, but more time to make better models of the problem domain.

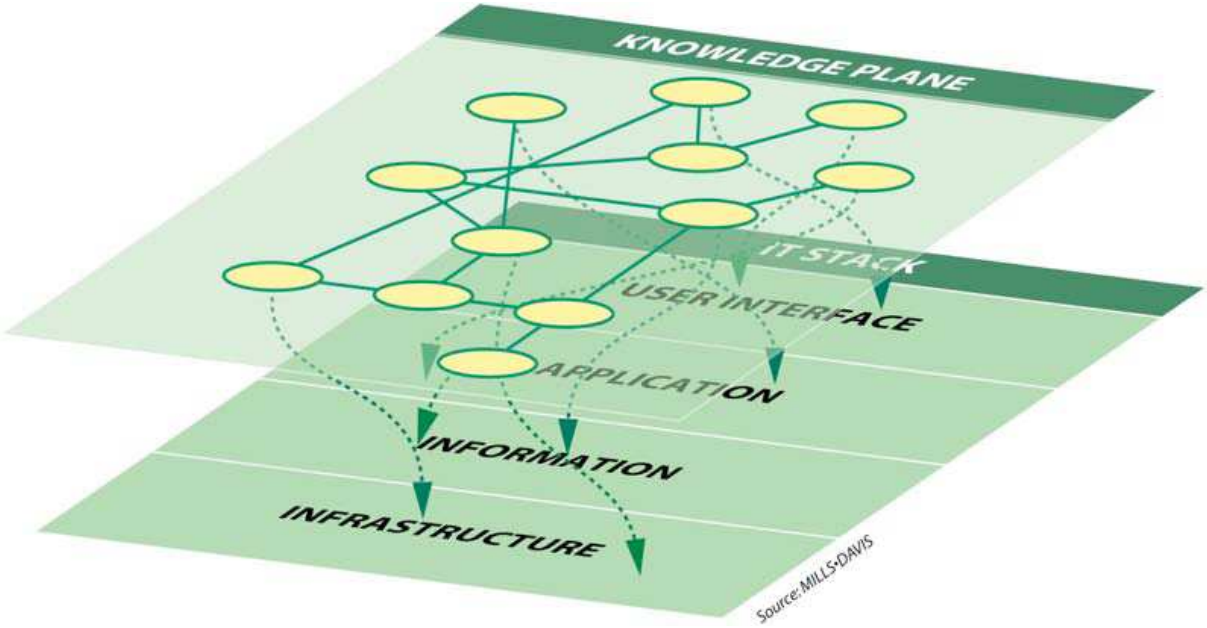


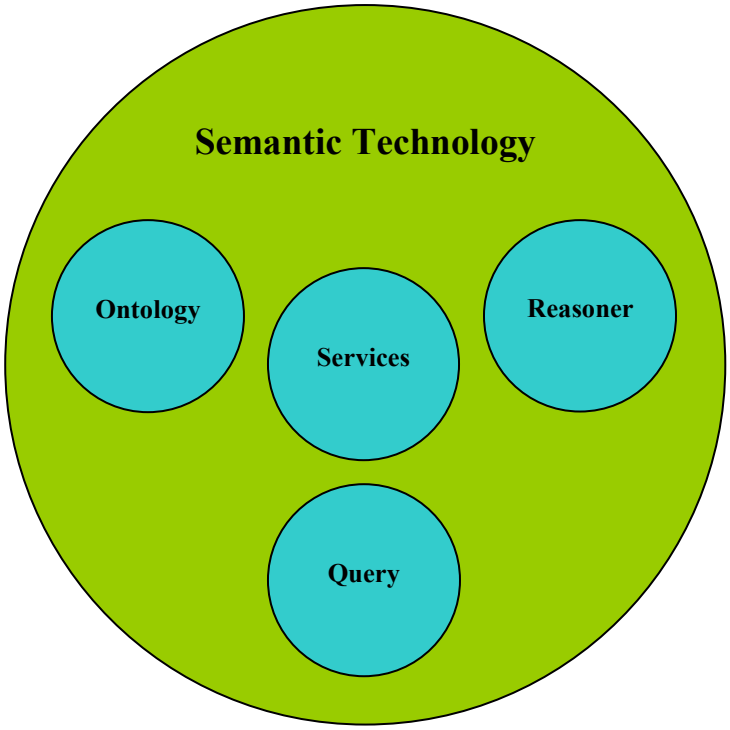
Figure 3 The knowledge plane will affect the complete ICT stack (Davis 2006)

Minimizing the needs to do changes in the source code will eventually make the programs more robust, and modifications to the models may be done during runtime.

Compared to traditional information technologies, semantic technologies offer tools to ease the making of more adaptable and flexible information and software. (Hansen, Gagnes et al. 2007)

Section 3: Core Technologies

In this section the 4 core semantic technologies is briefly described. The knowledge is represented in ontologies. These may be exposed to reasoning and rules. To be able to extract the information we need querying tools. All these components may be put together and executed in services.



Ontologies

The science of representing, storing and make knowledge accessible to the computers has its roots in the artificial intelligence (AI) community. The ability to make knowledge understandable to computers was believed to result in computer's ability to infer new conclusions from existing knowledge.

The Resource Description Framework, RDF is a graph based data model for representing knowledge. (Manola and Miller 2004)

RDF has an XML serialization, which makes it platform independent. RDF stores triples. A triple is a subject-predicate-object tuple, where subjects are resources and predicates specify the relationships between the subjects and the objects. When triples are combined a graph is created. An XML-document is a hierarchy that takes the form of a tree.

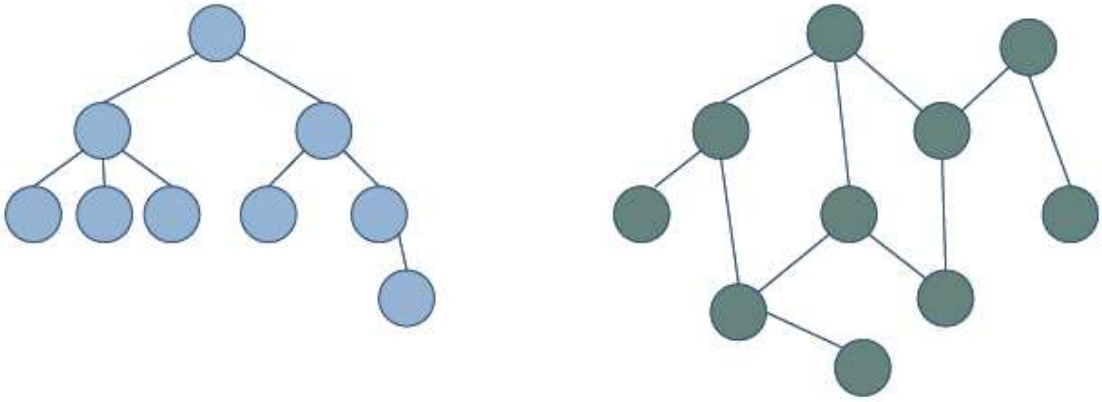


Figure 4 XML's tree model (left) compared to RDF's graph model (right)

An RDF model may be viewed in two ways a terminology box, TBox, for the concept definitions representing the general knowledge, and an assertion box, ABox, for the instances representing the knowledge specific to individuals of the domain.

An ontology is defined as a formal, explicit specification of a shared conceptualization. The terms are explained as follows:

- *Formal: The ontology should be computer readable. This means natural languages are excluded.*
- *Explicit: The type of concepts used and constraints on their use are explicitly defined.*
- *Shared: Reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group.*
- *Conceptualization: Refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon*.

(Studer, Benjamins et al. 1998)

An Ontology is a model of (some aspect of) the world that introduces vocabulary relevant to a domain of interest. It

- Includes names for classes and relationships.
- Specifies intended meaning of a vocabulary, that is formalized using suitable logic.
- Consists of two parts:
    - Axioms describing structure of the model
    - Facts describing some particular concrete situation

(Horrocks 2008)

To get a better understanding of what Ontologies are we can compare it to similar better known technologies.

Ontology vs.

- **Databases**: Both are used by applications at runtime. But as databases are has a closed world approach, which means that if an entity is not found it doesn't exist, whereas Ontologies have an open world approach, which means that an entity may exist even if it is not found in the ontology.

- **Object models**: Both describe classes and attributes. Unlike object models Ontologies are set based and dynamic, which means that an instance may belong to or even become another class if the content is modified in the ontology.
- **Business rules**: Both encode rules, but Ontologies organize rules using axioms, which belong to the Assertion box.
- **XML schemas**: Both are executable on the web. But unlike XML Schemas, Ontologies are graphs as opposed to trees. They can be used for reasoning.

(Horrocks 2008)

Ontologies can also be considered as a taxonomy with relations, limitations and rules. Taxonomy is a vocabulary presented in a structure.

Today a set of different languages may be used to declare an ontology. One way to evaluate if a model actually is an ontology is to evaluate the expressiveness of the language used to define it. A range going from simple word lists to Ontologies based on very expressive logics may be found.

The most commonly used language today is the Web Ontology Language, OWL. This is a recommendation from the World Wide Web Consortium, W3C. (Bechhofer, van Harmelen et al. 2004)

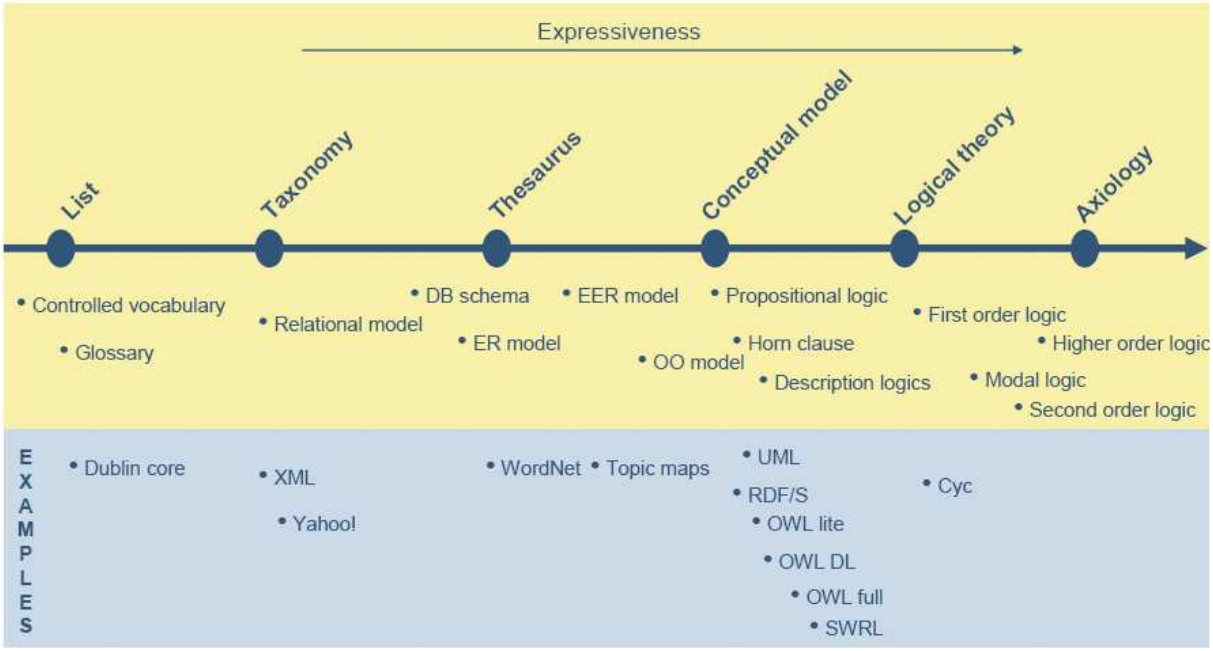


Figure 5 Expressiveness shown in an ontology spectrum. (Davis 2006)

Ontologies may be linked to other ontologies by having the exact same concepts or axioms. One ontology may also import another ontology and refer to concepts defined in this. This makes it possible to develop Ontologies in a structured and modular way.

When two different models has been developed that are semantically equal but syntactically or structurally different, the ontologies need to be bridged. This is ontology mapping and is used to solve interoperability on the semantic level.

Also metadata may be defined by using ontologies. This means that the metadata can have a higher precision in the definition of the meaning of the terms. This also gives us the ability to map ontologies on the metadata level.

Reasoning and Rules

Reasoning in connection with semantic technologies is used to check the consistency of the design of the ontology. This means that rules are defined to check that no concept definitions lead to contradictions.

Rules may also be defined to infer new information based on the information already present in the knowledge base.

The rules may be defined as axioms in the ontology itself or by customized rules setup by the user. If both types of rules are used it is called hybrid reasoning.

Using only axioms as basis for reasoning may incur limited expressivity for computing inferences as compared to modelling customized rules later.

An example of this in OWL is the fact that without custom rules it is not possible to reason that an individual B is the uncle of individual A based on the fact that B is the sibling of an individual C which in its turn is the parent of A.

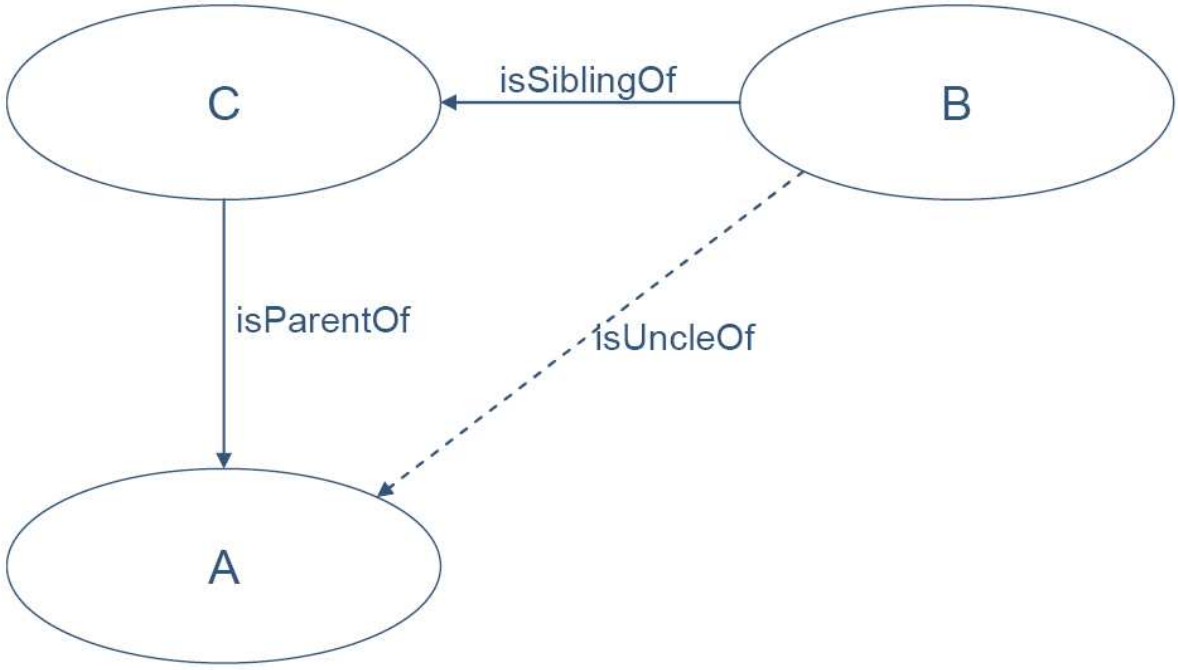


Figure 6 The OWL Uncle example (Hansen, Gagnes et al. 2007)

Such expressivity may be obtained by introducing a rule language to allow the user to provide rules. This also corresponds well to the idea that rules play an important part in encoding knowledge in a domain.

Querying

As for databases a querying language is important for extracting data from a knowledge base. In systems based on semantics data is typically stored in a knowledge base.

The main W3C effort for providing a querying tool is the SPARQL Protocol and RDF Query Language, SPARQL. (Feigenbaum, Clark et al. 2008)

SPARQL is as the name indicates a SQL like language to access RDF data over a network and a result format for returning result sets. SPARQL is most commonly used for accessing instance data.

The SPARQL syntax is very similar to the relational database querying language SQL, Structure Query Language. It has four different forms for querying:

- SELECT – Returns a table of results
- CONSTRUCT – Returns an RDF graph, based on an template
- ASK – Boolean query
- DESCRIBE – Returns an RDF Graph selected by the query processor

The remote query endpoint may be either SOAP or HTTP. The results is wrapped within an XML Query Result format.

An example query:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?book ?who
WHERE { ?book dc:creator ?who }
```

The result being:

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">

 <head>
   <variable name="book"/>
   <variable name="who"/>
 </head>
 <results distinct="false" ordered="false">
   <result>
     <binding name="book"><uri>http://www.example/book/book5</uri></binding>
     <binding name="who"><bnode>r29392923r2922</bnode></binding>
   </result>
...
</sparql>
```

(Feigenbaum, Clark et al. 2008)

Agents and Services

Agents being one of the initial initiatives within the semantic Web are defined, without human interaction to:

- **Autonomous** – be able to make decisions on behalf of the user.
- **Proactive** – be able to take initiative when appropriate
- **Social** – interact with other agents in order to complete a task

In addition the idea is that agents should be able to fulfill a user specified task or goal by:

- **Discover** – find all relevant semantic annotated web services to solve a this task
- **Negotiate and contract** – Choose the most appropriate services among the available
- **Compose** – combine services to compose a new service to achieve this task
- **Mediate** – do all necessary mapping to overcome heterogeneity.
- **Invoke & Monitor** - services following programmatic conventions

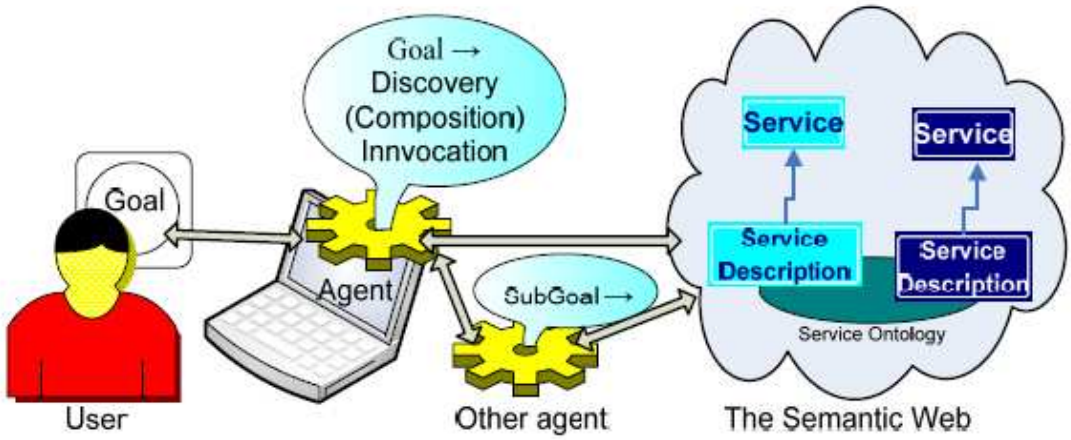


Figure 7 Agents utilizing Semantic Web Services using the same ontology (Hansen, Gagnes et al. 2007)

This figure shows how agents use the service ontology to discover services, compose more complex services, and invoke services to accomplish the tasks or goals that is specified by the user.

Service providers must describe their services and users define goals/requests according to the concepts of the service ontology in question. The agent uses the request and the descriptions to reason about all services discovered and to match the request to a service description.
A complex service might require the agent to invoke several services in order to reach the goal. Invocation is done by the agent and resolves the user goal. This is called service composition.

If the service requestor is using a different ontology from the one used to describe the service, the service requestor needs to link his request to the one used by the service in order for the Service Request to understand the service description.

This linking may be done either on the client side or by the use of a service registry.

There are still a lot of initiatives within the area of Semantic Web Services. There are discussions with regards to both how semantics is viewed by the web services and what logical languages to be used.

The main initiatives are currently:

- Web Ontology Language – Services, **OWL-S**, which originates from the work done by US Dept. of Defence in the DARPA Agent Markup Language project, DAML.
- Web Service Modelling Ontology, **WSMO**, run by the European Commission Science Foundation Ireland and Vienna city government.
- Semantic Web Services Framework, SWSF, which is a part of the DAML project.
- Semantic Annotations for WSDL, **SAWSDL**, derived from WSDL-S. WSDL-S is a project from IBM and the University of Georgia.

Even if most of these specifications have become rather complex, none of them is able to deliver all the functionality needed to accommodate the semantic web.

WSMO tends to be the most promising initiative, but has designed an ontology language and a framework for service descriptions using logic they considered to better represent services.

SAWSDL is accepted by the W3C as a recommendation. SAWSDL takes a more pragmatic approach by adding semantic annotations to WSDL files using external ontologies. There are no restrictions on the ontology language used to specify the concepts the services exchange. This makes it possible to use both existing and future external ontologies and to describe a service according to multiple ontology languages. As this specification doesn't jeopardize the existing WSDL protocol, ordinary web services and semantic annotated services can run side by side.

Section 4: The Semantic Web

The concept called The Semantic Web is a fundamental shift in paradigm. Tim Berners-Lee, the founder of the World Wide Web, has predicted the web to become a web of data as opposed to a web of documents. (Berners-Lee, Hendler et al. 2001)

The web content will be annotated to ontologies and the entire Web stack can be viewed as getting a knowledge backplane.

With the Semantic Web it will be easier for users (and applications) to get hold of data from databases, knowledge bases, documents, calendars, geo-systems and processing meters. The data from all kind of sources may be re-used, aggregated and even inferring new information.

W3C Semantic Web Standards

In developing the components of this Semantic Web, the World Wide Web Consortium (W3C) is providing technologies and standards. W3C aims to help and promote the development of the Semantic Web and to ease the implementation within and between enterprises and large organizations.

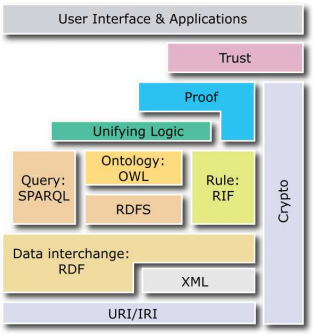


Figure 8. The Semantic Web Layer cake

Based on Uniform Resource Identifiers, URI, and/or XML RDF is the format for representing information on the Semantic Web. RDF, meaning Resource Description Framework, is often used to

represent, among other things, personal information, social networks, metadata about digital artifacts, as well as to provide means of integration over distributed sources of information.

The SPARQL query language for RDF is designed to query RDF represented data in an SQL mannered approach.

The OWL Web Ontology Language is a language for defining and instantiating Web ontologies. Ontology is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related.

RDF-S, which is RDF Semantics and OWL, is more expressive than RDF and will in this manner be more restrictive than RDF by itself.

RIF, Rule Interchange Format, is still under development. This format will be used to exchange formats between disparate rule systems.

The rest of the stack is still regarded as work to-be-done.

The Semantic Web is an extension of the current web where the use of knowledge representation techniques is envisioned to allow computers to collect data on the web on behalf of humans. (Hansen, Gagnes et al. 2007)

To make use of these capabilities the data or even all resources residing on the World Wide Web need to be linked to the metadata declared in an ontology. In order to do this annotation W3C is working on a couple of initiatives RDFa and GRDDL. RDFa is a syntax that enables RDF data to be embedded in HTML and in this manner becoming readable by humans by using a browser.

GRDDL is another initiative that creates annotations to web pages automatically by transformations specified in the web document itself.

Section 5: Selected Capabilities

In this section a selection of the capabilities already listed in the previous section is presented in more details. Only the functionalities that are of interest within the scope of this thesis will be described.

Semantic Service Oriented Architecture, SOA.

Service Oriented Architecture, SOA, is the paradigm the IT industry is relating to today. See Figure 1 Elements of the ICT stack being loose coupled (Norheim 2007).

SOA is an architectural style for creating and using described business processes, implemented as services. SOA also defines how to use the IT infrastructure to allow different applications to exchange data and participate in business processes.

These functions are loosely coupled with the existing infrastructure as operating systems, middleware and legacy applications. SOA separates functions into distinct services, which can be distributed over a network and can be combined and reused to create business applications.

These services communicate with each other by passing data from one service to another or by coordinating an activity between two or more services.

Loose coupling describes an approach where integration interfaces are developed without making assumptions on other service capabilities. This means that the risk that modifications in one application will require changes in other applications is reduced.

Integration between two applications may be loosely coupled by using Message-oriented middleware, MOM. By using MOM the availability of one system does not affect other systems.

Another way to obtain loose coupling in format is by using middleware to perform Data transformation, meaning that differences in data models is handled by some mapping tool. In Web Services or Service Oriented Architecture, loose coupling may mean simply that the implementation is hidden from the caller. See **Information Integration**.

Loosely coupled services, even if they use incompatible system technologies, may be joined to create composite services. Or they can be disassembled just as easily into their functional components.

The reason for doing this is that it becomes easier to change the orchestration as the business needs changes.

Web services

The most popular technology for implementing SOA today is web services.

Web Services provide means of interoperability. The protocols used for messaging are becoming standardized. But there are still issues in the area of data interoperability. Web Services use XML Schema to restrict the syntax in the messages.

XML Schema has no scalable way to link between various schemas. We may quickly end up with an exponentially growing numbers of point-to-point links. See Figure 9 Point to Point transformation (a) vs. centralized (b) information integration (Hansen, Gagnes et al. 2007)

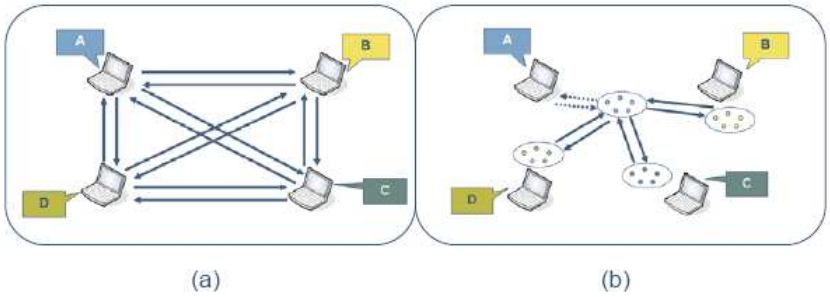


Figure 9 Point to Point transformation (a) vs. centralized (b) information integration (Hansen, Gagnes et al. 2007)

The users of a SOA can establish a shared semantic framework to ensure messages retain a consistent meaning across participating services. If this agreement is established, we are

moving into the domain of the **Semantic SOA**. See Figure 10 Semantic SOA shown as an interaction of Web services, Middleware and Semantic Technology (Hansen, Gagnes et al. 2007)

By creating a semantic layer with ontologies that represent the different schemas, it becomes possible to map different client ontologies to domain ontologies. The domain ontology may in turn be used to mediate between different client ontologies. See Figure 9 Point to Point transformation (a) vs. centralized (b) information integration (Hansen, Gagnes et al. 2007)

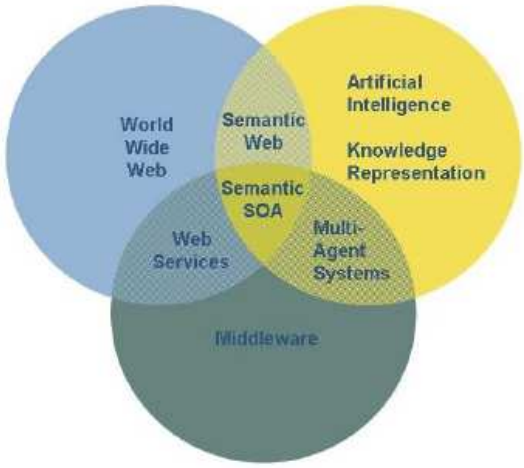


Figure 10 Semantic SOA shown as an interaction of Web services, Middleware and Semantic Technology (Hansen, Gagnes et al. 2007)

Information Integration and mediation

As discussed in chapter 'Agents and Services' discovery, negotiation, composition, mediation and enactment are areas where the use of semantic technologies is expected to enhance current SOA technologies.

Integration and/or mediation are probably the applications that are most mature, being the areas that have got the most focus up until now.

Integration

The main idea of semantic integration is to solve the mismatch between different formats, protocols and models at the semantic layer. Since ontologies are formal model based on logic, it is possible to use automatic reasoning tools to do the mapping. A requirement, however, is that these models are separated from the systems. This can be used to integrate information from heterogeneous sources automatically. This results in a model driven approach to solve mismatches at the semantic level.

By doing integration in this centralized way it is possible to reduce the exponential growth in point to point integration architecture to a near linear growth. The exponential growth, known as the n^2-problem, which is the maximum number of links required for n systems, is reduced to n integrations in a best case. See Figure 9 *Point to Point transformation (a) vs. centralized (b)* information integration (Hansen, Gagnes et al. 2007)

Most large organizations suffer from having many different systems that should share information. In addition the industry is experiencing a higher demand to adapt to new collaboration partners as fast as possible. Due to these changing requirements for information sharing the semantic approach seems to be promising as it makes information more adaptable and offers better supporting services.

Mediation

According to wikipedia **Mediation** is an activity in which a neutral third party, the mediator, assists two or more parties in order to help them achieve an agreement on a matter of common interest. In the context of the semantic web mediation can be regarded as the process of solving mismatches between services in terms of formats, protocols and languages.

Mediators are agents, most probably transparent to the endpoint services, which take care of the process of mediation. Format mediation is very close related to information integration, because it will involve information integration. In addition it will format the output to an appropriate format.

Protocol mediation is set out to resolve protocol mismatches between to services. One of the services could for instance use SOAP, whilst the other service is expecting HTTP. Or if they use the same protocol, one of the services requires two interactions while the other requires four.

Language mediation is set out to resolve the translation challenge between services, for instance an English speaking service should be able to communicate with a Norwegian speaking service.

Information Fusion

Another aspect of information integration is Information Fusion. Whilst information integration is merging data from disparate sources, information fusion is in addition the process of adding or inferring new information based on the information sources.

The term information fusion is in its turn often used interchangeably with data fusion, which can be defined as: *a formal framework in which are expressed means and tools for the alliance of data originating from different sources. It aims at obtaining information of greater quality; the exact definition of greater quality will depend upon the application.* (Wald, Teledetection et al. 1999)

Because relations are expressed logically in ontologies there are good reasons to expect that these technologies will be able to solve the information fusion challenge.

This area is expected to evolve in the near future as the rule engines become more standardized. As of now data or information fusion needs to be hard coded in the mediators.

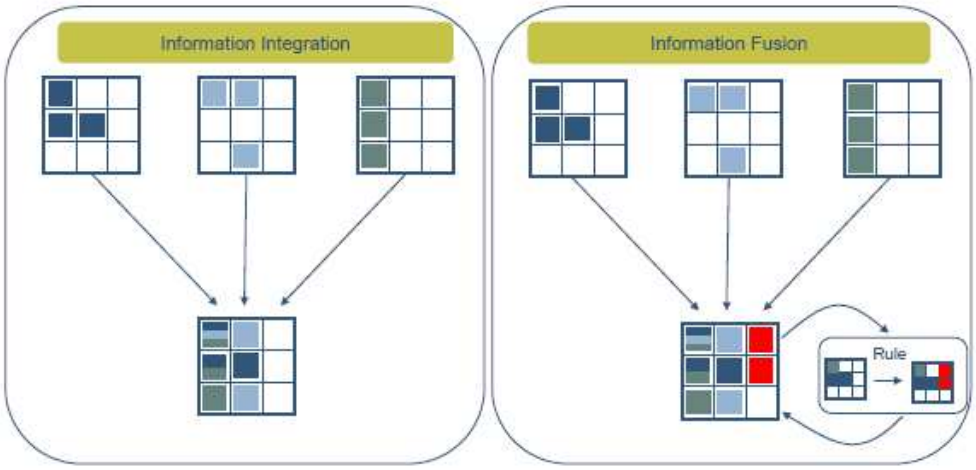


Figure 11 The difference between information integration and information fusion

Semantic Search

The traditional way of retrieving information is by doing text based searches. A user enters a set of words that describes the information wanted, the result being one or more documents where these words are present. The recent years indexing machines has become popular to make text based searches go faster.

Semantic searches are using semantic technologies in its search algorithms. This looks promising as a way to improve traditional searching strategies. Actors like Yahoo! and Google has invested a lot of research in this area.

The semantic technologies are providing means of handling concepts and relations. Every concept is related to other concepts in different ways, and by utilizing these relationships semantic search has the potential to provide search results with higher relevance to the user than traditional, text-based search alone.

One example:

A student has to call his supervisor. How can he find the e-mail with Henning's, his supervisor's phone number in it? He never uses the term "phone number," but writes things like "you can reach me at 46 41 30 31. Regards Henning"

Keyword search can't search for information about concepts like people's names or telephone numbers. Keyword search can't express relationships like "must occur in the same paragraph" or "the telephone number is in the footer of the mail". But semantics can!

So in the case above a number combination of the form "[+47]99999999" or an 8 digit number in an email sent from the .no domain is most likely a telephone number. And if "Henning", which by the way may be recognized as a name, appears in the same paragraph, this number is most likely Henning's phone number and may be returned as an answer to the query.

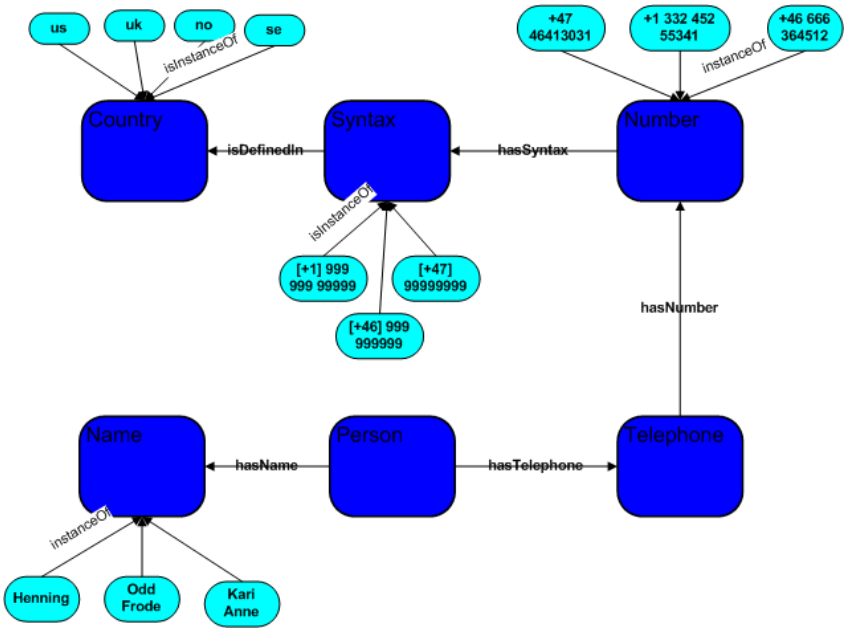


Figure 12 Ontology used to find telephone number based on syntax and name.

Section 6: Oil and Gas Applications of Semantic Technologies

Until now the potential of the semantic web technologies has been explored in general. This section will focus how these technologies may be used within the business of Oil & Gas, Drilling & Completion.

Integrated Operations in the High North

IO in the High North is a Joint Industry Project managed by DNV. The primary objective of this project is to enhance technology and develop prototypes for a digital infrastructure and a semantic platform to implement: Integrated Operations solutions. This new digital infrastructure shall facilitate a safe and cost efficient development of remotely managed operations in the northern parts of the Norwegian Sea. It will also be monitoring the environments of the hazardous conditions in the high North.

The infrastructure will be prototyped and consists of the following elements:

- **Real time information** between sensors, activators and nodes in a high capacity network.
- **Information transfer** by the use of web services and information validation.
- **Information integration**, by developing an oil and gas ontology to support the interpretation of sensor data, the platform for web services and information validation services

Furthermore, the developed prototype digital infrastructure and semantic platform shall be piloted to demonstrate the feasibility for unmanned drilling rigs, improved production operations and sub-ice operations.

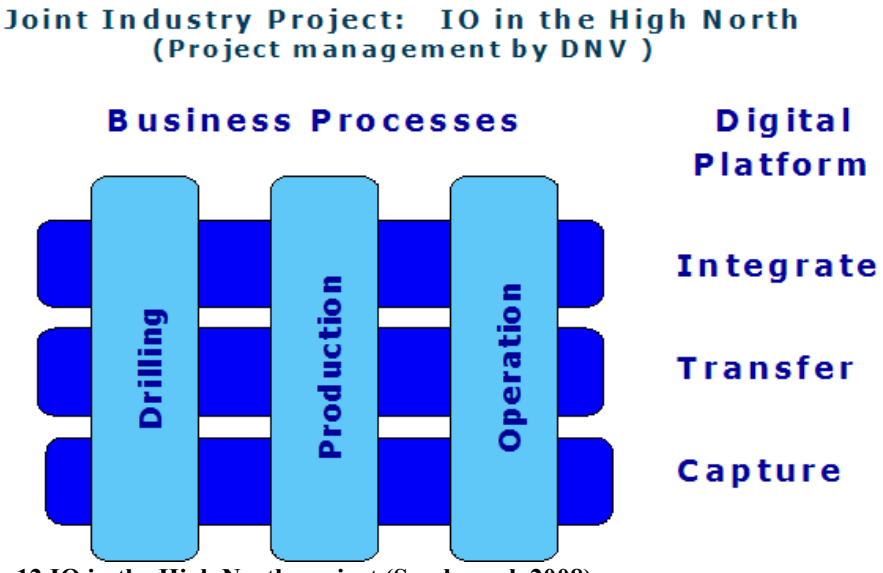


Figure 12 IO in the High North project (Sandsmark 2008)

This thesis shall evaluate SA-WSDL as a tool to develop the semantic platform for this project. This task is initiated by the PCA SIG Drilling & Completion.

Deploying SAWSDL in a pilot within Drilling & Completion

Business case

National Oilwell Varco is in cooperation with StatoilHydro, Computas AS, IRIS and two more major operating companies starting a RCN project which aims at creating a next-generation quantum leap technology for offshore Drilling Control Systems. In conjunction with this we are investigating the SAWSDL proposal from W3C and recommends that a proper Master Thesis is used as a conclusion and recommendation to the RCN project. Responsible for AutoConRig is Henning Jansen, National Oilwell Varco Stavanger

The AutoConRig project scope, quoted from the RCN application:

The nature of the Drilling industry imposes complex systems integration between several Independent vendors, service providers, disciplines and locations, whether it is a conventional drilling rig or an autonomous, semi-automatic and remotely operated (seabed) rig. The increased complexity due to a higher level of automation requires a higher level of integration capabilities than what is currently being used by the industry. A high-level integrated control system for autonomous and semi-automated drilling control needs standardization to serve the industry as a whole.

Both exchange formats and the underlying semantics will be implemented in the AutoConRig project and submitted for standardization. The technologies required are divided into two layers of integration and interoperability capabilities:

1) Conventional ICT service oriented architecture, (SOA, utilizing Web Services and XML) with an addition of Semantic Annotation for Ontology based data integration. The SOA integration layer is utilizing the XML-based WITSML standard for meta-data and contextual Drilling information (www.witsml.org), with an additional layer of semantic annotations

based on ontology in PCA (POSC Caesar Association) Reference Data System in accordance to ISO-15926. The AutoConRig project will develop a pilot for, and demonstrate the benefit of introducing Semantic Annotations to the WITSML standard. If applicable, additions to WITSML will be developed and submitted for standardization to Energistics, the governing body of the WITSML standard.

2) OPC industrial communication standard, targeting the new Unified Architecture - UA, standard. The OPC layer of the integration model developed in AutoConRig will utilize the new OPC UA (http://www.opcfoundation.org/UA) standard for real-time control and monitoring parameters. A drilling control specific standard will be developed on OPC UA, which will have a semantic foundation in ISO-15926 and the drilling related ontology in PCA's Reference Data System.

The challenge of coping with the increase of real-time data in Oil & Gas

The Norwegian Oil Industry Association, OLF, is the association for oil and supplier companies engaged in the field of exploration and production of oil and gas on the Norwegian Continental Shelf. OLF claims that only one-fourth of the estimated oil and gas resources on the Norwegian Shelf have been produced. One of the main tasks of OLF is finding solutions which help increase the efficiency and lifetime of fields and activities on the Norwegian shelf. In this context OLF is contributing in the Integrated Operations projects. OLF strongly believes that getting hold of more real time data will provide a foundation for doing better decisions, which again will lead to increased production. They have even estimated a possible 300 billion Norwegian kroner profit on this.

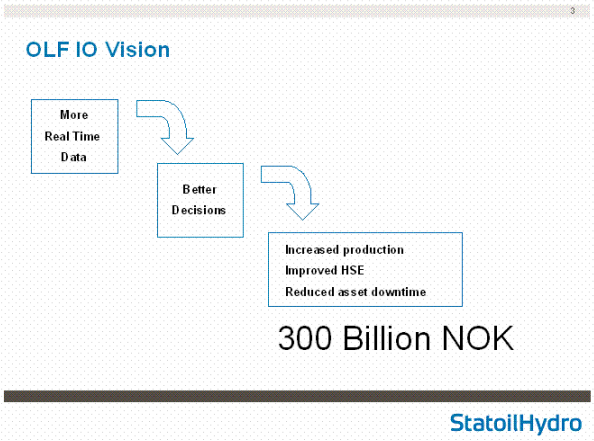


Figure 13 The OLF IO Vision

In the last few years the industry has managed to increase the real time data stream significantly. But, in stead of achieving better decision support, the industry now experiences an information overload that results in reduced decision quality. The reason for this is that the operators have not enough resources nor the ability to interpret this ever increasing flow of data. This in turn leads to human strain and stress that may reduce the quality of the decisions made by the operators.

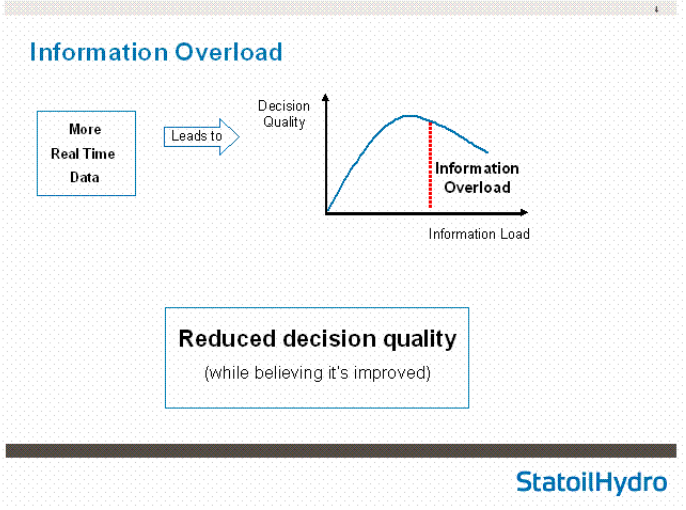


Figure 14 Resulted in information overload

This is the reason why the OLF IO projects now are looking into the technology of semantics. Is it possible to have the

computers take over some of the tasks of the humans? If so, when and how should the computer involve the humans? Is it possible to build trust between computers and humans? Is it possible to focus humans on only the most important problems and let the computer deal with the remaining issues?

Figure 15. The control room of Aasgard C. Can these operators in the future relate to only one screen of information?

This is why this industry now takes the first steps in evaluating the semantic technologies to see if these can provide any solution to all of these questions. (Fjellheim, Bratvold et al. 2008)

ISO-15926

ISO-15926 was originally meant to be an ISO standard for *"Industrial automation and integration – Integration of life-cycle data for process plants including oil and gas production facilities"* (Wikipedia) But the developers made it so generic that it is now regarded to be a standard for "*data integration, sharing, exchange, and hand-over between computer systems*".

ISO 15926 has 7 parts where part 4 contains the Reference Data and Part 7 describes the implementation architecture based on the W3C Recommendations for the Semantic Web. This architecture will enable integration of distributed systems.

This model and libraries are used to represent the lifecycle information of installations, for instance an oil and gas producing platform and its components, for instance wells and completions.

ISO 15926 part 4 is the "Core Library", or the RDL, Reference Data Library. An RDL is like a class library which also contains reference individuals. A class in the RDL is the definition of a type, a kind. Individuals are not classes, they are unique. But an individual is member of one or more classes. To be a member of a class in ISO 15926 is called "to be classified as". The RDL contains classes and a few reference individuals. A reference individual is an individual that is referenced so often that it makes sense to store it in the RDL. Examples of reference individuals are: London (city), Germany (country), Shell (company).

The RDL can be browsed by using the RDS, The Reference Data System, developed by POSC/CAESAR.

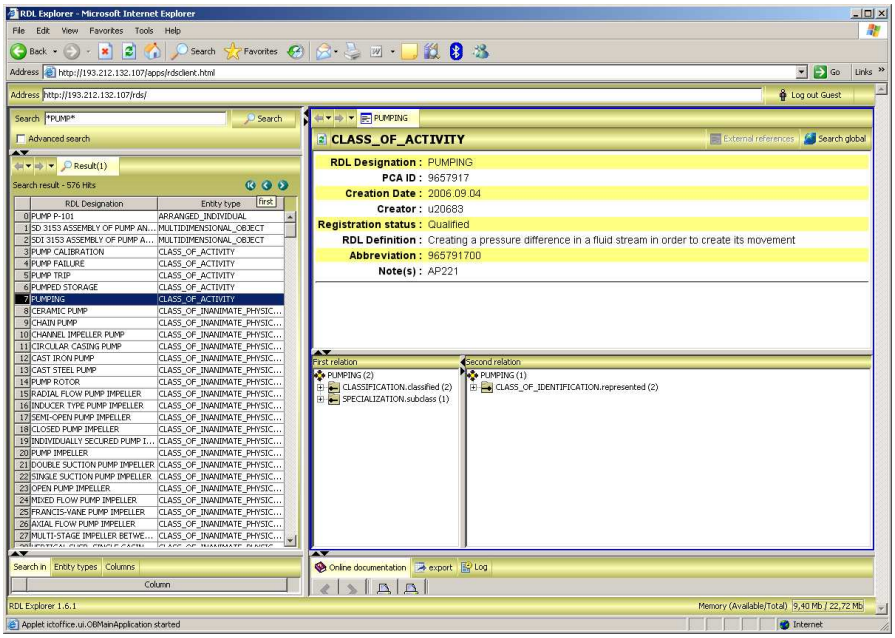


Figure 16. Snapshot of RDS

The ambition of ISO-15926 is to provide a standardized data model for all kind of facilities, thus enabling the industry to harmonize their internal proprietary data models. By harmonizing data models two or more models, usually within a domain of interest, is compared with the goal of reducing data redundancy and inconsistencies and improving the quality and format of data. This is usually done to perform data mapping, data normalization, or data integration. RDL provides interoperability to the industry, which means that data may be compared and exchanged.

Utilizing IS0 15926 enables organizations to meet their asset information requirements. The costs associated with defining, collecting, transforming, deploying and sustaining this information over the lifecycle of assets and facilities are reduced as the task is reduced to find the appropriate reference in RDL.

However, the number of instances and triples tells us that the RDL repository is very large, which may imply search performance issues and cumbersome mapping. Also the defined number of relationships is low compared to the number of classes. Due to these facts, the RDL will not be used as an ontology in this thesis. A mapping ontology that may be regarded as a snapshot of the RDL tailored to the needs of this thesis will be designed and used in stead.

Number of top-level classes	13
Number of sub-classes (is-a)	221
Number of relationships	16
Number of sub ontologies	2
Number of instances (Experiences)	Ca 11.600*
Number of triples	Ca 500.000*

AKSIO Drilling Ontology statistics. * Instances is increasing in the order of 5/day
(Fjellheim and Norheim 2007)

WitsML

WITSML, the Well site Information Transfer Standard Mark up Language, is a standard for exchanging data within the domain of drilling. It is developed by Energistics. Energistics is an international non-profit standardization organisation. Energistics hosts SIGs, Special Interest Groups, to gather the industry, solve issues and come up with ideas on how to improve the collaboration within this industry.

The objective of WITSML is gathering data at real time, seamless flow of well data between operators and service companies to speed up and enhance decision making.

WITSML is a standard for sending well site information in an XML document format between business partners. XML schemas are used to define the content of an XML document. The WITSML *data schema* consists of a **set** of independent but related data object schemas. A *data object schema* defines a set of data that can be transmitted within a single XML document (e.g.; well, well bore, rig, etc.). Data object schemas contain attributes, elements, and included component sub-schemas.

The following objects are defined in the WITSML standard Cement job, Conventional Core, Distributed Temperature Survey, Fluids Report, Formation Marker, Log, Message, Mud Log, Operational Report, Rig, Survey Program, Trajectory Station, Tubular, Well and Wellbore. In this thesis we are particular interested in the **SurveyProgram** object which is included in the defined pilot.

Daily Drilling Report - DDR

The Daily Drilling Report, required by the Petroleum Safety Authority is used to report ongoing drilling activities on a daily basis. This report covers information from wells drilled on the Norwegian Continental shelf. The format on this report is based on the WITSML standard specification. The pilot deployed in this thesis evaluates how to utilize SAWSDL to be able to map data into this format.

The following *Report part VII: Survey* of the DDR specification lists the elements and attributes of the Survey object:

All types of directional surveys measuring azimuth and/or inclination have to be reported.

WITSML tag	**Ref.**	**Units**	**Description/ general remarks**
< surveyStation ><dTim>		date	The date at which the directional survey took place
<surveyStation><md>	3.5-MD	meter	Measured depth (RKB)
<surveyStation><tvd>	3.5-TVD	meter	True vertical depth (RKB)
<surveyStation><azi>	3.5-AZIMUTH	degree	Measured azimuth in degrees (0 - 360)
<surveyStation><incl>	3.5-INCL	degree	Measured inclination. If the inclination should be measured two times at the same depth, and with different results, the last reported inclination mesurement will be

			considered valid.

OPC UA

A Distributed Control System, DCS, refers to a control system used in the process industry that is located at the processing facility.

A PLC is a programmable logic controller which is a computer used for automation of industrial processes. A PLC is a real time system since output results must be produced in response to certain input conditions within a limited time, otherwise some unintended operations could be performed. PLCs are usually connected to sensors and actuators. PLCs read analogue input variables like temperature and pressure and may operate electric motors, pneumatic or hydraulic cylinders and a lot of other components.

Both PLC and DCS typically uses custom designed processors, proprietary interconnections and protocols for communication. This, of course, results in a lot of resources spent on writing low level interfaces to PLC or DCS that seldom or never become documented.

This forced the industry to develop a standard process interface, the OPC. The first standard resulted from the collaboration of a number of leading worldwide automation suppliers working in cooperation with Microsoft. Originally based on Microsoft's OLE COM (component object model) and DCOM (distributed component object model) technologies, the specification defined a standard set of objects, interfaces and methods for use in process control and manufacturing automation applications to facilitate interoperability. The COM/DCOM technologies provided the framework for software products to be developed. There are now hundreds of OPC Data Access servers and clients.

OPC Unified Architecture

The existing OPC COM based specifications, OPC-DA is more than 10 years, and as technology moved on new requirements of interoperability had to be met. In order to become vendor independent the standard now aims to become cross-platform capable by using Web Services and SOA.

The Unified Architecture, OPC-UA, is described in a layered set of specifications broken into Parts. It is purposely described in abstract terms and in later parts married to existing technology on which software can be built. This layering is on purpose and helps isolate changes in OPC-UA from changes in the technology used to implement it.

OPC-DA XML

Provides flexible, consistent rules and formats for exposing processing plant floor data using XML, SOAP and Web Services.

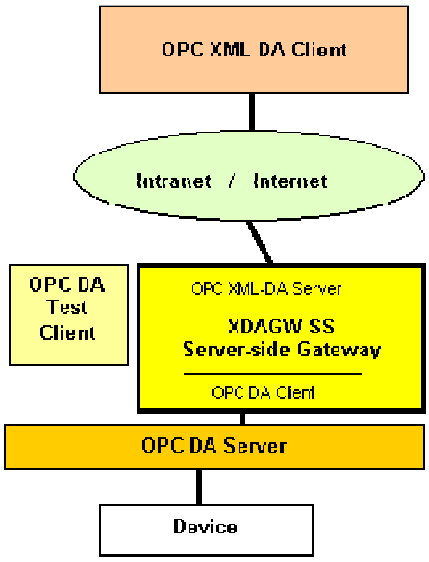


Figure 17. OPC-DA XML Architecture

Advosol

Advosol Inc. is a company that is a supplier of OPC software components and tools. They kindly has made available a number of free trial downloads and access to a XML-DA Server side gateway. This was most valuable in exploiting the OPC-DA XML services.

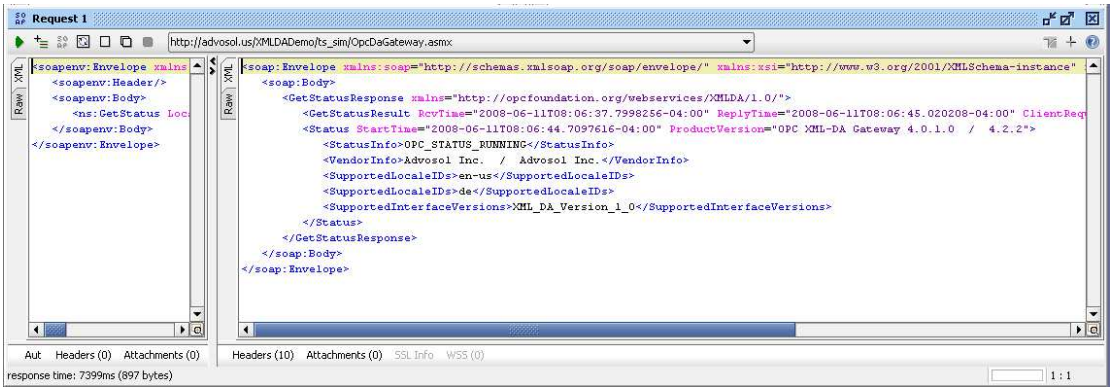


Figure 18. Example of output from the Advosol XML Gateway Service

These services was used to get templates showing what the XML looked like, in order to create XML outputs that were used in the SurveyStation pilot in this thesis.

Issues

The area of the semantic web is still very young and immature. This means of course that there are still a lot of issues to be resolved regarding semantic technologies. This section focuses on some of the areas that influenced setting up the prototype.

Several Semantic Web Services initiatives

There have been several initiatives in how to merge or annotate semantics to web services. This project used a lot of resources in exploring the different initiatives to resolve which technology was the most feasible approach in solving the task.

Given the immaturity of the area, there are still discussions and disagreements among researchers working on different initiatives. This fact means that they cannot agree on standards and common best practices. There are hardly any commercial vendors that will develop tools for this industry and the Open Source community tends to develop their own standards. This is delaying the availability of development tools and execution environments that will fulfill the requirements of production quality systems.

In this thesis SAWSDL is the product to be evaluated. Although the SAWSDL extension to WSDL has become a recommendation by W3C (Joel Farrel and Holger Lausen 2007), SAWSDL is a limited subset of the tasks the Semantic Web Services aims to solve.

So, additional tools are required to fulfill the task of developing a pilot that fulfills all capabilities of the semantic web.

As this arena gains more maturity, history has proven that standards will evolve and that the different initiatives will converge into these standards.

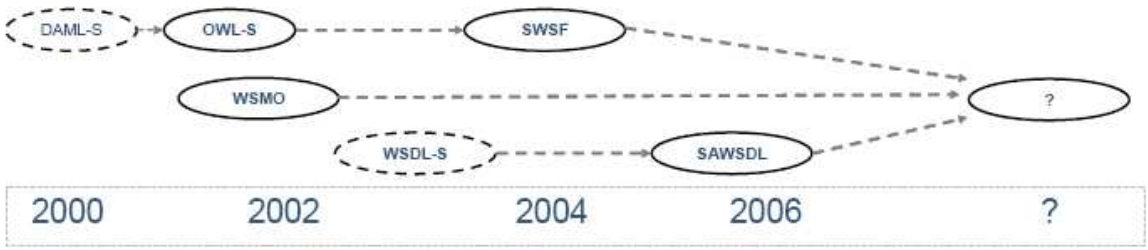


Figure 19 When will the different Semantic Web Service initiatives merge? (Hansen, Gagnes et al. 2007)

SAWSDL

The Web Services Description Language (WSDL) specifies a way to describe the functionalities of a web service and how and where to invoke it. The WSDL W3C Recommendation does not include semantics in the description of Web services. Therefore, two services can have similar descriptions but having different meanings, or they can have different descriptions but similar meaning. Resolving these ambiguities in Web services descriptions is an important step toward automating the discovery and composition of Web services. Being able to discover and compose new web services based on existing web services will give us a more dynamic **web of data**, which would be an improvement to the current **web of documents**.

SAWSDL defines a set of extension attributes for the Web Services Description Language and XML Schema definition language. This opens up for giving description of semantics in the WSDL. The specification defines how semantic annotation is accomplished by the use of references to semantic models. SAWSDL is NOT a language for representing the semantic models. Instead it provides mechanisms by which concepts from the semantic models, defined in an ontology or in a Resource Description Format, RDF can be referenced from within WSDL and XML Schema components using annotations. These semantics when expressed in formal languages can help disambiguate the description of Web services during automatic discovery and composition of the Web services. (Joel Farrel and Holger Lausen 2007) SAWSDL was recommended by W3C August 2007.

SAWSDL defines 3 extension attributes; *modelReference*, *liftingSchemaMapping* and *loweringSchemaMapping*. The modelReference is used to specify the association between a WSDL or XML Schema component and a concept in some semantic model. It is used to annotate XML Schema type definitions, element declarations, and attribute declarations as well as WSDL interfaces, operations, and faults.

The liftingSchemaMapping and loweringSchemaMapping extension attributes are added to XML Schema element declarations and type definitions for specifying mappings between semantic data and XML. The modelReference is used to directly reference a concept in a semantic model. If a component or element cannot be referenced directly, liftingSchemaMapping and loweringSchemaMapping may be used to point to data mapping transformation scripts or procedures. Lifting allows transforming from XML to semantic data and Lowering is used to transform from semantic data to XML. (Wilms 2007)

Section 7: Tests

Prototype

The purpose of the prototype is to demonstrate the potential usefulness of SAWSDL as a way to link the knowledge plane to the infrastructure of common Oil & Gas upstream data integration environments. The prototype is demonstrating how to map a small subset of Daily Drilling Report – DDR and a simplified proprietary dataset in OPC-UA. The purpose of the prototype is to be able to receive a Survey Station in OPC-UA format and express it as DDR.

Survey Station

A Survey Station is a point in the drilling process where a measurement of the inclination and azimuth of the borehole is performed. The Survey Station (the measured point) is used to calculate the trajectory of the well.

Daily Drilling Report - Survey Station

The Daily Drilling Reporting schema has a Survey Station complex element. The survey station consists of the following five fields:

- dTim - Date/Time for Survey Operation
- md - Measured Depth
- tvd - True Vertical Depth
- incl - Hole inclination
- azi - Hole azimuth

OPC-UA data structure

Consists of 4 fields, the datetime is omitted.

- MDEPTH - Measured Depth
- TVDEPTH - True Vertical Depth
- INCL_V_DEG - inclination in degrees measured from vertical plane
- AZMH_TN_DEG - azimuth in degrees measured from true north

DrillReportSurvey - Ontology used

Initially, the ISO-15926 ontology was meant to be utilized as the common ontology. But, as this ontology more or less lacks properties, another ontology, the *DrillReportSurvey* Ontology, was created. Ontology or OWL properties describe the relationship among classes or instances of classes. This means that without any properties no reasoning can be performed. As reasoning is one of the most important aspects within the semantic web, it was decided to create an ISO-15926 'footprint' that defined such properties or relationships.

Initially the classes were named according to ISO-15926. The DrillReportSurvey ontology contains all classes needed to describe a simple DrillReportSurvey as needed for this prototype (see Prototype). As these naming remains it is a fairly straight forward task to link the classes to the ISO-15926 classes. The ISO-15296 will in this thesis be regarded as a set of references.

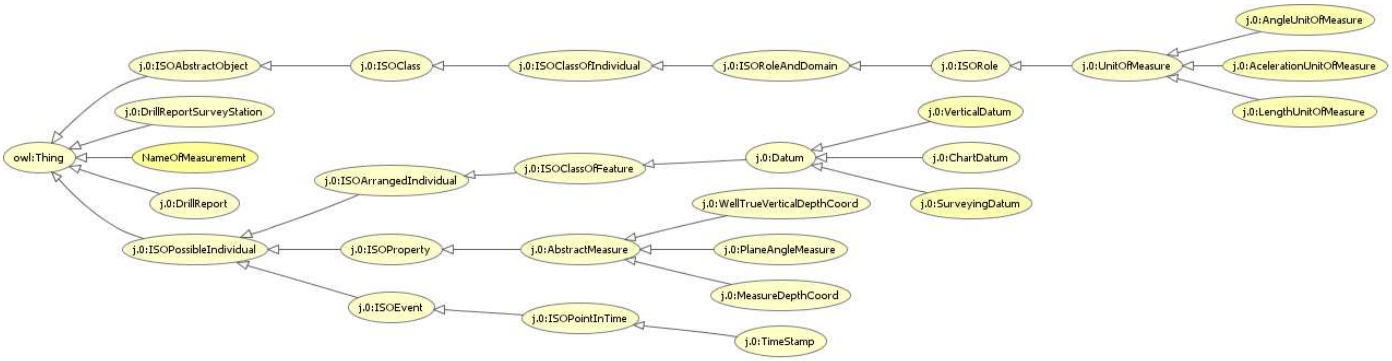


Figure 20 DrillReportSurvey

Development environment

Initially a lot of time was used to explore and find development tools that were needed to enable the development of the prototype. Finally the project ended up with the following list of tools needed to create a prototype.

- **Eclipse IDE** – Integrated Development Environment
- **Axis 2** – Java framework to create server and clients for web services (Eclipse plug-in)
- **SoapUI** – Tool used to test and exploit existing web services (Eclipse plug-in)
- **SAWSDL** – Standard for annotating semantics to existing web services
- **SAWSDL4J** – Java API that features methods for retrieving references in the wsdl
- **WSMO** – GUI used to annotate web services using SAWSDL (Eclipse plug-in)
- **XSLT** – Programming language to transform XML to other formats
- **SPARQL** – Structured Query Language for Querying RDF and OWL
- **Protégé** – Tool used to develop Ontologies using RDF and OWL
- **RacerPro** – Tool used to do automated reasoning

.

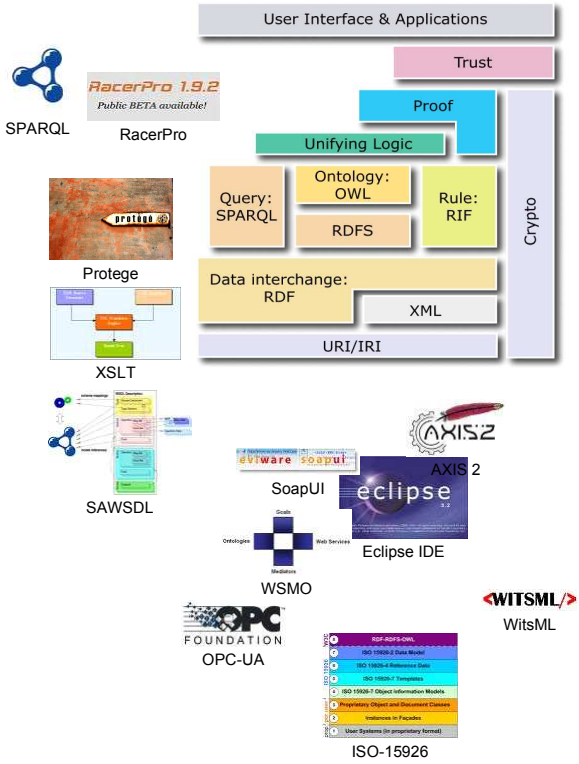


Figure 21. The Software stack deployed in this thesis

Eclipse

Eclipse is an integrated development environment used by Java developers. Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. The research and development projects within the semantic web have to a large degree made their tools available as eclipse plug-ins. This made eclipse a natural choice for setting up a development environment.

Apache Axis 2

Apache Axis 2 is a core engine for web services. It provides functionality to create clients to SOAP web services in Java. This enabled the project to easily create clients that could access the WitsML and OPC UA web services.

SoapUI

SoapUI is a software testing tool for Software Oriented Architecture used by the software developer. This tool was used to call real WitsML and OPC UA web services. The results were returned as XML documents contained in SOAP envelopes. This made it really easy to test the web services and explore the XML formats.

The XML outputs from the real web services were used to create SurveyStation XML Response documents. These responses were used to setup a mock web service that was run locally. In this way it was possible to simulate OPC DA XML web services that returned the 'hard coded' SurveyStation response document.

The next step was to implement a service agent that requested this mock service and got the response in return and for further processing.

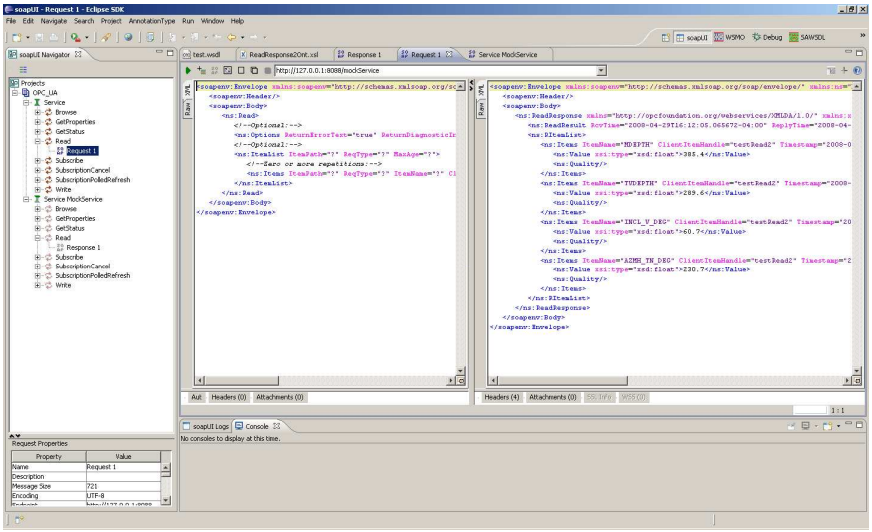


Figure 22. The SoapUI plug-in for the Eclipse IDE

Protégé

Protégé is a free, open source, ontology editor and knowledge base framework. It supports means of modeling Ontologies in OWL. This tool was used to model the OWL Ontologies used in the test cases.

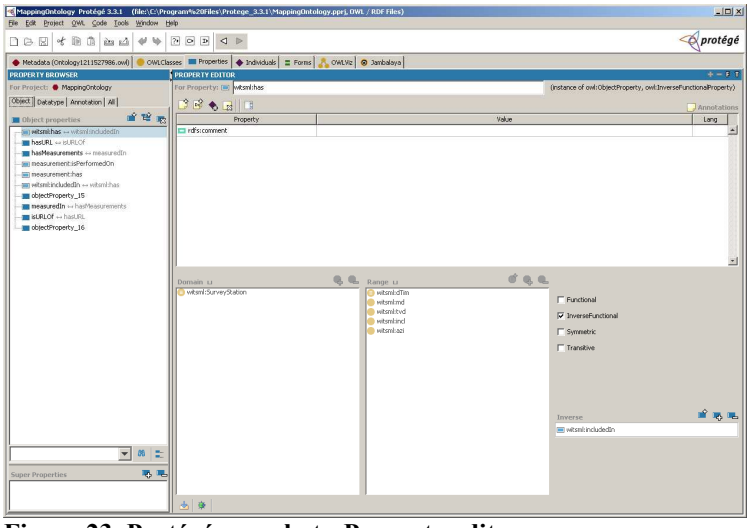


Figure 23. Protégé snapshot - Property editor

RacerPro

RacerPro is an OWL reasoner and inference server for the semantic web. RacerPro can also be seen as a semantic web information repository with optimized retrieval engine because it can handle large sets of data descriptions (e.g., defined using RDF). RacerPro was used to test reasoning mechanisms needed in the test cases of this thesis. RacerPro was deployed as a Protégé plug-in.

WSMO

Web Service Modelling Ontology, WSMO, is a developer framework for developing semantic applications. It supports development of agents that can discover and orchestrate web services. It also has a GUI to annotate ontology classes or instances to web service elements.

In the tests this GUI was used to do annotations. The WSMO module was deployed as an eclipse plug-in. The annotations were created by drag and drop functionality within the Eclipse IDE.

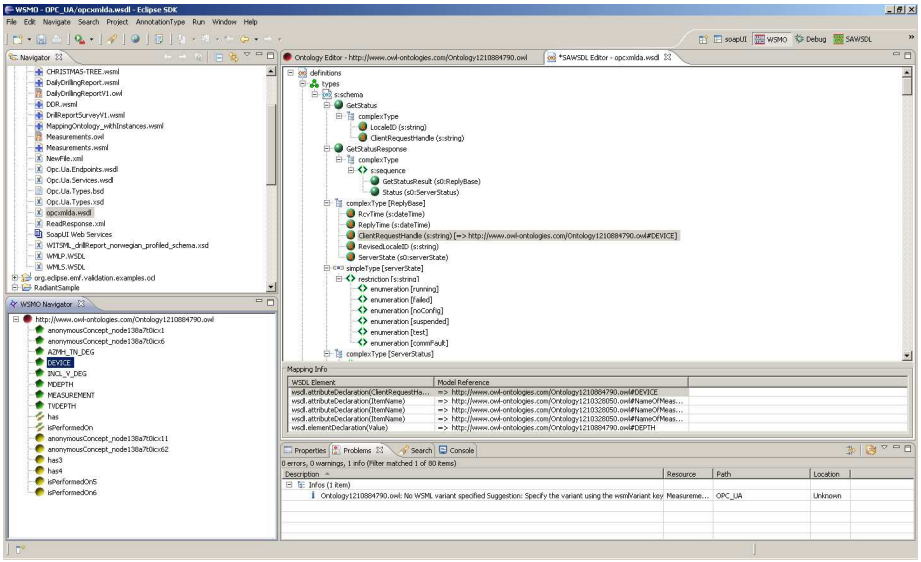


Figure 24. The WSMO Eclipse plug-in in action

Testing procedures

This chapter contains a brief description of how the tests were set up and how they were run. Some of the procedures are described in more detail in the Appendix E Testing procedures.

1. Use Protégé to define the OWL ontology to be used in the test
2. Import the OWL ontology into Eclipse by using the WSMO Eclipse plug-in.
3. Annotate the OWL classes and instances to elements in the web services to be used in the test by using the SAWSDL Editor in Eclipse IDE.
4. Setup the web services needed in the test by creating mock services. The mocked web services are created and deployed by the SoapUI Eclipse plug-in.
5. Implement agents in Java and run them within the Eclipse IDE. The SAWSDL4J Java API is used to get the annotated OWL references from the wsdl.
6. Evaluate any OWL output from the test by utilizing the RacerPro reasoning utility in Protégé.

Case 1: Matching Web Service Interfaces using a Shared Ontology

One of the main motivations for the SAWSDL specification is to provide mechanisms so that these semantics can be used to help automate the matching and composition of Web services.

In this section we present an example to show how to add such annotations for use during Web service interface matching and composition.

Consider the following scenario. A requestor submits a request to verify the existence of a certain parameter, 'MDEPTH'. This request is represented as a Browse Request operation in the OPC UA wsdl and a Get_Capabilities in the WitsML wsdl.

The OPC DA web services display the inputs it can supply and the outputs it expects of an item in the browse request service. The requestor will need to know that the Measured Depth value is found in an Item tag with a name that equals 'MDEPTH'.

```
<soap:Envelope  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
                xmlns:ns="http://opcfoundation.org/webservices/XMLDA/1.0/">
   <soap:Header/>
   <soap:Body>
      <ns:Browse LocaleID="nb-No" ClientRequestHandle="testBrowse" ItemPath=""
        ItemName="MDEPTH" ContinuationPoint="" MaxElementsReturned="5" BrowseFilter="all"
        ReturnAllProperties="true" ReturnPropertyValues="true" ReturnErrorText="true">
      </ns:Browse>
   </soap:Body>
</soap:Envelope>
```

Figure 25 OPC DA XML Browser Request

The Browse Response result would look something like this:

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Body>
      <BrowseResponse ContinuationPoint="" xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/">
         <BrowseResult RcvTime="2008-04-29T15:49:55.9144464-04:00" ReplyTime="2008-04-29T15:49:55.9144464-04:00" ClientRequestHandle="testBrowse" RevisedLocaleID="en-us"
               <Elements Name="mDepth" ItemName="Location.mDepth" IsItem="true" HasChildren="false">
                     <Properties Name="dataType" Description="Item Canonical DataType">
                           <Value xsi:type="xsd:QName">xsd:short</Value>
                     </Properties>
                     <Properties Name="value" Description="Item Value">
                           <Value xsi:type="xsd:float">398.5</Value>
                     </Properties>
                     <Properties Name="quality" Description="Item Quality">
                           <Value xsi:type="OPCQuality"/>
                     </Properties>
                     <Properties Name="timestamp" Description="Item Timestamp">
                           <Value xsi:type="xsd:dateTime">2008-04-29T15:49:55.754216-04:00</Value>
                     </Properties>
                     <Properties Name="accessRights" Description="Item Access Rights">
                           <Value xsi:type="xsd:string">readWritable</Value>
                     </Properties>
                     <Properties Name="description" Description="Item Description">
                           <Value xsi:type="xsd:string">Measured depth in wellbore</Value>
                     </Properties>
               </Elements>
         </BrowseResult>
      </BrowseResponse>
   </soap:Body>
</soap:Envelope>
```

Figure 26 OPC DA XML BrowseResponse

The WitsML service provider has a subscription request service that specifies, among other things, that the Subscriber wishes to receive the **WITSML** real time data objects. The Subscriber can determine what data object types are available by invoking the Publisher's GetCap (Get Capabilities) function, denoted by (1) in the diagram below.

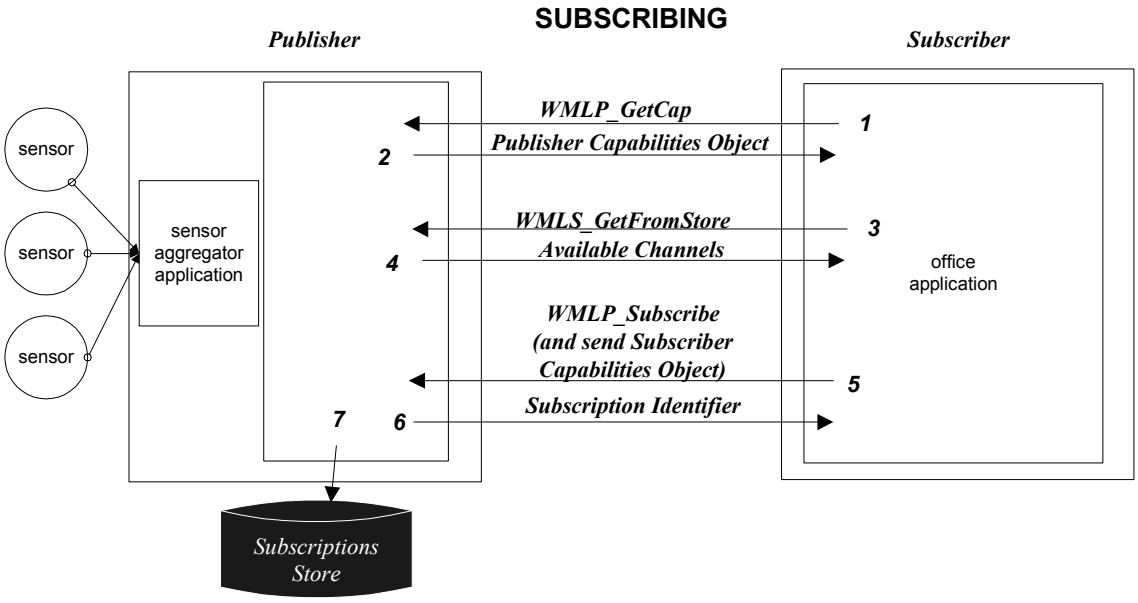


Figure 27 WitsML Publisher/Subscriber interaction (Members 2003)

The data returned from WitsML are organized in objects. 5 different objects exist; Well, Wellbore, Rig, Trajectory and TrajectoryStation. The GetCap operation will return the objects available for a given WitsML service provider. The XML 'mdCurrent' placeholder is included in the wellbore object.

At a high level the service offered by WitsML should match the OPC DA. However, the differences in the vocabulary used by the two services prevent making a match.

For example, the term Elements with an attribute called itemName that has a value of 'location.mdepth' used by OPC DA and the term mdCurrent used by the WitsML wellbore object are meant to uniquely identify the item in question.

```
<wellbore uidWell="W-1" uid="B-001">
      <nameWell/>
      <name/>
      <number/>
      <suffixAPI>x</suffixAPI>
      <numGovt/>
      <statusWellbore/>
      <purposeWellbore/>
      <typeWellbore/>
      <shape/>
      <dTimKickoff/>
      <mdCurrent uom="metre">398.5</mdCurrent>
      <tvdCurrent uom=""/>
      <mdKickoff uom=""/>
      <tvdKickoff uom=""/>
      <mdPlanned uom=""/>
      <tvdPlanned uom=""/>
      <mdSubSeaPlanned uom=""/>
      <tvdSubSeaPlanned uom=""/>
      <dayTarget/>
</wellbore>
```

A matching engine may not have sufficient information to identify them as related terms unless explicitly specified. Semantic annotations could be significant in order to identify terms.

If there were to be a common semantic model that can be used to annotate the WSDL of the OPC DA and the WitsML service provider, then a semantic engine could use this information to match the two Web services.

This case will try to annotate the elements in question to a common concept defined in a common ontology. We will use the DrillReportSurvey ontology defined in the section DrillReportSurvey - Ontology used.

In this case, annotation is done using modelReference extensibility attribute defined in SAWSDL.

The following ontology modifications and annotations were done in OPC-UA web services

The *NameOfMeasurement* class was added to the ontology as there was no placeholder for the ItemName in the DrillReportSurvey. The properties *isLinkedToMeasurement* and the inverse *hasMeasurementName* were defined between NameOfMeasurement and MeasuredDepthCoord.

NameOfMeasurement was annotated to *ItemName* in the *Browse* Request operation
NameOfMeasurement was annotated to *ItemName* in the *BrowseElement* complex type which is included in the BrowseResult which is the Browse Response message returned.

```
<s:element name="Browse">
    <s:complexType>
        <s:sequence>
            <s:element maxOccurs="unbounded" minOccurs="0"
                name="PropertyNames" type="s:QName"/>
        </s:sequence>
        <s:attribute name="LocaleID" type="s:string"/>
        <s:attribute name="ClientRequestHandle" type="s:string"/>
        <s:attribute name="ItemPath" type="s:string"/>
        <s:attribute name="ItemName"
            sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1210328050.owl#NameOfMeasurement" type="s:string"/>
        <s:attribute name="ContinuationPoint" type="s:string"/>
        <s:attribute default="0" name="MaxElementsReturned" type="s:int"/>
        <s:attribute default="all" name="BrowseFilter" type="s0:browseFilter"/>
        <s:attribute name="ElementNameFilter" type="s:string"/>
        <s:attribute name="VendorFilter" type="s:string"/>
        <s:attribute default="false"
            name="ReturnAllProperties" type="s:boolean"/>
        <s:attribute default="false"
            name="ReturnPropertyValues" type="s:boolean"/>
        <s:attribute default="false" name="ReturnErrorText" type="s:boolean"/>
    </s:complexType>
</s:element>
<s:simpleType name="browseFilter">
<s:complexType name="BrowseElement">
    <s:sequence>
        <s:element maxOccurs="unbounded" minOccurs="0"
            name="Properties" type="s0:ItemProperty"/>
    </s:sequence>
    <s:attribute name="Name" type="s:string"/>
    <s:attribute name="ItemPath" type="s:string"/>
    <s:attribute name="ItemName"
        sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1210328050.owl#NameOfMeasurement" type="s:string"/>
    <s:attribute name="IsItem" type="s:boolean" use="required"/>
    <s:attribute name="HasChildren" type="s:boolean" use="required"/>
```

Figure 28 SAWSDL modelReference annotations done in OPC-UA

The following annotations were done in WitsML:

The OWL class *MeasuredDepthCoord* was annotated to *mdHoleStart*, *mdHoleEnd*, *mdBitStart* and *mdBitEnd*.

```
<xsd:element maxOccurs="1" minOccurs="0" name="mdHoleStart"
    sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1205489371.owl#MeasureDepthCoord" type="witsml:measuredDepthCoord">
    <xsd:annotation>
        <xsd:documentation>Measured Depth at start of activity.  </xsd:documentation>
    </xsd:annotation>
</xsd:element>
<xsd:element maxOccurs="1" minOccurs="0" name="tvdHoleStart" type="witsml:wellVerticalDepthCoord">
    <xsd:annotation>
        <xsd:documentation>True Vertical Depth at start of activity  </xsd:documentation>
    </xsd:annotation>
</xsd:element>
<xsd:element maxOccurs="1" minOccurs="0" name="mdHoleEnd"
    sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1205489371.owl#MeasureDepthCoord" type="witsml:measuredDepthCoord">
    <xsd:annotation>
        <xsd:documentation>Measured Depth at end of activity.  </xsd:documentation>
    </xsd:annotation>
</xsd:element>
<xsd:element maxOccurs="1" minOccurs="0" name="tvdHoleEnd" type="witsml:wellVerticalDepthCoord">
    <xsd:annotation>
        <xsd:documentation>True Vertical Depth at end of activity.  </xsd:documentation>
    </xsd:annotation>
</xsd:element>
<xsd:element maxOccurs="1" minOccurs="0" name="mdBitStart"
    sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1205489371.owl#MeasureDepthCoord" type="witsml:measuredDepthCoord">
    <xsd:annotation>
        <xsd:documentation>Measured depth of bit at start of activity.  </xsd:documentation>
    </xsd:annotation>
</xsd:element>
<xsd:element maxOccurs="1" minOccurs="0" name="mdBitEnd"
    sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1205489371.owl#MeasureDepthCoord" type="witsml:measuredDepthCoord">
    <xsd:annotation>
        <xsd:documentation>Measured depth of bit at end of activity.  </xsd:documentation>
    </xsd:annotation>
</xsd:element>
```

Figure 29 SAWSDL modelReference annotations done in WitsML

In this case both WSDL documents are annotated with concepts from the same semantic model DrillReportSurvey see section *DrillReportSurvey* - Ontology used. The ontology contains the relationship between the concepts MeasuredDepthCoord and NameOfMeasurement. A semantic engine can use this relationship during Web service interface matching by parsing and reasoning over this semantic model. Therefore, the WSDL elements 'ItemName' in the OPC UA WSDL and the 'measuredDepthCoord' in the WitsML service WSDL match with one another. In addition the agent coordinating the reasoning needs to select only the Browse elements by the name of ‘MDEPTH’ for further integration.

Case 2: Matching Web Service by Ontology Mediation

In the previous section we made the assumption that we had a shared ontology between the WitsML and the OPC-UA service providers. This assumption may not always be true. Different vocabulary would most likely result in two different ontologies one for each side even if they belong to the same domain.

In such a case one can create a mapping ontology by capturing the relationships between the concepts used in the different ontologies. When a mapping ontology is available, then the semantic annotations extracted from the two services’ WSDL can be matched by using such a mapping ontology.

The following ontology (see Figure 30 The measurement ontology used by OPC UA web service) describes a MEASUREMENT that has a DEPTH in is linked to a DEVICE. The OPC-UA web service could be annotated to this ontology.

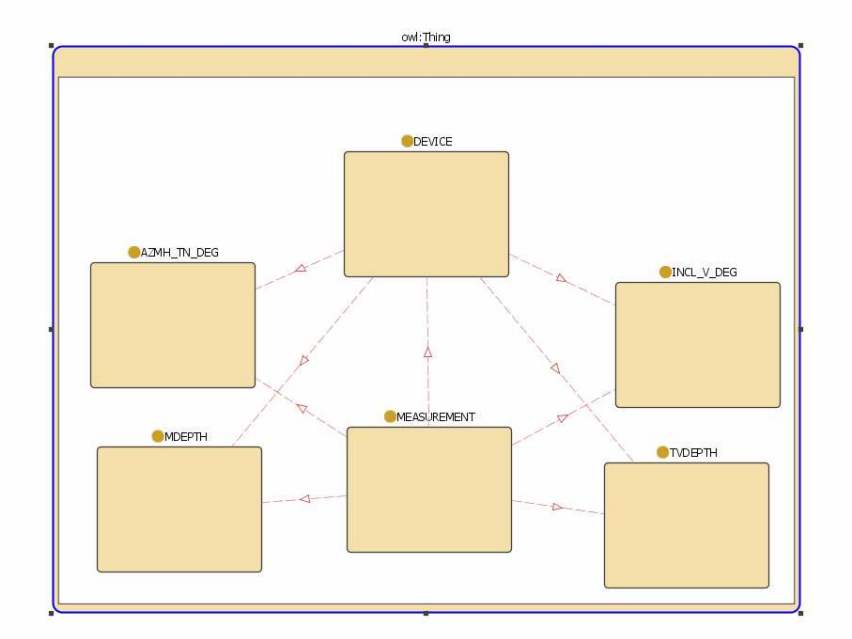


Figure 30 The measurement ontology used by OPC UA web service

The measurement ontology class may be annotated to the item name of the Read Response element in the OPC-UA wsdl, while the depth value itself may be linked to the Item value of the ReadResponse web service.

```
<s:complexType name="ItemValue">
    <s:sequence>
        <s:element maxOccurs="1" minOccurs="0"
            name="DiagnosticInfo" type="s:string"/>
        <s:element maxOccurs="1" minOccurs="0" name="Value"
        sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1210884790.owl#TVDEPTH http://www.owl-ontologies.com/Ontology1210884790.owl#MDEPTH
        <s:element maxOccurs="1" minOccurs="0"
            name="Quality" type="s0:OPCQuality"/>
    </s:sequence>
    <s:attribute name="ValueTypeQualifier" type="s:QName"/>
    <s:attribute name="ItemPath" type="s:string"/>
    <s:attribute name="ItemName"
        sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1210884790.owl#MEASUREMENT" type="s:string"/>
    <s:attribute name="ClientItemHandle" type="s:string"/>
    <s:attribute name="Timestamp" type="s:dateTime"/>
    <s:attribute name="ResultID" type="s:QName"/>
</s:complexType>
<s:complexType name="OPCQuality">
```

Figure 31. Depth measurement annotation in OPC-UA

The WitsML service has its own ontology, the SurveyStation ontology.

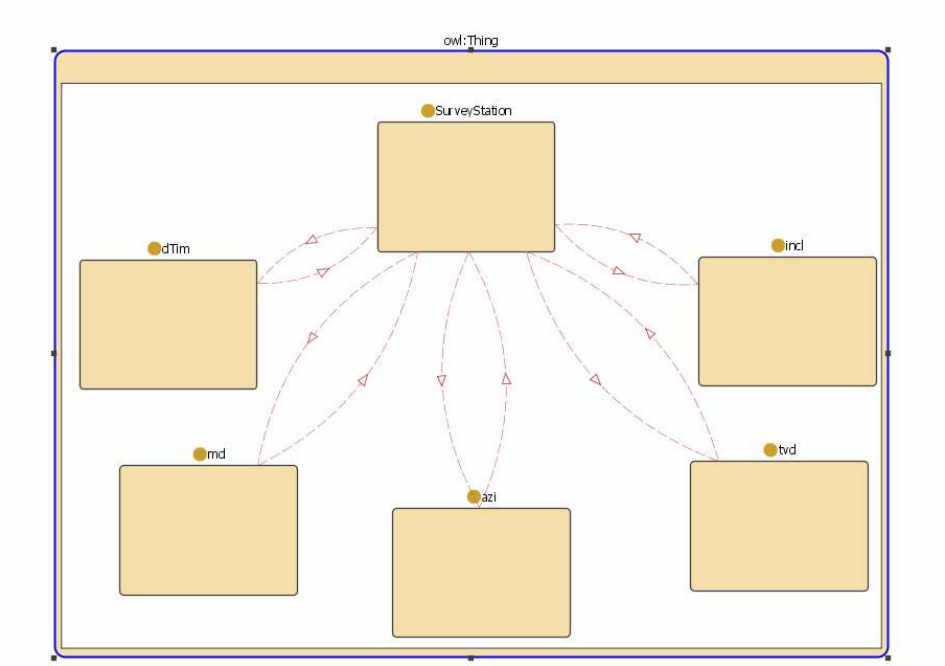


Figure 32. The SurveyStation ontology used by the DDR WITSML web service

The OWL classes SurveyStation, dTim, md, tvd, azi and incl are annotated to the respective elements in the WitsML schema.

```
<xsd:complexType name="cs_drillReportSurveyStation" sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1211490638.owl#SurveyStation">
    <xsd:annotation>
        <xsd:documentation>WITSML - Trajectory Station Component Schema</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
        <xsd:element maxOccurs="1" minOccurs="1" name="dTim"
            sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1211490638.owl#dTim" type="witsml:timestamp">
            <xsd:annotation>
                <xsd:documentation>The date at which the directional survey took place.</xsd:documentation>
                <xsd:appinfo source="http://projects.dnv.com/reference_data/RD4Browser/default.aspx?RDLDesignation=DATE OF SURVEY OPERATION"/>
            </xsd:annotation>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="1" name="md"
            sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1211490638.owl#md" type="witsml:measuredDepthCoord">
            <xsd:annotation>
                <xsd:documentation>Measured depth of measurement from the drill datum.</xsd:documentation>
                <xsd:appinfo source="http://projects.dnv.com/reference_data/RD4Browser/default.aspx?RDLDesignation=MEASURED DEPTH TO DIRECTIONAL SURVEY POINT RI
            </xsd:annotation>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="0" name="tvd"
            sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1211490638.owl#tvd" type="witsml:wellVerticalDepthCoord">
            <xsd:annotation>
                <xsd:documentation>Vertical depth of the measurements.</xsd:documentation>
                <xsd:appinfo source="http://projects.dnv.com/reference_data/RD4Browser/default.aspx?RDLDesignation=TRUE VERTICAL DEPTH FROM RKB TO SURVEY POINT
            </xsd:annotation>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="1" name="incl"
            sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1211490638.owl#incl" type="witsml:planeAngleMeasure">
            <!--PROFILE: Changed minOccurs from 0 to 1 for element incl.-->
            <xsd:annotation>
                <xsd:documentation>Hole inclination, measured from vertical.</xsd:documentation>
                <xsd:appinfo source="http://projects.dnv.com/reference_data/RD4Browser/default.aspx?RDLDesignation=MEASURED INCLINATION OF WELLBORE AT SURVEY P
            </xsd:annotation>
        </xsd:element>
        <xsd:element maxOccurs="1" minOccurs="1" name="azi"
            sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1211490638.owl#azi" type="witsml:planeAngleMeasure">
            <!--PROFILE: Changed minOccurs from 0 to 1 for element azi.-->
            <xsd:annotation>
                <xsd:documentation>Hole azimuth. Corrected to wells azimuth reference.</xsd:documentation>
                <xsd:appinfo source="http://projects.dnv.com/reference_data/RD4Browser/default.aspx?RDLDesignation=MEASURED AZIMUTH OF WELLBORE AT SURVEY POINT
            </xsd:annotation>
        </xsd:element>
```

Figure 33. The WitsML annotations to the SurveyStation ontology

Now, we can import these two Ontologies into a third new ontology, the MappingOntology. This ontology may contain defined relationships between the classes that span over the imported Ontologies.

<http://org3.example.com/ontologies/MappingOntology#> rdf:type owl:Ontology .

measurement:MDEPTH owl:equivalentClass SurveyStation:md .

As may be seen from this sample N3 code the DEPTH and md classes are equivalents. In this way the mapping ontology may provide the links between the web services elements having their own specialized ontology.

Case 3: Composing Web Services using ontology reasoning

In this section we will illustrate how semantic annotations can be used to compose Web services. The web services considered are the same discussed in the previous cases.

The scenario is as follows:

1. The user needs the current readings of the survey station of a given well. He inputs the wellbore to a virtual *getCurrentSurveyStation* web service. This is a composed web service.
2. A matching engine will be able to discover the appropriate OPC UA web services that report the measured metering for this well.
3. The OPC Browse service is utilized to find all Item Tags.
4. The matching engine will select the relevant Items and initiate an OPC read for each of them.
5. The value of the meter is contained within the *ItemValue* element contained in the *ReadResponse* element (see Figure 31. Depth measurement annotation in OPC-UA).
6. A reasoner will utilize the mapping ontology (see mapping below) and see that an ItemValue element equals the md, tvd, azi and incl classes of the WitsML ontology.
7. The matching engine will discover and utilize *createDDR* service which will generate the final result.

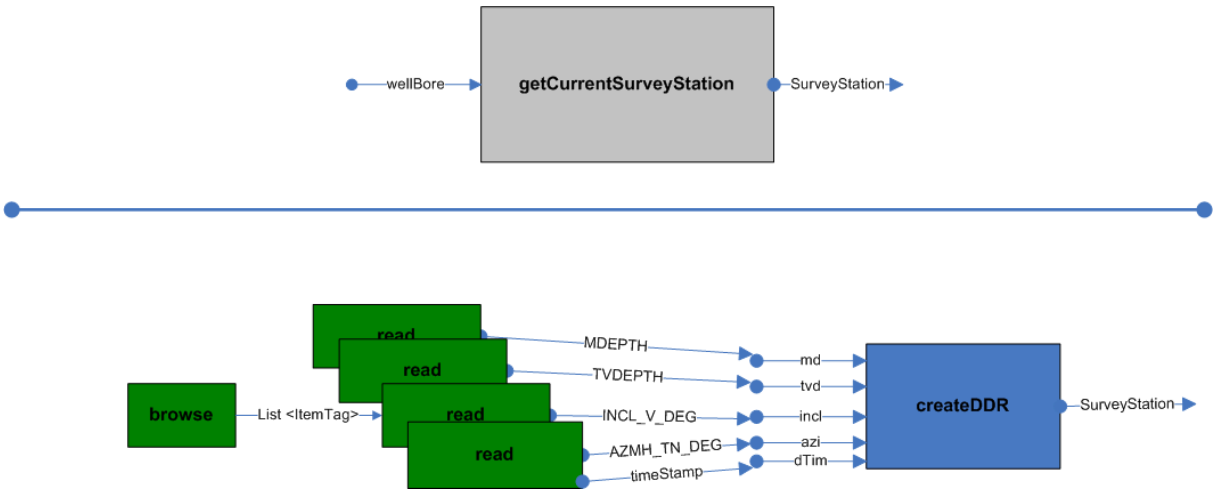


Figure 34 The composed service (top) is actually made up of several web services (bottom)

The semantic annotations of the web services is as in the previous section (see Case 2: Matching Web Service by Ontology Mediation) as the createDDR service is WitsML.

The mapping ontology will contain the following definitions:

<http://org3.example.com/ontologies/MappingOntology#> rdf:type owl:Ontology .

witsml:SurveyStation rdfs:subClassOf measurement:MEASUREMENT.
measurement:MDEPTH owl:equivalentClass witsml:md .

measurement:TVDEPTH owl:equivalentClass witsml:tvd .
measurement:AZMH_TN_DEG owl:equivalentClass witsml:azi .
measurement:INCL_V_DEG owl:equivalentClass witsml:incl .

These mappings will enable the reasoner utilized by the matching engine to work out the relationships between the OPC-UA elements and the WitsML elements.

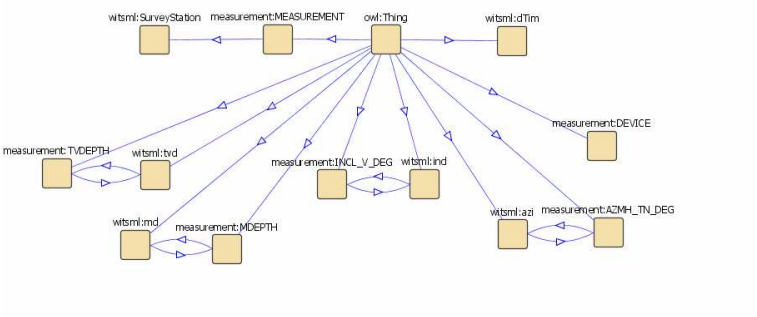


Figure 35. The mapping ontology

SAWSDL allows multiple annotations to be associated with those WSDL elements that can have a model reference extension. These annotations could be pointing to different concepts from the same semantic model or from different models altogether. For example, a WSDL element with the name "Value" can be associated with #TVDEPTH, #MDEPTH, #INCL_V_DEG and #AZMH_TN_DEG concepts all together.

```
<s:element maxOccurs="1" minOccurs="0" name="Value"
sawsdl:modelReference="http://www.owl-
ontologies.com/Ontology1210884790.owl#TVDEPTH http://www.owl-
ontologies.com/Ontology1210884790.owl#MDEPTH http://www.owl-
ontologies.com/Ontology1210884790.owl#INCL_V_DEG http://www.owl-
ontologies.com/Ontology1210884790.owl#AZMH_TN_DEG"/>
```

SAWSDL does not specify any relationship between these multiple annotations other than saying that they all apply. It is up to the consumers of these annotated WSDLs to use the ones that are relevant to them or to figure out the relationship between the concepts, if they so choose, by consulting the ontology that defines them.

The modelReference annotations may also belong to different Ontologies. If a requesting WSDL is referencing to multiple ontologies, this increases the likelihood of matching this request with other service WSDLs. Similarly, if a service WSDL is annotated with multiple concepts, possibly more request WSDLs will match it.

SAWSDL allows annotations to be added both at the complex type level and at the member element level. In cases where both complex type and the member elements have annotations, SAWSDL does not specify any relationship between the modelReferences on a complex type and those on the elements contained within a complex type. As SAWSDL has this flexibility, this might lead to inconsistencies within reasoning, due to conflicting properties between the modelReferences. It is up to the agent using these annotations to check the validity of this.

The SAWSDL modelReference attribute may also reference rules that set constraints on how to invoke services. For example, in the case 3 scenario, if we were to add the following "inputRule" constraints related to doing a request, they can be represented in SAWSDL using modelReferences as shown in the following wsdl part:

```
<wsdl:interface name="CreateDDRRequestService">
```

```
  <wsdl:operation name="CheckAvailabilityRequestOperation" pattern="http://www.w3.org/ns/wsdl/in-out">
    <wsdl:input element="CreateDDRRequestServiceRequest"
      sawsdl:modelReference="http://org1.example.com/rules#inputRule"/>
    <wsdl:output element="CreateDDRRequestServiceResponse"
      sawsdl:modelReference="http://org1.example.com/rules#outputRule1
              http://org1.example.com/rules#outputRule2
              http://org1.example.com/rules#outputRule3"/>
  </wsdl:operation>
</wsdl:interface>
```

This thesis does not consider rules as this is not a part of the SAWSDL specification as such. But to exemplify this, a rule may state that azimuth values greater than 360 are illegal. SWRL (Horrocks, Patel-Schneider et al. 2004) is a language for defining rules.

See “Appendix D Example of Rule syntax.“ for an example of rule syntax.

Defining Schema Mappings to Enable Web Service Invocation

The OPC-DA XML structure is generic in a way that all measurement readings are contained in a common complex type called *ItemValue*, See “Figure 31. Depth measurement annotation in OPC-UA”.

In the WitsML XML elements on the other hand there are specific placeholders for every type of measurements, see “Figure 33. The WitsML annotations to the SurveyStation ontology”
In this case, the contents of the *ItemName* and *Value* attributes of the ItemValue complex type needs to be transformed to the corresponding attributes of the complex type element *SurveyStation* when the matching agent invokes the Web service of the WitsML service provider.

To facilitate the association of such types of data transformations with Web services, SAWSDL provides a mechanism called Schema mapping. A Schema mapping allows the specification of transformation functions on the WSDL elements to map instance data defined by that XML schema document to the semantic data of the concepts in a semantic model. These transformation functions may be referenced by using the extension attribute *liftingSchemaMapping*.

It also allows the specification of transformation functions that transform data the opposite direction, from the semantic data of ontological concepts to the instance data values that adhere to the XML schema document that is being annotated. These transformation functions is referenced by the use of an extension attributes called *loweringSchemaMapping*.

These kinds of mappings are useful in general when the structure of the XML instance data does not correspond directly to the organization of the semantic data. Also, these types of mappings can be used to generate mediation code to support invocation of a Web services.

These lifting and lowering schema mappings are tested in the next two test cases within the context of the SurveyStation scenario. In the first case, see “Case 4: Lifting Schema Mapping”, how to do transformations between the *ItemValue* complex type element of OPC DA XML and the *MappingOntology* concepts is tested. In the second case, see “
A sample of such a transformation in XSLT is shown below.

```
<!DOCTYPE rdf:RDF
    [<!ENTITY xs "http://www.w3.org/2001/XMLSchema#">
    ]
```

```
>

<xsl:transform version="2.0"
    xmlns:ns="http://opcfoundation.org/webservices/XMLDA/1.0/"
    xmlns:measurement="http://www.owl-ontologies.com/Ontology1210884790.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:owl="http://www.w3.org/2002/07/owl#">

  <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes" />
  <xsl:template match="/ns:ReadResponse">
  <rdf:RDF>
      <owl:Ontology/>
      <measurement:MEASUREMENT>
        <measurement:has>
           <measurement:MDEPTH>
                <xsl:value-of select="ns:RItemList/ns:Items[@ItemName='MDEPTH']/ns:Value"/>
           </measurement:MDEPTH>
        </measurement:has>
     </measurement:MEASUREMENT>
     <measurement:MEASUREMENT>
        <measurement:has>
           <measurement:TVDEPTH>
                <xsl:value-of select="ns:RItemList/ns:Items[@ItemName='TVDEPTH']/ns:Value"/>
           </measurement:TVDEPTH>
        </measurement:has>
     </measurement:MEASUREMENT>
     <measurement:MEASUREMENT>
        <measurement:has>
           <measurement:AZMH_TN_DEG>
             <xsl:value-of select="ns:RItemList/ns:Items[@ItemName='AZMH_TN_DEG']/ns:Value"/>
           </measurement:AZMH_TN_DEG>
        </measurement:has>
     </measurement:MEASUREMENT>
     <measurement:MEASUREMENT>
        <measurement:has>
            <measurement:INCL_V_DEG>
                <xsl:value-of select="ns:RItemList/ns:Items[@ItemName='INCL_V_DEG']/ns:Value"/>
            </measurement:INCL_V_DEG>
        </measurement:has>
     </measurement:MEASUREMENT>
  <xsl:apply-templates select="ns:ReadResponse" />
</rdf:RDF>
</xsl:template>
</xsl:transform>
```

The following semantic data is obtained by applying this XSLT to the ReadResponse xml returned by the OPC UA Read service.

```
<!DOCTYPE rdf:RDF[
    <!ENTITY xs  "http://www.w3.org/2001/XMLSchema#" >
]
>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns=" http://www.owl-ontologies.com/Ontology1210884790.owl#"
  xml:base=" http://www.owl-ontologies.com/Ontology1210884790.owl#">

  <owl:Ontology />
  <MEASUREMENT>
    <has>
      <MDEPTH>385.4</MDEPTH>
    </has>
  </MEASUREMENT>
  <MEASUREMENT>
    <has>
      <TVDEPTH>289.6</TVDEPTH>
    </has>
  </MEASUREMENT>
  <MEASUREMENT>
    <has>
```

```
      <AZMH_TN_DEG>230.7</AZMH_TN_DEG>
    </has>
  </MEASUREMENT>
  <MEASUREMENT>
    <has>
      <INCL_V_DEG>60.7</INCL_V_DEG>
    </has>
  </MEASUREMENT>

</rdf:RDF>
```

", how to do transformations between the *MappingOntology* and the *SurveyStation* complex type element of the WitsML standard is tested.

XSLT (Kay 2007) has been used as the mapping language. SAWSDL specification by itself does not prescribe any specific mapping language. Users can choose a mapping language of their choice. (Joel Farrel and Holger Lausen 2007)

Case 4: Lifting Schema Mapping

A liftingSchemaMapping takes as input XML data, in a format being specified by a XML schema, and produces semantic data, in a format specified by a semantic model. Let us consider the *SurveyStation* scenario. The *ItemValue* complex type definition in the wsdl[1] is as follows:

```
            <s:element name="ReadResponse">
                <s:complexType
sawsdl:liftingSchemaMapping="file:///C:/Programfiler/xslt/ReadResponse2Ont.xslt">
                    <s:sequence>
                        <s:element maxOccurs="1" minOccurs="0"
                            name="ReadResult" type="s0:ReplyBase"/>
                        <s:element maxOccurs="1" minOccurs="0"
                            name="RItemList" type="s0:ReplyItemList"/>
                        <s:element maxOccurs="unbounded" minOccurs="0"
                            name="Errors" type="s0:OPCError"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
```

We will specify a lifting schema mapping notion on the ItemValue complex type so that an XML instance of a ReadResponse complex type can be mapped with the semantic data in the MappingOntology owl[2]. As we can see the liftingSchemaMapping attribute extension is now annotated to the ReadResponse element. This reference now refers to the transformation routine, in this case an XSLT transformation.

Given a ReadResponse message, as listed below, which corresponds to the schema listed above, the XSLT will transform this to an instance that adheres to the MappingOntology definition.

```
<ReadResponse xmlns="http://opcfoundation.org/webservices/XMLDA/1.0/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd=http://www.w3.org/2001/XMLSchema xmlns:xsl="http://www.w3.org/1999/XSL/Transform" >
```

[1] A more comprehensive listing of the WSDL may be found in "Part of the OPC UA WSDL used in Case 4 and Case 5" shows samples of the OPC UA WSDL that has *Read*, *ItemValue* and *ReadResponse* complex types.
[2] see "Appendix C Ontology samples used in the tests."

```
<ReadResult RcvTime="2008-04-29T16:12:05.065672-04:00"
            ReplyTime="2008-04-29T16:12:05.065672-04:00" ClientRequestHandle="testRead1"
            RevisedLocaleID="en-us" ServerState="running"/>
   <RItemList>
      <Items ItemName="MDEPTH" ClientItemHandle="testRead2"
             Timestamp="2008-04-29T16:12:04.0842608-04:00">
         <Value xsi:type="xsd:float">385.4</Value>
         <Quality/>
      </Items>
      <Items ItemName="TVDEPTH" ClientItemHandle="testRead2"
             Timestamp="2008-05-24T14:19:47.3103392-04:00">
         <Value xsi:type="xsd:float">289.6</Value>
         <Quality/>
      </Items>
      <Items ItemName="INCL_V_DEG" ClientItemHandle="testRead2"
             Timestamp="2008-05-24T14:19:47.3103392-04:00">
         <Value xsi:type="xsd:float">60.7</Value>
         <Quality/>
      </Items>
      <Items ItemName="AZMH_TN_DEG" ClientItemHandle="testRead2"
             Timestamp="2008-05-24T14:19:47.3103392-04:00">
         <Value xsi:type="xsd:float">230.7</Value>
         <Quality/>
       </Items>
   </RItemList>
</ReadResponse>
```

A sample of such a transformation in XSLT is shown below.

```
<!DOCTYPE rdf:RDF
    [<!ENTITY xs "http://www.w3.org/2001/XMLSchema#">
    ]
>

<xsl:transform version="2.0"
    xmlns:ns="http://opcfoundation.org/webservices/XMLDA/1.0/"
    xmlns:measurement="http://www.owl-ontologies.com/Ontology1210884790.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:owl="http://www.w3.org/2002/07/owl#">

  <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes" />
  <xsl:template match="/ns:ReadResponse">
  <rdf:RDF>
      <owl:Ontology/>
      <measurement:MEASUREMENT>
        <measurement:has>
           <measurement:MDEPTH>
               <xsl:value-of select="ns:RItemList/ns:Items[@ItemName='MDEPTH']/ns:Value"/>
           </measurement:MDEPTH>
        </measurement:has>
     </measurement:MEASUREMENT>
     <measurement:MEASUREMENT>
        <measurement:has>
           <measurement:TVDEPTH>
               <xsl:value-of select="ns:RItemList/ns:Items[@ItemName='TVDEPTH']/ns:Value"/>
           </measurement:TVDEPTH>
        </measurement:has>
     </measurement:MEASUREMENT>
     <measurement:MEASUREMENT>
        <measurement:has>
           <measurement:AZMH_TN_DEG>
             <xsl:value-of select="ns:RItemList/ns:Items[@ItemName='AZMH_TN_DEG']/ns:Value"/>
           </measurement:AZMH_TN_DEG>
        </measurement:has>
     </measurement:MEASUREMENT>
     <measurement:MEASUREMENT>
        <measurement:has>
            <measurement:INCL_V_DEG>
               <xsl:value-of select="ns:RItemList/ns:Items[@ItemName='INCL_V_DEG']/ns:Value"/>
            </measurement:INCL_V_DEG>
        </measurement:has>
     </measurement:MEASUREMENT>
  <xsl:apply-templates select="ns:ReadResponse" />
```

```
</rdf:RDF>
</xsl:template>
</xsl:transform>
```

The following semantic data is obtained by applying this XSLT to the ReadResponse xml returned by the OPC UA Read service.

```
<!DOCTYPE rdf:RDF[
    <!ENTITY xs  "http://www.w3.org/2001/XMLSchema#" >
]
>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns=" http://www.owl-ontologies.com/Ontology1210884790.owl#"
  xml:base=" http://www.owl-ontologies.com/Ontology1210884790.owl#">

  <owl:Ontology />
  <MEASUREMENT>
    <has>
      <MDEPTH>385.4</MDEPTH>
    </has>
  </MEASUREMENT>
  <MEASUREMENT>
    <has>
      <TVDEPTH>289.6</TVDEPTH>
    </has>
  </MEASUREMENT>
  <MEASUREMENT>
    <has>
      <AZMH_TN_DEG>230.7</AZMH_TN_DEG>
    </has>
  </MEASUREMENT>
  <MEASUREMENT>
    <has>
      <INCL_V_DEG>60.7</INCL_V_DEG>
    </has>
  </MEASUREMENT>

</rdf:RDF>
```

Case 5: Lowering Schema mapping

Lowering schema mapping is used to transform RDF to XML. A lowering schema may be annotated to a web service operation in the same way we did for a lifting schema.

```
<s:element name="Read"
    sawsdl:loweringSchemaMapping="file:///C:/eclipse/workspace/XML-DA/Ont2ReadRequest.xsl">
  <s:complexType>
     <s:sequence>
        <s:element maxOccurs="1" minOccurs="0" name="Options" type="s0:RequestOptions"/>
        <s:element maxOccurs="1" minOccurs="0" name="ItemList" type="s0:ReadRequestItemList"/>
     </s:sequence>
  </s:complexType>
</s:element>
```

Once a liftingSchema is annotated to a web service operation, an XSLT transform can create a specific XML that can be used to represent semantic data of an OWL ontology. But when specifying a loweringSchema, we don't know how the semantic data is represented. There are many ways to represent semantic data in XML RDF.

One way to solve this is to use SPARQL (Feigenbaum, Clark et al. 2008). This is a language to query the semantic data. The result may be presented in a well defined XML format. This means that once we know the xml format of the SPARQL output, we can create an XSLT transform to get an XML that is accepted by the web service operation in question.

The SPARQL query that is utilized in the loweringSchema could look something like this:

```
<lowering>
  <sparqlQuery>
    PREFIX measurement: "file:///C:/Program Files/Protege_3.3.1/Measurements.owl#"
    SELECT ?MDEPTH ?TVDEPTH ?INCL_V_DEG ?AZMH_TN_DEG
    WHERE {
      ?measurement measurement:has ?mdepth .
        ?mdepth measurement:MDEPTH ?MDEPTH .
      ?measurement measurement:has ?tvdepth .
        ?tvdepth measurement:TVDEPTH ?TVDEPTH .
      ?measurement measurement:has ?azi .
        ?azi measurement:AZMH_TN_DEG ?AZMH_TN_DEG .
      ?measurement measurement:has ?incl .
        ?incl measurement:INCL_V_DEG ?INCL_V_DEG }
  </sparqlQuery>
```

This would generate the following XML when applied on the result from the liftingSchema Mapping transformation:

```
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="MDEPTH" />
    <variable name="TVDEPTH" />
    <variable name="AZMH_TV_DEG" />
    <variable name="INCL_V_DEG" />
  </head>

  <results>
    <result>
      <binding name="MDEPTH">
        <literal>385.4</literal>
      </binding>

      <binding name="TVDEPTH">
        <literal>289.6</literal>
      </binding>

      <binding name="AZMH_TV_DEG">
        <literal>230.7</literal>
      </binding>

      <binding name="INCL_V_DEG">
        <literal>60.7</literal>
      </binding>

    </result>
  </results>
</sparql>
```

As soon as an XML result is created as output from the SPARQL script another XML transform could produce the following result, which can be transmitted as a SOAP request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns="http://opcfoundation.org/webservices/XMLDA/1.0/">
   <soapenv:Header/>
   <soapenv:Body>
      <ns:Read>
        <ns:ItemList>
           <ns:Items ItemName="MDEPTH"/>
           <ns:Items ItemName="TVDEPTH"/>
           <ns:Items ItemName="AZMH_TV_DEG"/>
           <ns:Items ItemName="INCL_V_DEG"/>
```

```
        </ns:ItemList>
      </ns:Read>
   </soapenv:Body>
</soapenv:Envelope>
```

As we have seen, the lifting- and loweringSchema mapping annotations enable the transformation of data between Web services which is important in order to be able to invoking Web services.

Section 8: Results, Findings and Possibilities

Case 1: Matching Web Service interfaces using a shared ontology

It was possible to match the OPC DA XML *ItemName* element and the WitsML *mdCurrent* elements by annotating them to the DrillReportSurvey ontology classes *NameOfMeasurement* and *MeasuredDepthCoord* respectively.

However, the most importing findings were that a placeholder for the *ItemName*, *NameOfMeasurement*, had to be defined and so did the relations between this new class and the existing *MeasuredDepthCoord.* But, having defined this, it was easy to make out the relationship between the elements in the different web services.

This means that a semantic engine should be able to do reasoning, based on the annotations done on the web services and the modifications done to the DrillReportSurvey ontology.

In "*Appendix B Case 1: Matching Web Service Interfaces using a Shared Ontology – Sample code.*" a sample java program used to retrieve annotations from the wsdl is listed. This sample shows that we are able to extract the model reference, which in turn may be used to do reasoning as we can get hold of all relevant ontology entities.

Case 2: Matching Web Service by Ontology Mediation

The case 2 test showed us that it is possible to build relations between classes being defined in different Ontologies. These Ontologies have to be imported into a common ontology.

This fact is really exciting, as it means that the mapping may be defined outside the web service layer. This also means that the web services may run unaffected by any changes done to the mapping, whilst the reasoning will be changed as the relations are changed. If the reasoners are able to relate to the Ontologies when it comes to finding class relationships, these relationships may be changed in run time.

The following architecture could very well be implemented for our pilot.

An ontology is defined for each web service deployed in our scenario. This ontology is highly customized for the web service it is linked to. There are a number of tools that may be utilized to automatically generate such Ontologies. One example is the CMU wsdl2owl-s tool which generates an OWL-S ontology based on a web service's wsdl. (Shafiq, Moran et al. 2007)

SAWSDL is used to annotate the web service to the web service tailored ontology. This annotation should be fairly straight forward as the elements in the wsdl and the classes in the

ontology should have a 1:1 mapping. Also this annotation should be fairly static as there will be no need to change the ontology if the web service remains unchanged.

Secondly a Mapping ontology describing our area of interest could be defined. In our pilot this would typically be the DrillReportSurvey ontology.

This ontology may in turn import all the related web service ontologies and even the ISO-15926-4 ontology. Now we can start defining relationships between the classes of the different Ontologies. The following source being an example of such mappings.

DrillReportSurvey:DEPTH owl:equivalentClass ontWitsMl:measuredDepth.
ontWitsMl:openHoleCasing RDF:subClassOf DrillReportSurvey :DEVICE
ontWitsMl:measuredDepth RDF:subClassOf ontOpcUa:ItemValue
ontWitsMl:measuredDepth RDF:subClassOf ISO15926-4:MeasureDepthCoord

Finally, we are able to implement reasoning agents that only have to relate to our Mapping ontology. Reasoning agents may be developed by utilizing Jena. Jena is a Java framework for writing Semantic Web applications developed by HP Laboratories in Bristol and donated to the Open Source community. This framework implements a lot of features, among other methods for communicating with reasoner tools. (HP Laboratories 2001-2008)

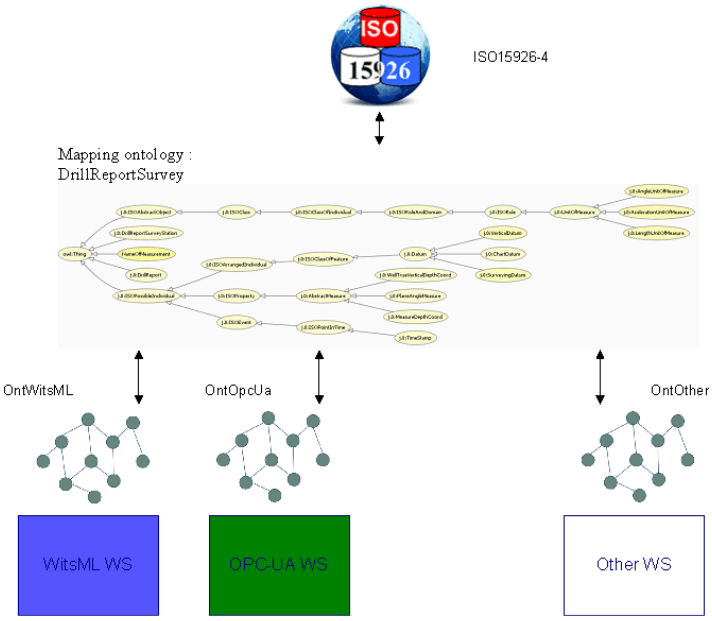


Figure 36 Web Service matching by doing Ontology Mediation

The main benefit by implementing architectures like this is that modifications only need to be done in the mapping ontology. There is no need in doing changes to application code, avoiding system shutdowns, costly and resource consuming programming, testing and deployments. Changes may be executed during runtime.

Case 3: Composing Web Services using Ontology Reasoning

In "Case 3: Composing Web Services using ontology reasoning" we saw that a matching engine can utilize Ontologies to do web service composition or web service choreography and orchestration. By this we mean that this engine will have the ability to

- Discover relevant web services. How to do this is discussed below.

- Orchestrate web services, which means to deploy the appropriate services in the right sequence with appropriate input.
- Do reasoning by utilizing a mapping ontology as showed in "Case 2: Matching Web Service by Ontology Mediation" and by utilizing a reasoning agent. In this thesis RacerPro (KG 2004) was used.

The web service discovery may very well be utilized by defining relations between a wellbore instance and an instance in an ontology designed for that particular web service. This relationship will be discovered by the reasoner and may be used by the engine to invoke the service.

Another way of invoking the service is by annotating a wellbore ontology instance to an element in a published web service. By adding more annotations to this web service it should be fairly easy to discover relevant web services.

```
<s:element name="Read" sawsdl:modelReference="http://www.owl-ontologies.com/Ontology1211527986.owl#A-1">
    <s:complexType>
        <s:sequence>
            <s:element maxOccurs="1" minOccurs="0"
                name="Options" type="s0:RequestOptions"/>
            <s:element maxOccurs="1" minOccurs="0"
                name="ItemList" type="s0:ReadRequestItemList"/>
        </s:sequence>
    </s:complexType>
</s:element>
```

Figure 37 Annotation of a wellbore instance to the Read request element in OPC UA

In either way the SAWSDL specification will provide useful functionality by means of annotating web services to Ontologies. But the matching engine still needs to be developed.

Case 4: Lifting Schema Mapping

In order to be able to transform the output from a web service response into a RDF/XML that specifies an ontology a transformation script needs to be created. This script may be referenced in the SAWSDL defined liftingSchemaMapping attribute that may be included in an element definition of the web service description file.

Case 5: Lowering Schema mapping

In order to be able to transform an ontology that is defined in any format into a XML that may be used as input to invoke a web service a transformation script needs to be created. This script may be referenced in the SAWSDL defined loweringSchemaMapping attribute that may be included in an element definition of the web service description file.

As the ontology format may vary, the SPARQL querying language needs to be utilized to transform this format into a well known format in order to achieve an input that may be accepted by the web service that is to be invoked.

General findings

SAWSDL is merely a specification. In fact it is quit small and does only specify how and where these 3 new attributes; modelReference, liftingSchemaMapping and loweringSchemaMapping, should be placed on WSDL Elements and XML Schema Elements.

Any parser (or other application) must support SAWSDL to get any use out of it. Even then parsers don't get anything but the URIs for modelReferences or lifting and lowering mappings.

This was probably done to avoid SAWSDL being tied to any particular semantic representation, e.g. RDF or OWL, or a particular transformation representation, e.g. XPath or XQuery.

So, if we want to deal with concepts or classes in ontologies we must load them with a library and then request the resource we're looking for with the URI supplied by the SAWSDL parser. SAWSDL4J (Gomadam, Brewer et al. 2007) are a Java API built to extract such annotations.

The Uri's for the schema mappings represents the actual locations of the documents that do the mapping. But SAWSDL as such doesn't offer any functionality to invoke these scripts.

Frameworks like Jena (HP Laboratories 2001-2008) or Sesame (openRDF.org 2007) are able to directly attach concepts from an ontology. Jena is a framework to build semantic applications. It offers a programmatic environment to deal with RDF, RDFS, OWL and SPARQL. It also includes a rule-based inference engine. This framework is probably a candidate to be used to build matching engines, mediation servers and do transformations.

Section 9: Summary

The World Wide Web as we know it today is a very large set of distributed documents. Even if the response is created instantly, this response is a document that needs the interpretation of a human.

The Semantic Web will utilize the current the '*web of documents'*. But by adding an additional layer of semantics it will turn the web into '*a web of data*'. The semantic technology offers functionality to automatically discover web services, combining them and on the fly offer a new web service that responds to any user requests.

Semantics is the study of meaning. Semantic models are models expressed in formal languages defining how concepts or classes relate to each other. These models enables a human, or even better a machine, to start searching for information in one source, and then move through a set of sources that are connected not by wires but by having some relations in the meaning of the subject.

In the Semantic Web data is represented by semantic models. There are two main W3C Standards that are used to define such models: Resource Description Framework, RDF and Web Ontology Language, OWL.

The Web Services Description Language (WSDL) specifies a way to describe the functionalities of a web service and how and where to invoke it. The WSDL specification does not include semantics in the description of Web services. Therefore, two services can have almost similar descriptions while meaning different things.

Resolving this ambiguity in Web services descriptions is big step toward automated discovery and composition of Web services, which will be a key productivity enabler in application integration.

Semantic Annotations for WSDL and XML Schema, SAWSDL, is a specification that defines how semantic concepts can be added to WSDL components. SAWSDL does not specify a language for representing the semantic models. Instead, it provides mechanisms by which concepts from the semantic models can be referenced from within WSDL components as annotations.

These semantics will remove any disambiguates in the description of Web services during discovery and composition of the Web services. It enables semantic annotations for Web services not only for the discovery but also for invoking them.

SAWSDL is an extension to the existing WSDL framework. It doesn't relate to any specific semantic representation language. SAWSDL is recommended by W3C and intends to close the gap between Web Services and the Semantic Web.

SAWSDL defines 3 extension attributes; modelReference, liftingSchemaMapping and loweringSchemaMapping. In this thesis they are all tested on a business case defined by the PCA SIG for Drilling and Completion, the SurveyStation business case.

The following test cases were defined and executed in this thesis:

In “Case 1: Matching Web Service Interfaces using a Shared Ontology” we found that it is possible to annotate classes of an ontology to elements of a web service defined in a wsdl. We also showed how to extract such annotations for further use in an agent of some sort.

In “Case 2: Matching Web Service by Ontology Mediation” we found that it is possible to lever the mapping and reasoning to the semantic layer of the architecture. This seems to be very valuable in the sense that all modifications as it comes to reasoning may be done without any need to change source code. This means that such modifications may be done in runtime.

In “Case 3: Composing Web Services using ontology reasoning” we found that it was possible to compose multiple web services into one by the use of a matching engine. This engine is able to discover, orchestrate and deploy web services to give the requested response. However, an assumption must be made, that the web services is correctly annotated to an appropriate ontology.

The SurveyStation scenario was used to demonstrate how to mediate between the OPC and the WitsML web services via a mapping ontology. This required the capability of transforming the representation of data between XML and semantic data represented as RDF/XML.

In “Case 4: Lifting Schema Mapping” we showed how we according to the SAWSDL specification was able, by using XSLT, to transform data represented as XML into RDF/XML.

In “Case 5: Lowering Schema mapping” we showed that we had to use SPARQL to extract data from the semantic model to ensure that the data was represented in XML RDF. In this

way we could transform the data to be represented in an XML format that was accepted by the web service.

Section 10: Conclusion and further work

The scope of this thesis was to evaluate how the SAWSDL specification could be utilized within the Oil & Gas industry. The task was initiated by the PCA SIG for Drilling and Completion and regarded as the first small step in defining how to set up an environment for deploying the semantic web within integrated operations.

SAWSDL, Semantic Annotated Web Service Definition Language, is merely a specification and offers no functionality to fulfill any of the requirements defined to enable the semantic web by itself. But this specification defines how to annotate concepts or classes defined in an ontology to elements defined in the web service description, the WSDL. In addition it specifies how to annotate transformation scripts to elements of the WSDL. These references are key enablers for linking the web services to semantic technologies used in the semantic web. Tools like Jena, Sesame, WSMO, Racer Pro, SAXON and many more may utilize these references to do discovery, contracting, negotiation, composing, mediation, invoking and monitoring.

There are a set of other initiatives that offers the ability to couple web services and the semantics. But these are usually requiring other tools that are proprietary and often only available commercially.

The SAWSDL extension to WSDL has become a recommendation (Joel Farrel and Holger Lausen 2007) by W3C. SAWSDL is not dependent on any other tools; in fact it is based solely on other W3C recommended specifications. W3C being a non-profit organization (Jacobs 2007) is dedicated to work for the benefits of the world wide web.

The thesis therefore concludes by stating that it highly recommends the use of SAWSDL as the preferred way of annotating semantics to web services.

But, additional work will be required to fulfill the task of developing a pilot for Oil & Gas fulfilling all capabilities that a semantic web service should provide.

Further work

In the following this thesis suggests 4 follow up tasks that will provide contributions to a full blown pilot that will demonstrate all the capabilities that the SAWSDL specification provides annotations for.

Implement an ISO-15926-7 plug-in for Protégé.

ISO 15926 is an International Standard for the representation of lifecycle information for process plants, including oil and gas production facility. This is specified by a generic, conceptual data model that is suitable as the basis for implementation in a shared database or data warehouse.

ISO 15926-4 Reference Data - defines the initial set of standard reference data for oil and gas production facilities. It defines standard data model terminology. It is a managed

collection of process plant lifecycle data classes which are common to many process plants or of interest to many users.

ISO-15926-7 Implementation in OWL+RDF - specifies the methods by which part 2 and part 4 can be implemented using Semantic Web technologies defined by W3C.

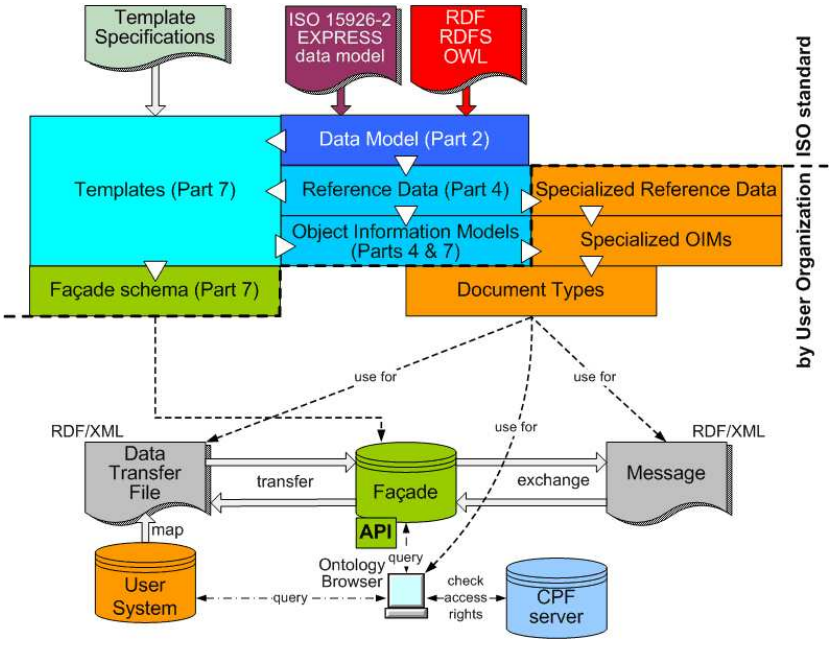


Figure 38 ISO-15926-7 Overview (Teijgeler 2007)

However, ISO-15926 is so huge that deploying the hole model at runtime is both resource consuming and difficult to handle. A candidate should look into the possibility of making snapshots of the model. Such snapshots could be any area of interest of the model that is needed to solve a particular use case. The use case used in this thesis could be a candidate for a pilot.

In addition, a Protégé plug-in, would be very useful for viewing, editing and even do reasoning over. Racer Pro is a reasoning engine that is already integrated into Protégé.

This functionality would be very useful in implementing test cases 1 trough 3.

Develop a transformation processing engine.

To implement the test cases 4 and 5, a transformation engine that does XSLT transformations or transformations between XML and RDF/XML needs to be implemented. SAXON (Kay 2008) may be used to develop such an engine.

The Saxon package is a collection of tools for processing XML documents. The main components are an XSLT 2.0 processor that can be used to run XSLT 1.0 stylesheets. It has support for XPath 2.0, XQuery and XML Schema. In addition it offers a java interface, which means that is can be used directly in java programs.

Develop a matching engine.

To fulfill the tasks designed in Cases 1-3 a matching engine should be implemented. This engine must be capable of doing discovery, negotiation, filtering, choreography, orchestration

and reasoning. The Jena framework (HP Laboratories 2001-2008) developed by HP Laboratories could very well be a candidate for facilitating this.

Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine.

The Jena Framework includes:

- A RDF API
- Reading and writing RDF in RDF/XML, N3 and N-Triples
- An OWL API
- In-memory and persistent storage
- SPARQL query engine

Detection of critical events in IO Oil & Gas by using semantic rule engines.

As described in section "the increase of real-time data actually decreases the ability to do decisions. The semantic web technology should provide means of tracking events and decide the criticality.

Rules could be defined in a way that combinations of data values trigger alarms at different levels of criticality. RuleML (Boley 2006) or SWRL are languages that define such rules. (Horrocks, Patel-Schneider et al. 2004)

In this way the operator may be presented a list of the most critical events in one list. All less important events will be moved towards the end of the list and the operators may pick tasks from the top of the list.

This rules defined in either RuleML or SWRL may be annotated to elements of a web service.

Appendix A Abbreviations

ABox	Assertion Box
AI	Artificial Intelligence
BPEL	Business Process Execution Language
BPM	Business Process Modelling
DDR	Daily Drilling Report
DL	Description Logics
EMF	Eclipse Modelling Framework
ESB	Enterprise Service Bus
HTML	HyperText Markup Language
IDE	Integrated Development Environment
MDA	Model-Driven Architecture
MOM	Message Oriented Middleware
ODM	Ontology Definition Metamodel
OLF	Oljeindustriens Lands Forening – The Norwegian Oil Industry Association
OMG	Object Management Group
OPC UA	OPen Connectivity Unified Architecture
OPC XMLDA	OPC XML Data Access
OWL	Web Ontology Language
OWL-S	Web Ontology Language for Services
PCA	POSC Caesar Association
PSL	Process Specification Language
RDF	Resource Description Framework
RDF-S	Resource Description Framework – Schema
RDL	Reference Data Library
RDS	Reference Data System
RIF	Rules Interchange Format
RuleML	Rule Markup Language
SAWSDL	Semantic Annotations for WSDL
SOA	Service-Oriented Architecture
SPARQL	SPARQL Protocol And RDF Query Language
SQL	Structured Query Language
SWRL	Semantic Web Rules Language
SWSF	Semantic Web Services Framework
SWSL	Semantic Web Services Language
TBox	Terminology Box
UML	Unified Modelling Language
W3C	World Wide Web Consortium
WITSML	Well site Information Transfer Standard Markup Language
WSDL	Web Service Definition Language
WSMO	Web Service Modelling Ontology
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language Transformation

Appendix B WSDL

Part of the OPC UA WSDL used in Case 4 and Case 5

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
    targetNamespace="http://opcfoundation.org/webservices/XMLDA/1.0/"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://opcfoundation.org/webservices/XMLDA/1.0/"
    xmlns:sawsdl="http://www.w3.org/ns/sawsdl"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:wsdl="http://www.w3.org/ns/wsdl">
    <types>
<s:schema elementFormDefault="qualified" targetNamespace="http://opcfoundation.org/webservices/XMLDA/1.0/">
          <s:element name="Read">
                <s:complexType>
                    <s:sequence>
                        <s:element maxOccurs="1" minOccurs="0"
                            name="Options" type="s0:RequestOptions"/>
                        <s:element maxOccurs="1" minOccurs="0"
                            name="ItemList" type="s0:ReadRequestItemList"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="RequestOptions">
                <s:attribute default="true" name="ReturnErrorText" type="s:boolean"/>
                <s:attribute default="false" name="ReturnDiagnosticInfo" type="s:boolean"/>
                <s:attribute default="false" name="ReturnItemTime" type="s:boolean"/>
                <s:attribute default="false" name="ReturnItemPath" type="s:boolean"/>
                <s:attribute default="false" name="ReturnItemName" type="s:boolean"/>
                <s:attribute name="RequestDeadline" type="s:dateTime"/>
                <s:attribute name="ClientRequestHandle" type="s:string"/>
                <s:attribute name="LocaleID" type="s:string"/>
            </s:complexType>
            <s:complexType name="ReadRequestItemList">
                <s:sequence>
                    <s:element maxOccurs="unbounded" minOccurs="0"
                        name="Items" type="s0:ReadRequestItem"/>
                </s:sequence>
                <s:attribute name="ItemPath" type="s:string"/>
                <s:attribute name="ReqType" type="s:QName"/>
                <s:attribute name="MaxAge" type="s:int"/>
            </s:complexType>
            <s:complexType name="ReadRequestItem">
                <s:attribute name="ItemPath" type="s:string"/>
                <s:attribute name="ReqType" type="s:QName"/>
                <s:attribute name="ItemName" type="s:string"/>
                <s:attribute name="ClientItemHandle" type="s:string"/>
                <s:attribute name="MaxAge" type="s:int"/>
            </s:complexType>
            <s:element name="ReadResponse">
                <s:complexType      sawsdl:liftingSchemaMapping="file:///C:/Programfiler/xslt/ReadResponse2Ont.xslt">
                    <s:sequence>
                        <s:element maxOccurs="1" minOccurs="0"
                            name="ReadResult" type="s0:ReplyBase"/>
                        <s:element maxOccurs="1" minOccurs="0"
                            name="RItemList" type="s0:ReplyItemList"/>
                        <s:element maxOccurs="unbounded" minOccurs="0"
                            name="Errors" type="s0:OPCError"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="ReplyItemList">
                <s:sequence>
                    <s:element maxOccurs="unbounded" minOccurs="0"
                        name="Items" type="s0:ItemValue"/>
                </s:sequence>
                <s:attribute name="Reserved" type="s:string"/>
            </s:complexType>
            <s:complexType name="ItemValue">
                <s:sequence>
                    <s:element maxOccurs="1" minOccurs="0"
                        name="DiagnosticInfo" type="s:string"/>
                    <s:element maxOccurs="1" minOccurs="0" name="Value" sawsdl:modelReference="http://www.owl-
ontologies.com/Ontology1210884790.owl#TVDEPTH http://www.owl-ontologies.com/Ontology1210884790.owl#MDEPTH http://www.owl-
ontologies.com/Ontology1210884790.owl#INCL_V_DEG http://www.owl-ontologies.com/Ontology1210884790.owl#AZMH_TN_DEG"/>
                    <s:element maxOccurs="1" minOccurs="0"
                        name="Quality" type="s0:OPCQuality"/>
                </s:sequence>
                <s:attribute name="ValueTypeQualifier" type="s:QName"/>
                <s:attribute name="ItemPath" type="s:string"/>
                <s:attribute name="ItemName" sawsdl:modelReference="http://www.owl-
ontologies.com/Ontology1210884790.owl#MEASUREMENT" type="s:string"/>
                <s:attribute name="ClientItemHandle" type="s:string"/>
                <s:attribute name="Timestamp" type="s:dateTime"/>
                <s:attribute name="ResultID" type="s:QName"/>
            </s:complexType>
          </types>
    <message name="ReadSoapIn">
        <part element="s0:Read" name="parameters"/>
    </message>
    <message name="ReadSoapOut">
        <part element="s0:ReadResponse" name="parameters"/>
            </message>
    <portType name="Service">
```

```
    <operation name="Read">
        <input message="s0:ReadSoapIn"/>
        <output message="s0:ReadSoapOut"/>
    </operation>
    <binding name="Service" type="s0:Service">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="Read">
            <soap:operation
                soapAction="http://opcfoundation.org/webservices/XMLDA/1.0/Read" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
    </binding>
</definitions>
```

Appendix C Java Source code

Case 1: Matching Web Service Interfaces using a Shared Ontology – Sample code.

```
    public static void main(String[] args) throws Exception{
        if (args.length >1 || args.length <1){
            System.out.println("Only one argument is accepted - that should be the file name!");
            //return;
        }

        Description description = SAWSDLUtil.getDescription("file:///C:/eclipse/workspace/XML-DA/OpcDaGateway2.wsdl");

        // Find the modelReference specified for the first interface
        Interface[] interfaces = description.getInterfaces();
        ModelReference modelRef = SAWSDLUtil.getModelRef(interfaces[0]);

        // Locate the Browse element in the wsdl
        ElementDeclaration elDecl = description.getElementDeclaration(new QName("http://opcfoundation.org/webservices/XMLDA/1.0/", "Browse"));

        XmlSchemaElement element = (XmlSchemaElement) elDecl.getContent();
        XmlSchemaComplexType cT = (XmlSchemaComplexType) element.getSchemaType();

        // Get all the Attributes for the Browse element
        XmlSchemaObjectCollection attributes = cT.getAttributes();

        // Iterate over all Attributes for the Browse element to check if any
        // Meta information, that is any sawsdl references, is defined
        Iterator attribIterator = attributes.getIterator();
        while (attribIterator.hasNext()) {
            Object o = attribIterator.next();
            if (o instanceof XmlSchemaAttribute) {
                XmlSchemaAttribute attr = (XmlSchemaAttribute) o;
                if (attr.getMetaInfoMap() != null) {
                    // Meta information found, present it
                    System.out.println("Element ->" + element.getName());
                    System.out.println("MetaInfo :");
                    System.out.println(attr.getMetaInfoMap().toString());
                }
            }
        }
    }
}
```

Figure 39 Case 1 Java Sample Code

Output from executing this program:

```
Element ->Browse
MetaInfo :
{interface edu.uga.cs.lsdis.wsdl20.extensions.sawsdl.ModelReference=Mref=
      http://www.owl-
ontologies.com/Ontology1210328050.owl#NameOfMeasurement
}
```

Appendix C Ontology samples used in the tests.

The MEASUREMENT.owl

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://www.owl-ontologies.com/Ontology1210884790.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-ontologies.com/Ontology1210884790.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="INCL_V_DEG"/>
  <owl:Class rdf:ID="TVDEPTH"/>
  <owl:Class rdf:ID="MDEPTH"/>
  <owl:Class rdf:ID="DEVICE"/>
  <owl:Class rdf:ID="AZMH_TN_DEG"/>
  <owl:Class rdf:ID="MEASUREMENT"/>
  <owl:ObjectProperty rdf:ID="isPerformedOn">
    <rdfs:domain rdf:resource="#MEASUREMENT"/>
    <rdfs:range rdf:resource="#DEVICE"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="has">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#DEVICE"/>
          <owl:Class rdf:about="#MEASUREMENT"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#MDEPTH"/>
          <owl:Class rdf:about="#TVDEPTH"/>
          <owl:Class rdf:about="#INCL_V_DEG"/>
          <owl:Class rdf:about="#AZMH_TN_DEG"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.3.1, Build 430)  http://protege.stanford.edu -->
```

The DDR.owl

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.owl-ontologies.com/Ontology1211490638.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-ontologies.com/Ontology1211490638.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="SurveyStation"/>
  <owl:Class rdf:ID="azi"/>
  <owl:Class rdf:ID="tvd"/>
  <owl:Class rdf:ID="md"/>
  <owl:Class rdf:ID="incl"/>
  <owl:Class rdf:ID="dTim"/>
  <owl:ObjectProperty rdf:ID="includedIn">
    <owl:inverseOf>
      <owl:InverseFunctionalProperty rdf:ID="has"/>
    </owl:inverseOf>
    <rdfs:range rdf:resource="#SurveyStation"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#dTim"/>
          <owl:Class rdf:about="#md"/>
          <owl:Class rdf:about="#tvd"/>
```

```
          <owl:Class rdf:about="#incl"/>
          <owl:Class rdf:about="#azi"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:ObjectProperty>
  <owl:InverseFunctionalProperty rdf:about="#has">
    <owl:inverseOf rdf:resource="#includedIn"/>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#dTim"/>
          <owl:Class rdf:about="#md"/>
          <owl:Class rdf:about="#tvd"/>
          <owl:Class rdf:about="#incl"/>
          <owl:Class rdf:about="#azi"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
    <rdfs:domain rdf:resource="#SurveyStation"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
  </owl:InverseFunctionalProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.3.1, Build 430)  http://protege.stanford.edu -->
```

The MappingOntology.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.owl-ontologies.com/Ontology1211527986.owl#"
     xml:base="http://www.owl-ontologies.com/Ontology1211527986.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:protege="http://protege.stanford.edu/plugins/owl/protege#"
     xmlns:p1="http://www.owl-ontologies.com/assert.owl#"
     xmlns:witsml="http://www.owl-ontologies.com/Ontology1211490638.owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:measurement="http://www.owl-ontologies.com/Ontology1210884790.owl#">
    <owl:Ontology rdf:about="">
        <owl:imports rdf:resource="http://www.owl-ontologies.com/Ontology1211490638.owl"/>
        <owl:imports rdf:resource="http://www.owl-ontologies.com/Ontology1210884790.owl"/>
    </owl:Ontology>
    <Wellbore rdf:ID="A-1"/>
    <Wellbore rdf:ID="A-2"/>
    <Wellbore rdf:ID="B-1_ST2"/>
    <Wellbore rdf:ID="C-1"/>
    <owl:ObjectProperty rdf:ID="hasMeasurements">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
        <rdfs:domain rdf:resource="#Wellbore"/>
        <rdfs:range rdf:resource="http://www.owl-ontologies.com/Ontology1211490638.owl#SurveyStation"/>
        <owl:inverseOf rdf:resource="#measuredIn"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="hasURL">
        <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#InverseFunctionalProperty"/>
        <rdfs:domain rdf:resource="#WebService"/>
        <rdfs:range rdf:resource="#URL"/>
        <owl:inverseOf rdf:resource="#isURLOf"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="isURLOf">
        <rdfs:domain rdf:resource="#URL"/>
        <rdfs:range rdf:resource="#WebService"/>
        <owl:inverseOf rdf:resource="#hasURL"/>
    </owl:ObjectProperty>
    <owl:ObjectProperty rdf:ID="measuredIn">
        <rdfs:domain rdf:resource="http://www.owl-ontologies.com/Ontology1211490638.owl#SurveyStation"/>
        <rdfs:range rdf:resource="#Wellbore"/>
        <owl:inverseOf rdf:resource="#hasMeasurements"/>
    </owl:ObjectProperty>
    <rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1210884790.owl#AZMH_TN_DEG">
        <owl:equivalentClass rdf:resource="http://www.owl-ontologies.com/Ontology1211490638.owl#azi"/>
    </rdf:Description>
    <rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1210884790.owl#INCL_V_DEG">
        <owl:equivalentClass rdf:resource="http://www.owl-ontologies.com/Ontology1211490638.owl#incl"/>
    </rdf:Description>
    <rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1210884790.owl#MDEPTH">
        <owl:equivalentClass rdf:resource="http://www.owl-ontologies.com/Ontology1211490638.owl#md"/>
    </rdf:Description>
    <rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1210884790.owl#TVDEPTH">
        <owl:equivalentClass rdf:resource="http://www.owl-ontologies.com/Ontology1211490638.owl#tvd"/>
```

```
    </rdf:Description>
    <owl:ObjectProperty rdf:ID="objectProperty_15"/>
    <owl:ObjectProperty rdf:ID="objectProperty_16"/>
    <WebService rdf:ID="OPC-UA_A-1">
        <hasURL rdf:resource="#URL_12"/>
    </WebService>
    <owl:Class rdf:ID="URL"/>
    <URL rdf:ID="URL_12">
        <owl:equivalentProperty
rdf:datatype="http://www.w3.org/2001/XMLSchema#string"></owl:equivalentProperty>
        <rdfs:comment xml:lang="en"
            >http://www.advosol.com/t-SamplesXML.aspx</rdfs:comment>
        <isURLOf rdf:resource="#OPC-UA_A-1"/>
    </URL>
    <owl:Class rdf:ID="WebService"/>
    <owl:Class rdf:ID="Wellbore"/>
    <Wellbore rdf:ID="Wellbore_9"/>
    <rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1211490638.owl#dTim">
        <owl:equivalentClass rdf:resource="http://www.owl-
ontologies.com/Ontology1211490638.owl#SurveyStation"/>
    </rdf:Description>
    <rdf:Description rdf:about="http://www.owl-ontologies.com/Ontology1211490638.owl#SurveyStation">
        <owl:equivalentClass rdf:resource="http://www.owl-ontologies.com/Ontology1211490638.owl#dTim"/>
        <rdfs:subClassOf rdf:resource="http://www.owl-ontologies.com/Ontology1210884790.owl#MEASUREMENT"/>
    </rdf:Description>
</rdf:RDF>
```

.

Appendix D Example of Rule syntax.

SWRL

```
; rule inputRule

Implies (
     Antecedent
        (
                    swrlb:greaterThanOrEqual(D-variable(azi) 360)
        )
          Consequent(fail(I-variable(response), true)))
```

Appendix E Testing procedures.

A. Use Protégé to define the OWL ontology to be used in the test

See *"A Practical Guide To Building OWL Ontologies Using The Protege-OWL Plugin and CO-ODE Tools Edition 1.0"*

B. Import the OWL ontology into Eclipse by using the WSMO Eclipse plug-in.

The sequence of steps that the user should follow in order to be able to copy WSML descriptions from the local workspace into a remote repository is:

1. Create a WSML description using the WSMO Editor and save it in a project in the workspace.

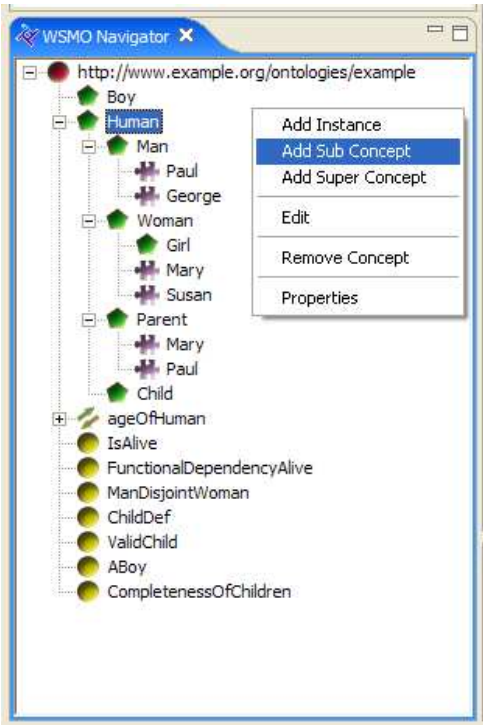


Figure 40. WSMO Navigator - ontology

2. Switch to the repository perspective.
3. Connect to a remote repository.
4. Copy the WSML file into the remote repository (Import from Workspace from the context menu.

> Note that WSML descriptions stored in a remote repository cannot be edited directly. The proper sequence of steps for editing a WSML description from a remote repository is:
>
> 1. Copy the WSML description from the remote repository into the local workspace (Save in Workspace from the context menu)
>
> 2. Switch back to the WSMO perspective
>
> 3. Apply the desired modifications to the WSML description (e.g. edit the WSMO entity using the respective editors from the WSMO perspective)
>
> 4. Copy the modified WSML description from the local workspace back into the remote repository (Import from Workspace from the context menu. At this point, the user will be prompted for confirmation whether to overwrite the WSML description that already exist in the repository

Annotate the OWL classes and instances to elements in the web services to be used in the test by using the SAWSDL Editor in Eclipse IDE.

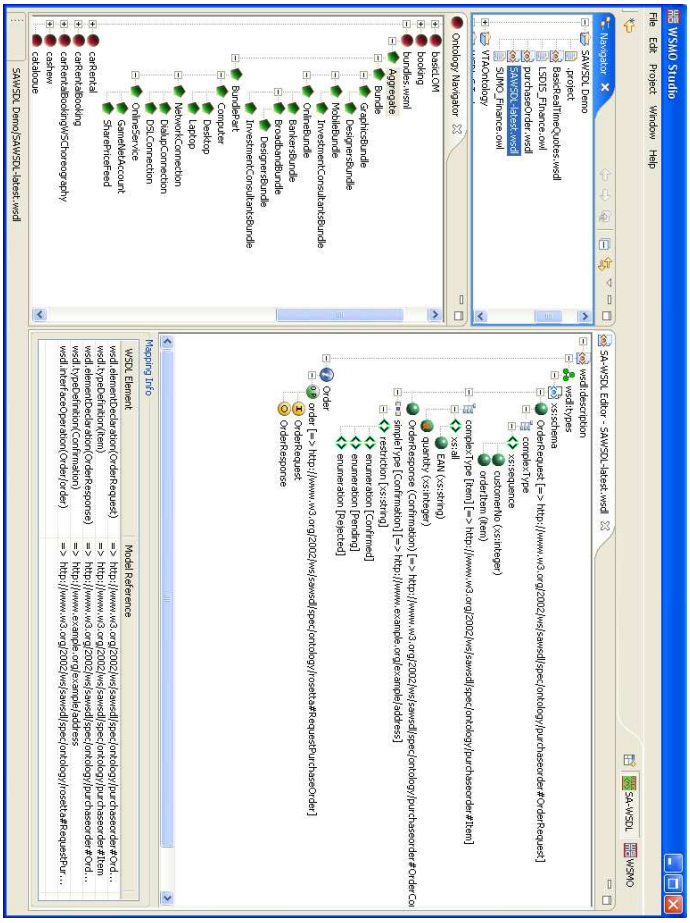


Figure 41. SAWSDL editor

Once the SAWSDL perspective is activated and a WSDL / SAWSDL file is opened from the Navigator, the user may start specifying mappings between WSMO elements (i.e. concepts in WSMO ontologies) and WSDL elements.

At present the following mappings are supported:

- Concept to simple / complex XML type
- Concept to WSDL interface
- Concept to WSDL operation
- Concept to WSDL message / element

The exact mappings can either be performed by dragging & dropping WSMO concepts (from the Ontology Navigator) into the corresponding WSDL elements (into the SAWSDL editor), or by selecting a WSDL element and choosing Add Reference from the context menu. In the latter case the user will be presented with the list of ontologies and concepts to select from.

Existing mappings can also be removed.

Setup the web services needed in the test by creating mock services. The mocked web services are created and deployed by the SoapUI Eclipse plug-in.

See http://www.soapui.org/userguide/overview.html

Implement agents in Java and run them within the Eclipse IDE. The SAWSDL4J Java API is used to get the annotated OWL references from the wsdl.

See http://knoesis.wright.edu/opensource/sawsdl4j/tute.html

References

Baker, C. J. O. and K.-H. Cheung (2005). Semantic Web: Revolutionizing the knowledge discovery in the life sciences. Boston/Dordrecht/London**:** 37.

Bechhofer, S., F. van Harmelen, et al. (2004). OWL web ontology language reference. W3C Recommendation, 10 February 2004.

Berners-Lee, T., J. Hendler, et al. (2001). "The Semantic Web." The Scientific American **May**: 7.

Boley, H. (2006, 2008-02-28). "The Rule Markup Initiative." from http://www.ruleml.org/.

> Rules in (and for) the Web have become a mainstream topic since inference rules were marked up for E-Commerce and were identified as a Design Issue of the Semantic Web, and since transformation rules were put to practice for document generation from a central XML repository (as used here). Moreover, rules have continued to play an important role in AI shells for knowledge-based systems and in Intelligent Agents, today both needing a Web interchange format, and such XML/RDF-standardized rules are now also usable for the declarative specification of Web Services.

Davis, M. (2006). Semantic wave 2006, part-1: executive guide to billion dollar markets.

Feigenbaum, L., K. G. Clark, et al. (2008, 15 Jan 2008). "SPARQL Protocol for RDF." Recommendation. from http://www-mit.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115/.

> The SPARQL Protocol and RDF Query Language (SPARQL) is a query language and protocol for RDF. This document specifies the SPARQL Protocol; it uses WSDL 2.0 to describe a means for conveying SPARQL queries to an SPARQL query processing service and returning the query results to the entity that requested them. This protocol was developed by the W3C RDF Data Access Working Group (DAWG), part of the Semantic Web Activity as described in the activity statement

Fjellheim, R. A., R. B. Bratvold, et al. (2008). CODIO - Collaborative Decision-making in Integrated Operations. SPE Intelligent Energy Conference and Exhibition. Amsterdam, Society of Petroleum Engineers.

> We describe CODIO, an Integrated Operations (IO) development project currently underway in the North Sea region: The aim of the project is to optimize the drilling process by ensuring that drilling teams generate a continuous stream of right decisions made at the right time. Current IO projects do not fully address the challenges posed by decision making in the new IO environment with abundant real-time data, operations centers, virtual multidisciplinary teams, mobile workforces etc. The CODIO project develops and evaluates a new model for decision support in IO. The model is based on advanced decision theory, combined with real-time situation assessment, collaborative ICT-supported work processes, and a semantic model for sharing data and knowledge. Decision support in CODIO exploits recent advances in Bayesian techniques, adapted to dynamic and (near) real-time decision situations. Constructing decision models "on the fly" is enabled by collaboration technology including visualization. Previous work on applying Semantic Web Standards (OWL, RDF) in IO is used to support proactive information gathering and sharing.

Fjellheim, R. A. and D. Norheim (2007). AKSIO - Active Knowledge System for Integrated Operations. Oslo, Computas**:** 20.

Gomadam, K., D. Brewer, et al. (2007). "SAWSDL4J." from http://knoesis.wright.edu/opensource/sawsdl4j/.

The SAWSDL4J project is an attempt to provide a clean object model for SAWSDL documents. SAWSDL is a leading specification from W3C that governs the subject of attaching semantic references to standard WSDL descriptions. Please check the official SAWSDL page at W3C for more details

Hansen, B. J., T. Gagnes, et al. (2007). Semantic Technologies, Norwegian Defence Research Establishment (FFI)**:** 56.

Horrocks, I. (2008). Ontologies and Databases. Semantic Days 2008. Stavanger, OLF.

Horrocks, I., P. F. Patel-Schneider, et al. (2004). "SWRL: A Semantic Web Rule Language Combining OWL and RuleML." World Wide Web Consortium.

HP Laboratories, B. (2001-2008). "Jena – A Semantic Web Framework for Java ", from http://jena.sourceforge.net/.

Jacobs, I. (2007). "About the World Wide Web Consortium (W3C)." from http://www.w3.org/Consortium/.

Joel Farrel, I. and D. I. Holger Lausen. (2007, 28 August 2007). "Semantic Annotations for WSDL and XML Schema." W3C Recommendation, from http://www.w3.org/TR/sawsdl/.

This document defines a set of extension attributes for the Web Services Description Language and XML Schema definition language that allows description of additional semantics of WSDL components. The specification defines how semantic annotation is accomplished using references to semantic models, e.g. ontologies. Semantic Annotations for WSDL and XML Schema (SAWSDL) does not specify a language for representing the semantic models. Instead it provides mechanisms by which concepts from the semantic models, typically defined outside the WSDL document, can be referenced from within WSDL and XML Schema components using annotations.

Kay, M. (2007, 23 January 2007). "XSL Transformations (XSLT) Version 2.0." World Wide Web Consortium, from http://www.w3.org/TR/xslt20/.

This specification defines the syntax and semantics of XSLT 2.0, a language for transforming XML documents into other XML documents.

XSLT 2.0 is a revised version of the XSLT 1.0 Recommendation [XSLT 1.0] published on 16 November 1999.

XSLT 2.0 is designed to be used in conjunction with XPath 2.0, which is defined in [XPath 2.0]. XSLT shares the same data model as XPath 2.0, which is defined in [Data Model], and it uses the library of functions and operators defined in [Functions and Operators].

XSLT 2.0 also includes optional facilities to serialize the results of a transformation, by means of an interface to the serialization component described in [XSLT and XQuery Serialization].

Kay, M. (2008, 12.may 2008). "The XSLT and XQuery Processor." from http://saxon.sourceforge.net/.

KG, R. S. G. C. (2004). "What is RacerPro? ." from http://www.racer-systems.com/products/racerpro/index.phtml.

Kopecky, J., T. Vitvar, et al. (2007). "SAWSDL: Semantic Annotations for WSDL and XML Schema." IEEE INTERNET COMPUTING **11**(6): 8.

Manola, F. and E. Miller (2004). RDF Primer. **10**.

McComb, D. (2005). The CIO's guide to semantics, version 2, Semantic Arts, Incorporated.

Members, T. (2003). <WITSML/> Core Application Program Interface. W. C. A. V. 1.2.doc, www.npsinc.com.

Norheim, D. (2007). "Slide from a 'Semantic Interoperability' presentation, Computas AS.

openRDF.org. (2007). "Sesame." from http://www.openrdf.org/index.jsp.
Sesame is an open source Java framework for storing, querying and reasoning with RDF and RDF Schema. It can be used as a database for RDF and RDF Schema, or as a Java library for applications that need to work with RDF internally. For example, suppose you need to read a big RDF file, find the relevant information for your application, and use that information. Sesame provides you with the necessary tools to parse, interpret, query and store all this information, embedded in your own application if you want, or, if you prefer, in a separate database or even on a remote server. More generally: Sesame provides application developers a toolbox that contains useful hammers, screwdrivers etc. for doing 'Do-It-Yourself' with RDF.

Polikoff, I. and D. Allemang (2003). TopQuadrant Technology Briefing. Semantic Technology vl. l, TopQuadrant technical report, available at http://www. TopQuadrant. com/document ts/TQ03_Semantic_Technology_Briefmg. PDF.

Pollock, J. T. and R. Hodgson (2007). Adaptive Information: Improving Business Through Semantic Interoperability, Grid Computing, and Enterprise Integration. **20:** 81-84.

Sandsmark, N. (2008). "IO in the High North 2008-2011." from http://www.posccaesar.com/en-GB/PortalObject/2862/POSCCaesar.aspx.

Shafiq, O., M. Moran, et al. (2007). Investigating Semantic Web Service Execution Environments: A Comparison between WSMX and OWL-S Tools. Internet and Web Applications and Services, 2007. ICIW 07.
Second International Conference on Morne, Mauritius.
The application of semantics in Web Services as Semantic Web Services for dynamic discovery, composition, invocation and monitoring has been very helpful in enabling Enterprise Application Integration and E-Commerce. There are many initiatives that

aim to realize the Semantic Web Services to enable effective exploitation of semantic annotations, and two major of them are Web Service Modeling Ontology (WSMO) and Ontology Web Language for Services (OWL-S). Several tools have been developed to realize both the conceptual models i.e. Web Services Execution Environment (WSMX) is the reference implementation for WSMO, on the other side OWL-S reference implementation exists in the form of loose collection of individual tools like OWL-S Editor, OWL-S Matchmaker, OWL-S Virtual Machine, OWL-S IDE, WSDL2OWL-S converter and OWL-S2UDDI converter etc. In this paper, we have conducted a comparison of both the reference implementations to identify similarities and differences between them and to evaluate their potential to become widely accepted implementation recommendations.

Studer, R., V. R. Benjamins, et al. (1998). Knowledge engineering: Principles and methods, Elsevier. **25:** 161-197.

Teijgeler, H. (2007). "ISO 15926 Part 7 – Implementation of ISO 15926 ", from http://www.fiatech.org/projects/idim/dsciso15926.htm.

Wald, L., G. Teledetection, et al. (1999). Some terms of reference in data fusion. **37:** 1190-1193.

Wilms, H. (2007, 31 aug 2007). "Semantic Annotations for WSDL and XML Schema (SAWSDL) Becomes a W3C Recommendation." Tracking change and innovation in the enterprise software development community, from http://www.infoq.com/news/2007/08/sawsdl-recommendation.