



University of Fribourg

Informatics Department

Bachelor in Management of Information

ONTOLOGIES FOR THE SEMANTIC WEB

Bachelor Thesis

Michael Dejen

Av. De Chailly 34

1012 Lausanne

Adviser

Dr. Stephan Hüsemann

December 2007

Abstract

It was back in 1998; Tim Berners-Lee (www inventor) exposed his taught about the next-step of Hypertext Markup Language (HTML) the medium we use to display information which is human-oriented to “Semantic Web” a structure in a way useful and understandable by machines which is machine-oriented.

Before understanding the Semantic Web, one must first be somewhat familiar with the enabling technologies upon which the Semantic Web is based. The extensible markup language (XML), uniform resource identifiers (URIs), resource definition framework (RDF) and ontologies are all keys to the realization of the Semantic Web. At the first part of this bachelor thesis we will try to have a broad overview of each technology, and in the second part we will survey, analyze and see existed ontology file with the hypermedia-based OWL Ontology editor “Swoop” the power of ontologies to the Semantic Web.

Keywords: XML, URI, RDF, RDF Schema, OWL, Ontology, Semantic Web.

Acknowledgement

I would like to acknowledge and extend my heartfelt gratitude to Dr. Hüsemann Stefan to the completion of this Bachelor thesis possible. I wish to thank as well all those who helped me, without them, I could not have completed. Most especially to my family and friends and to God, who made all things possible.

Table of contents

I	Introduction	1
1	Problem description	1
2	Questions / subjects to be treated.....	2
2.1	What is Semantic Web?	2
2.2	How is the Semantic Web related to the existing Web?.....	2
2.3	How does the Semantic Web differ from the existing Web?.....	2
2.4	Is the Semantic Web just research, or does it have industrial applications?.....	3
2.5	Where can we use the Semantic Web?.....	3
3	Methodology	3
4	Objective and output of this work	4
II	Theoretical Part: Semantic Web	5
1	Definition of Semantic Web	5
2	Semantic Web Architecture and Application	6
2.1	First Generation – Keywords	7
2.2	Second Generation – Statistical Forecasting	8
2.3	Third Generation – Natural Language Processing.....	8
2.4	Forth Generation – Semantic Web Architecture and Application	9
3	Standards linked to Semantic Web	10
3.1	Global naming scheme (URIs).....	10
3.2	Extensible Markup Language (XML)	10
3.3	Standard Syntax – RDF	12
3.4	Describing properties - RDF Schema	17
3.5	Ontology	19
3.6	Logic and Proof	22
3.7	Trust.....	22
4	Usage of the Semantic Web	23
4.1	Enterprise Information Discovery and Integration.....	23
4.2	E-Commerce	23
4.3	Knowledge Management	24
	Conclusion	24

III Practical part: Ontologies for the semantic web	26
1 Usage of ontology.....	26
1.1 Good ontology outcome.....	27
2 Swoop – OWL Ontology editor	29
2.1 Reasons to use the ontology tool SWOOP	32
2.1 Description Logic Reasoners.....	33
3 Navigating an ontology using Swoop	37
4 SWOT analysis: Ontologies for the Semantic Web.....	50
4.1 Strengths	50
4.2 Weaknesses	50
4.3 Opportunities	51
4.4 Threats.....	52
5 Future of Information Management with Semantic Web.....	53
Conclusion and possible Future Work	55
References.....	57

Table of Figures

Figure 1: Semantic Web Stack	6
Figure 2: RDF Schema typed hierarchy.....	17
Figure 3: Swoop screenshot	29
Figure 4: SWOOP Architecture	31
Figure 5: The statements seen by the OntModel	36
Figure 6: Swoop's overview	37
Figure 7: Definition of ALCON(D).....	38
Figure 8: List of Reasoners	39
Figure 9: Multiple Inheritance with no Reasoner	39
Figure 10: Multiple Inheritance with the reasoner “Pelette”	40
Figure 11: Natural Language tab view	40
Figure 12: Concise Format for the Multiple Inheritance Class.....	41
Figure 13: Axiom vs. RDF/XML.....	41
Figure 14: Axioms of superclass “Male” vs. subclass “MaleStudentWith3Daughters”	42
Figure 15: Irrelevant parts of axioms for the super and subclasses	42
Figure 16: Disjoint Axioms.....	43
Figure 17: Max. Branching Factor of Class Tree with the reasoned switched on	44
Figure 18: Visualization with no Reasoner	45
Figure 19: Collapsed class view.....	46
Figure 20: Visualization with reasoner switched on	46
Figure 21: Visualizing the Multiple Inheritance	47
Figure 22: Species Validation view of the downloaded ontology	48
Figure 23: Link to the structure of the Class.....	49
Figure 24: List of Ontologies loaded	49



Table of Codes

Code 1: HTML Code example	11
Code 2: XML code example.....	11
Code 3: Example of HTML code	13
Code 4: Example of XML/RDF code.....	16
Code 5: RDF Schema syntax in XML.....	18
Code 6: OWL code for intersectionOf property.....	21

I Introduction

1 Problem description

What the traditional Web does for the text documents in our lives, the Semantic Web does for all our data and information. Today, on our Web pages, we can build a pointer to another Web page. But we can't link data together in the way we can link pages together. We can't point from a value in one database to some other value in some other database. To use a simple example, if your driver's license number is in one place and your vehicle identification number is in another, there should be a way of linking those two things together. There should be a way for machines to understand that those two things are related.

One of the major obstacles to this has been the fact that most information on the Web is designed for human consumption, and even if it was derived from a database with well defined meanings (in at least some terms) for its columns, that the structure of the data is not evident to a machine browsing the web. Leaving aside the artificial intelligence problem of training machines to behave like people, the Semantic Web approach instead develops languages for expressing information in a machine processable form.

Right now, it's very difficult to browse data on the Web. We can use a search engine that gives us the results of a query and draws them as a list, but we can't click on one of those values and see what it really means and what it's really related to. Today's social networking is trying to improve this, with things like tagging. But if you typed "polish" and we typed "polish," how do we know we're talking about the same thing? You might be talking about a language and we might be talking about something that goes on furniture. On the other hand, if those two names are precisely identified, they don't accidentally overlap and it's easier to understand the data we've published. So the technology of the Semantic Web is, in a sense, the technology of precise vocabularies.

The Semantic Web would allow a machine to go out across the Web and find the things we're looking for. It's very hard for this to happen with just language

descriptions. Our idea is to have machine-readable information shadowing the human-readable stuff. So if we have a page that says, "His name is Michael Dejen. Here is a picture of his daughter," the machine realizes that it's a person, that has a first name and a last name, that he is the father of another person, and that she's a female person. The level of information a machine needs would vary from application to application, but just a little of this could go a long way—as long as it can all be linked together. And the linking is the Web part of the Semantic Web. This is all about adding meaning to the stuff we put on the Web—and then linking that meaning together.

2 Questions / subjects to be treated

During this bachelor thesis we'll try to find out the answer for the questions which might be related and raised about the Semantic Web. The following questions will be addressed.

2.1 What is Semantic Web?

How could we come up with the idea of Semantic Web? Whose idea was it starting from the beginning? We will try to see as well what we mean by "meta-data".

2.2 How is the Semantic Web related to the existing Web?

We will see the different points which relate the Semantic Web to that of the web that we are familiar with.

2.3 How does the Semantic Web differ from the existing Web?

We'll see the major differences between the Semantic Web and the existing web, why do we want to come up with the idea of extending the existing web?

2.4 Is the Semantic Web just research, or does it have industrial applications?

As any technological innovation, the Semantic Web has gone through research labs, and still is. So it's right to ask who are willing to inject this new technology in order to see whether it's promising or not.

2.5 Where can we use the Semantic Web?

There are domains which are forecasting to increase their productivity by applying the Semantic Web, like that of: Enterprise information discovery and integration; E-Commerce; Knowledge management; etc.

3 Methodology

The Semantic Web activities are mostly of the World Wide Web Consortium (W3C) so we are mainly concentrating on their links while doing the theoretical part of this bachelor thesis. Besides that assisting the registered conferences and some tutorials concerning the Semantic Web like that of: A Short tutorial on Semantic Web by York Sure (from University of Karlsruhe); Understanding Ontologies found on the web by Bijan Parsia and Bernardo Cuenca Grau (from University of Manchester); Where the social web meets the Semantic Web by Tom Gruber (from Intraspect Software) and watched the interview with Tim Berner-lee (Inventor of the World Wide Web) and Marko Grobelnik (from the department of Intelligent Systems of the J. Stefan Institute); Why is Business Semantics the New Hot Topic? - 2004 Interview with Dave McComb, published in DM Review's DM Direct Newsletter. And at last but not least referring the books, amongst all: Semantic Web Services: Theory, Tools and Applications by Cardoso, J.; The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management. John Wiley & Sons...

4 Objective and output of this work

The objective of the semantic web in general is to provide very significant potential benefits to end users and developers in a number of key ways; ease of information integration, more efficient searching, more effective re-use, etc. And specifically the objective of this work is to see the evolution of web towards knowledge management. The Semantic Web envisions a metadata-rich Web where presently human-readable content will have machine- understandable semantic. The Web Ontology Language (OWL) from W3C is an expressive formalism for modelers to define various logical concepts and relations.

The first part of this work focuses on the definitions and descriptions of the Semantic Web standards and components. How they function and interoperate.

And the second part focuses on the survey and analyze of an ontology file called Koala.owl which is found in the bookmark of the ontology editor SWOOP. We simply download the owl file in the editor SWOOP and use the editor as a framework for automating the analysis tasks.

II Theoretical Part: Semantic Web

1 Definition of Semantic Web

Semantics means the study of meaning and Semantic technologies are about software standards and methodologies which allow us to have unambiguous meaning about the information that we have in hand. So what do we mean by Semantic Web is that it's a web which uses the three key software standards: RDF, RDFS and OWL in order to manage the billions of documents out there in the net by creating a link and an index to strengthen machines interpretation of the information which are found in the documents (Wilshire 2006).

The Semantic Web vision was conceived by Tim Berners-Lee, the inventor of the World Wide Web. The World Wide Web changed the way we communicate, the way we do business, the way we seek information and entertainment – the very way most of us live our daily lives (Altova 2006). Calling it the next step in Web evolution, Berners-Lee defines the Semantic Web as “a web of data that can be processed directly and indirectly by machines.”

In the Semantic Web data itself becomes part of the Web and is able to be processed independently of application, platform, or domain(Altova 2005). This is in contrast to the World Wide Web as we know it today, which contains virtually boundless information in the form of documents. We can use computers to search for these documents, but they still have to be read and interpreted by humans before any useful information can be extrapolated. Computers can present you with information but can't understand what the information is well enough to display the data that is most relevant in a given circumstance. The Semantic Web, on the other hand, is about having data as well as documents on the Web so that machines can process, transform, assemble, and even act on the data in useful ways.

The Semantic Web Stack (Figure 1 below) from a 2000 presentation by Tim Berners-Lee (Inventor of the World Wide Web) could allow us to grasp the idea behind it.

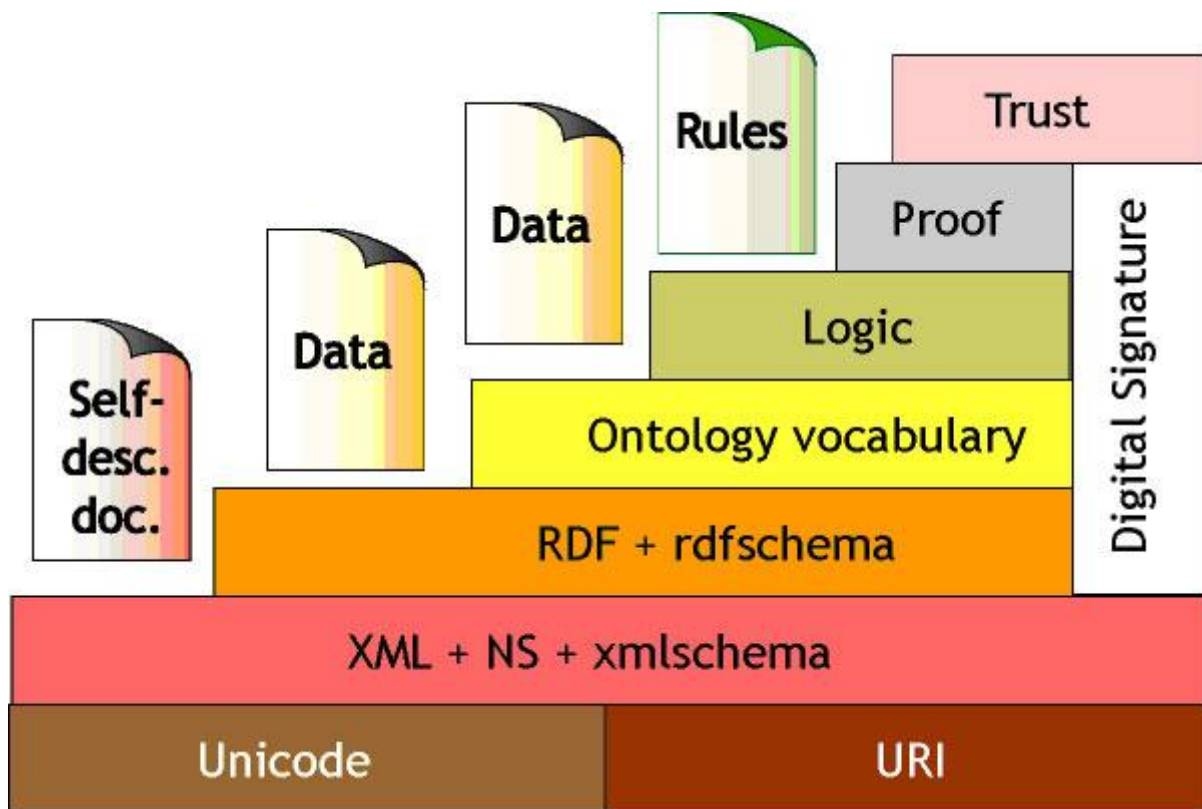


Figure 1: Semantic Web Stack (Berners-Lee 2000)

The stack (Figure 1 above) shows how these languages are expected to be found, one on top of another. We illustrate the couches of the stack (Figure 1 above) in more detail, to help explain the foundations of this new and emerging web in the section 3.

2 Semantic Web Architecture and Application

Semantic Web architecture and applications are the next generation in information architecture (Generation 2006). In the coming sub-chapters, which are entirely taken from the document (Generation 2006), we will review four generations of information management applications (Keywords, Statistical, Natural Language & Semantic Web) and defines their key features and limitations.

2.1 First Generation - Keywords

Keyword technologies were originally used in IBM's free text retrieval system in the late 1960's. These tools are based on a simple scan of a text document to find a key word or root stem of a key word. This approach can find key words in a document, and can list and rank documents containing key words. But, these tools have no ability to extract the meaning from the word or root stem, and no ability to understand the meaning of the sentence.

Advanced search: Most keyword systems now include some form of Boolean logic "AND", "OR" functions to narrow searches. This is often called "advanced search". But, using Boolean logic to exclude documents from a search is not "advanced". It is an arbitrary and random means to reduce the size of the source database to reduce the number of documents retrieved. This "advanced search" significantly increases false negatives by missing many relevant source documents.

Application: Keyword tools are appropriate for creating a word location or a list of documents which contain specific defined keywords and root stems. These are not capable of understanding similar words, the meanings of words, or their relationships, or context.

Problems: The most common problems with keyword tools are; a) false negatives (no matches found because the word or stem are not exactly identical: "big" & "large"), b) false positives (too many unrelated matches found because a root stem finds many unrelated words: "process" & "processor") and c) scale factors (keyword search tool produce very long random lists of documents if the source database is large, and the relevance rankings are highly misleading).

Examples: The most common examples of key word tools are web site "search" tools and the Microsoft "Find" function (control + "f" key) in Microsoft Office applications.

2.2 Second Generation – Statistical Forecasting

Statistical forecasting first finds keywords; and then calculates the frequency and distance of these keywords. Statistical forecasting tools now include many techniques for predictive forecasting, most often using inference theory. The frequency and distribution of words has some general value in understanding content. But, cannot understand the meaning of words or sentences; or provide context. These tools are still limited by keyword constraints; and can only infer simplistic meaning from the frequency and distribution of words.

Applications: Statistical forecasting tools are appropriate for performing simple document searches where the desired output is a list of documents which contain specific words which must then be read and classified and summarized manually by end users. These are not capable of understanding the meaning or context or relationships of documents.

Problems: The most common problems with statistical forecasting tools are; a) keyword limitations of false positives and false negatives, b) misunderstanding the meaning of words and sentences (“man bites dog” is the same as “dog bites man”), c) lack of context, d) scale factors: a single statistical relevance ranking creates huge “Google” lists of many irrelevant documents (“you have 100,000 hits”).

Examples: The most common statistical forecasting tool is “Google” and many other tools using inference theory and similar analysis and predictive algorithms.

2.3 Third Generation – Natural Language Processing

Natural language processors focus on the structure of language. These recognize that certain words in each sentence (nouns and verbs) play a different role (subject-verb-object) than others (adjectives, adverbs, articles). This understanding of grammar increases the understanding of key words and their relationships. (“man bites dog” is different from “dog bites man”). But, these tools cannot extract the understanding of the words or their logical relationship beyond their basic grammar.

And, these cannot perform any information summary, analysis or integration functions.

Applications: Natural language tools are appropriate for linguistic research and word-for-word translation applications where the desired output is a linguistic definition or a translation. These are not capable of understanding the meaning or context of sentences in documents, or integrating information within a database.

Problems: The most common problems with linguistic tools are; a) keyword limitations of false positives and false negatives, b) misunderstanding the context (does “I like java” mean an island in Indonesia, a computer programming language or coffee?). Without understanding the broader context, a linguistic tool only has a dictionary definition of “Java” and does not know which “Java” is relevant or what other data related to a specific “java” concept we will see some more in the section 3.3.

Examples: The most common natural language tools are translator programs which use dictionary look up tables to convert words and language-specific grammar to convert source to target languages.

2.4 Forth Generation – Semantic Web Architecture and Application

Semantic web architecture and applications are a dramatic departure from earlier database and applications generations. Semantic processing includes these earlier statistical and natural language techniques, and enhances these with semantic processing tools. We will see the Standards linked to Semantic Web in detail in the first theoretical part and analyze the ontologies for the Semantic Web in the second practical part of this Bachelor thesis.

3 Standards linked to Semantic Web

3.1 Global naming scheme (URIs)

If any Semantic Web application is to be able to access and use data from any other such application, every data object and every data schema/model must have a unique and universal means of identification (hp 2007). These identifiers are called Universal Resource Identifiers (URIs).

One form of the URI that we are all familiar is the URL or Unified Resource Locator. It's an address of any webpage. In addition to URLs, there are other forms of URIs examples include the "mailto:" (Swartz and Hendler October, 2001) URIs which are used to encode email addresses and so on.

The URIs are free to be declared by the author about everything he needs, whether it's real or virtual. But the problem here is that, since anyone can create a URI, we will inevitably end up with multiple URIs representing the same thing and we have no way to figure out whether two URIs are definitely the same or not (Fensel, Lausen et al. 2006).

3.2 Extensible Markup Language (XML)

XML was designed to be a simple way to send documents across the Web. It allows anyone to design their own document format and then write a document in it (Geroimenko 2006). These document formats can include markup to enhance the meaning of the document. If we include enhanced meaning in our documents, they become much more useful. Instead of only being able to be used by one program (a web browser, for example) they can be used by many programs, each only using the markup it understands and ignoring the rest. Even better, each program is free to interpret the markup in the way that's best for it. For example, in a document where words are marked as "emphasized" a web browser might display them in bold. On the other hand, a voice browser (which reads web pages out loud) may read them with extra emphasis. Each program is free to do what it feels is appropriate.

XML is user definable and domain specific markable language as we said in the previous paragraph. Let's try to compare and contrast it with HTML by basing ourselves to the "Code 1 and Code 2" below and see why we cannot use it in order to have ontology in the Semantic Web.

HTML:

```
<H1>Information Systems</H1>
  <UL>
    <LI>Teacher: Dr. Stefan Hüsemann
    <LI>Students: Bachelor
  </UL>
```

Code 1: HTML Code example (cf. [Sure 2003, p. 8])

XML:

```
<course>
  <title> Information Systems </title>
  <teacher> Dr. Stefan Hüsemann </teacher>
  <student> Bachelor </student>
</course>
```

Code 2: XML code example (cf. [Sure 2003, p. 8])

The HTML can be interpreted as a text that our browser is able to interpret to make a nice layout. So we know that in the code HTML (Code 1 above) H1 makes the characters found here "Information Systems" bold and to make it bigger and so on. This is supposed to be known by the browser which means that it's hard coded in the browser. HTML was designed to contain instructions for browsers on how to display the pages to users. It contains very little information about the real semantics of the page; it treats every information as pages.

The next step is to be more general, that's XML. We say this is a course (Code 2 above) which is additional information which was not included in the HTML (Code 1

above). So the question here is why don't we use XML for the Semantic Web, given the fact that we want to have ontologies in the Semantic Web and XML seems to fit, so why not? The answer is, XML doesn't define what the domain specific ontological vocabularies are, and that is the ontological modeling primitives. As shown in the (Code 2 above) the course for us is a concept which is relevant for us and we know that it is related to a person and this person is a teacher who teaches students that are in the bachelor level. This is what we want to say also to the computer, otherwise it cannot handle and cannot draw inference out of that. If we want this to happen we need to have prearrangement with the other end too and this is what is completely missing in the XML. We only have a labeled tree with a root node course and as a sub nodes title, teacher and students (Code 2 above) in the XML and that is just about it.

Even though, it's feasible if all the agents have a close collaboration. Let's say if we have agents in a small and stable community and we store ontologies in an XML based format and we defined some domain specific vocabularies mainly in XML Schema for our own purpose and this is hard coded to our tools. Therefore no other tool without our knowledge can make a use out of it. It means that it's only our community who knows what the specifications of the vocabularies are about.

So as a small conclusion XML cannot be used because of the fact that it is not machine understandable language.

3.3 Standard Syntax – RDF

As we tried to see in the previous parts we have as a basic infrastructure XML and Namespaces and on the top we add RDF and RDF Schema which are the first set of ontological primitives where we have prearranged agreements on. So this is the next step in the hierarchy (Figure 1 above).

The computer industry has agreed, by large, to use XML to represent not only human readable documents, but data in general. The XML standards give a syntactic structure for describing data (hp 2007). Unfortunately, XML can be used in many different ways to describe the same data. This makes it too open and arbitrary to

support the type of widespread and ad hoc data integration envisaged for the Semantic Web. The semantic web vision proposes to represent machine processable information using Resource Description Framework (RDF), which extends XML. RDF defines a general common data model that adheres to web principles (W3C 2001) The W3C are strong supporters of this approach.

What we get out of RDF is that it provides metadata of web resources. The common definition of metadata is data about data, to say something about anything. In order to achieve this, RDF community introduces object attribute triples (Object, attribute & value) and chained them to form a graph. RDFs have XML syntax and are serialized with XML. As HTML represents documents RDF represents data.

Let's have an example to give us a clear picture of how we could compare our traditional HTML with that of XML/RDF code (Code 3 below for HTML and Code 4 below for the XML/RDF code) from (Mazzocchi 2000).

```
<html>

  <body>

    <h1>My trip to the Java Island</h1>

    <h3>by Michael Dejen (michael.dejen@unifr.ch)</h3>

    <p>The trip on the island was great:

      <ul>

        <li>I tasted great Java coffee</li>

        <li>I finished my latest program written in
          Java</li>

        <li>I did a great trip on the islands around
          Java</li>

      </ul>

    </p>

    <p>The trip was arranged by Travels@Java.com Inc.
      (info@javatravels.com).</p>

  </body>

</html>
```

Code 3: Example of HTML code (cf. [Mazzocchi 2000, p. 5])

The semantic analysis of this page is very hard for humans and close to impossible for computer programs. For example, the second sentence could mean that I finished my latest program that I wrote when visiting the island of Java, or, more reasonably, that I finished my latest programming written using the Java programming language.

But the page in general requires our knowledge to contain notions about Java as an island, as a programming language and as a coffee brand.

We could continue our semantic analysis by saying that the page is written using the English language, written by Michael Dejen who can be reached at the email address `michael.dejen@unifr.ch` and the page is mainly about a trip to the island of Java.

But computer programs that perform on syntax analysis rather than semantic analysis might not be able to understand that `Travels@Java` is not the author of the page, nor it's a valid email address.

But the author of the page knows the meaning of what he writes and he would like to transmit it as such to humans as well as computer programs and not letting them guess about what it contains.

So we write:

```
<?xml version="1.0"?>

<page xmlns="http://mysite.org/page"
      xmlns:geo="http://geography.gov/terms"
      xmlns:food="http://fao.org/metadata/food/en"
      xmlns:man="http://onu.gov/metadata/mandkind/en"
      xmlns:man="http://acm.gov/metadata/computing"
      xmlns:email="http://ietf.org/schemas/email"
      xmlns:com="http://nafta.org/terminology/en"
      xml:lang="en">

  <title>My trip on the <geo:island>Java
    Island</geo:island></title>

  <author>

    <man:man man:age="28"
      email:address="Michael.dejen@unifr.ch"> Michael Dejen
    </man:man>

  </author>

  <content>

    <p>My trip on the island was great:

    <ul>

      <li>I tasted great <food:coffee>Java</food:coffee>
        coffee</li>

      <li>I finished my latest program written in
        <comp:lang>Java</comp:lang></li>

      <li>I did a great trip on the islands around
        <geo:island>Java</geo:island></li>

    </ul>
```

```

    </p>
    <p>The trip was arranged by
        <com:company com:type="incorporated"
            email:address"info@javatravels.com">
            Travels@Java.com Inc.
        </com:company>
    </p>
</content>
</page>

```

Code 4: Example of XML/RDF code (cf. [Mazzocchi 2000, p. 6])

This page uses XML and Namespaces to add specific semantic information about the included content while removing all style information. While it's harder for humans to understand this page as it is, it's much easier for computer programs since all the necessary semantic information is placed by the page author and namespaces identify the semantic connections univoquely.

No heuristic or guess takes place when a computer program parses this page because it doesn't need to understand the page to be able to associate content to its semantic areas, which are here identified by namespaces (Mazzocchi 2000).

So, the word Java is connected to three different meanings and will be up to the user requesting the page to determine which meaning she/he is interested it, the page parser does not have to understand any of this but simply index the page with this information and use it a query time to rate the page.

We must understand that the above is nothing new since all centralized information systems have been able to perform context sensitive searches for decades and, in fact, this is a particular flavor of queries that relational databases have been performing for decades.

Instead of specifying rows and tables, we search for some text included in specific contexts. For example, search for Java inside the island tag of the <http://geography.gov/terms> namespace and trip in English language.

3.4 Describing properties - RDF Schema

RDF itself is a composable and extensible standard for building data models. To support the definition of a specific vocabulary for a data model, which can itself be published, another layer is required. RDF schema allows a designer to define and publish the vocabulary used by an RDF data model, i.e. define the data objects and their attributes. For instance, it might define that people have a phone attribute. RDFS also uses *class* and *subclass*, so for example *employee* could be defined as a sub-class of *person* and so on (Sure 2003).

In order for a computer programs understand a term with different signification. The RDF Schema specification defines a way to indicate such mechanical semantic connections between metadata schemas and allow the search engines to be more flexible during searching.

RDF Schema defines vocabulary for RDF and organizes this vocabulary in a typed hierarchy (Sure 2003). What we have is a class and a `subClassOf` a relationship and type which could be an instance relationship for the property (relationship) and `subPropertyOf` which would be the starting and ending point of a relationship.

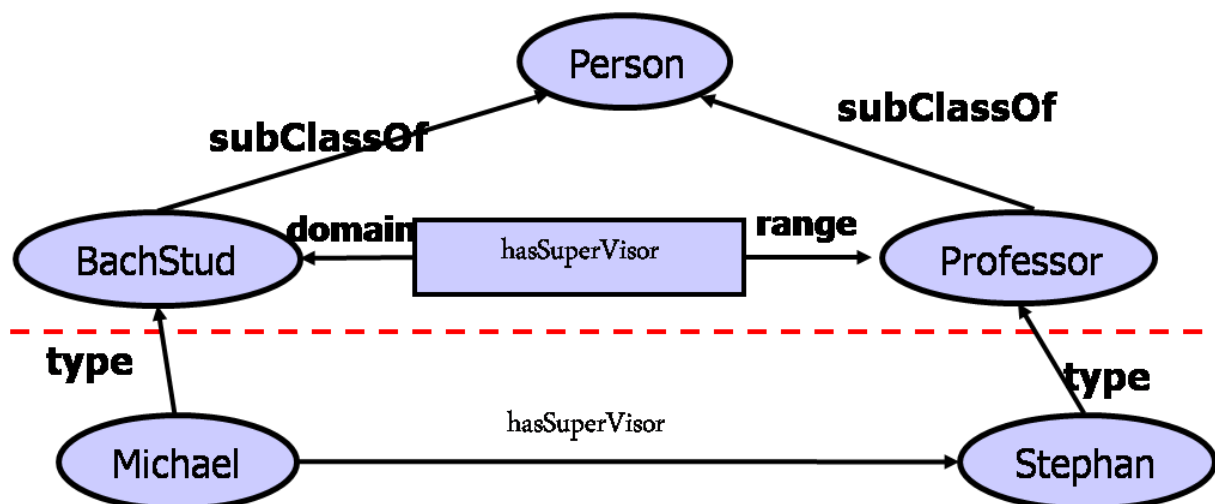


Figure 2: RDF Schema typed hierarchy (cf. [Sure 2003, p. 14])

As we look in the Figure 2 above of the RDF Schema, we see that the domain would be "BachStud" and the range would be "Professor" with the triple chain. We can see as well that "Michael" would be a type or instance of the class "BachStud" which is a subClassOf "Person" and "Stephan" would be a type or instance of the class "Professor" which is a subClassOf "Person" as that of the class "BachStud".

RDF Schema syntax in XML seems like (Sure 2003) (code below):

```
<rdf:Description ID="MotorVehicle">
  <rdf:type resource="http://www.w3.org/...#Class"/>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/...#Resource"/>
</rdf:Description>

<rdf:Description ID="Truck">
  <rdf:type resource="http://www.w3.org/...#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description ID="registeredTo">
  <rdf:type resource="http://www.w3.org/...#Property"/>
  <rdfs:domain rdf:resource="#MotorVehicle"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Description>

<rdf:Description ID="ownedBy">
  <rdf:type resource="http://www.w3.org/...#Property"/>
  <rdfs:subPropertyOf rdf:resource="#registeredTo"/>
</rdf:Description>
```

Code 5: RDF Schema syntax in XML (Sure 2003, p. 15)

We could say that "MotorVehicle" is a class which would be the first expression of the Code 5 above. We could also say a "Truck" is a class and a subClassOf a "MotorVehicule". In the third expression a "MotorVehicle" as a domain is "registeredTo" a "Person" which would be the range. In the forth expression (Code 5 above) we could say that something is "ownedBy" a person for example and the property "ownedBy" is a subPropertyOf "registeredTo". These short glimpses could give us more or less the technical details of the RDF Schema syntax in XML.

As a small conclusion why RDF is serialized with XML first of all is that it gives us a flexibility to embed the metadata in to the XML documents and if there are RDF aware agents, they can easily make a use out of it.

But still what is missing in RDF and RDF Schema is that there is no precisely described meaning (Sure 2003). We have no inference model and this is the key aspect if we have complex problems that we would like to solve and if we have large set of actions to execute. And this is not included in the RDF and RDF Schemas.

3.5 Ontology

The next step from RDF and RDF Schema in the Semantic layer (Figure 1 above) is to move to ontology vocabulary, for example to define an inference model which was missing in the RDF and RDF Schema, this is actually ongoing work (research) but it has a standard by the World Wide Web Consortium which is called OWL (Web Ontology Language). So what do we exactly mean when we are talking about Ontology is that in order to support the knowledge sharing process, we all need to speak the same language and understand one another. Therefore, ontologies enable a better communication between humans, between humans and machines (software agents) and between machines (software agents). It enables also the reuse of domain knowledge like if someone develops an ontology in detail and if the others find it useful for their research for example or whatever their needs, they can simply reuse it for their domains or extend it to their domain of interests.

If data is to be truly 'understandable' by multiple applications, and therefore become information, semantic interoperability is required. Syntactic interoperability is all about parsing data correctly. Semantic interoperability requires mapping between terms, which in turn requires content analysis. This requires formal and explicit specifications of domain models, which define the terms used and their relationships. Such formal domain models are sometimes called ontologies (hp 2007). Ontologies define data models in terms of classes, subclasses, and properties like that of the RDF and RDF Schema that we saw above in the section 3.3 and 3.4 above.

OWL builds on RDF and RDF Schema and adds more vocabulary for describing properties and classes: among, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties and characteristics of properties (e.g. symmetry), and enumerated classes (W3C 2004). OWL is presented as a successor of RDFS and an emerging web ontology language standard recommended by W3C.

OWL provides three increasingly expressive sublanguages designed for use by specific communities of users and implementers (W3C 2004):

- OWL Lite - which supports users primarily needing a classification hierarchy and simple constraints.
- OWL DL - which supports users who want maximum expressiveness without losing computational completeness and decidability of reasoning systems.
- OWL Full - which is intended for users who want maximum expressiveness and the syntactic freedom of RDF without computational guarantees.

Let's have a look at an example from the explanation of the property `owl:intersectionOf` from (W3C 2004).

The `owl:intersectionOf` property links a class to a list of class descriptions. An `owl:intersectionOf` statement describes a class for which the class extension contains precisely those individuals that are members of the class extension of all class descriptions in the list.

An example:

```

<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Tosca" />
        <owl:Thing rdf:about="#Salome" />
      </owl:oneOf>
    </owl:Class>
    <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#Turandot" />
        <owl:Thing rdf:about="#Tosca" />
      </owl:oneOf>
    </owl:Class>
  </owl:intersectionOf>
</owl:Class>

```

Code 6: OWL code for intersectionOf property (W3C 2004)

In this example the value of `owl:intersectionOf` is a list of two class descriptions, namely two enumerations, both describing a class with two individuals. The resulting intersection is a class with one individual, namely Tosca. As this is the only individual that is common to both enumerations.

NOTE: This assumes that the three individuals are all different. In fact, this is not by definition true in OWL. Different URI references may refer to the same individuals, because OWL does not have a "unique names" assumption. We can find OWL language constructs for making constraints about equality and difference of individuals.

Instead of presenting the web information as it is. We may think to analyze and process its' contents. This is the reason why we use the OWL; it would lead us to

the right direction. This is due to the additional vocabularies along with the formal semantics comparing it to the XML, RDF and RDF Schema (Sure 2003). Owl was designed to be read by computer applications in order to have the exact meaning of the information of the web, as said before to process its' contents and to let the computers integrate information from the web easily.

3.6 Logic and Proof

These two upper layers from the Semantic Web layer (Figure 1 above) are still ongoing lab researches; they are not yet available on the web. The concept behind these layers is that it would be nice if we could state any logical principles which allow the computer to make inference and deductions (Swartz and Hendler October, 2001). After this logical principle declaration, it makes sense to prove things. It means that computers could be able to make proofs by using the logical statements (rules) stated in the logic layer.

3.7 Trust

This is the last stage of the Semantic Web layer (Figure 1 above). We have said until now that we can say almost anything about everything in the Semantic Web. So if we are allowed to do so or to use the data of the third person, then the privacy security issue seems to be in a big threat. That is the reason why the World Wide Web Consortium (W3C) is doing many ongoing researches to overcome this threat, like the development of the systems which are aware of the policies of data, interlink sets of rules, in general machines which work in the policy aware way (Sure 2003). The Trust issue is a serious issue because for the business applications this layer would be the crucial one.

4 Usage of the Semantic Web

In this section we will try to have an overlook of the possible areas where we can use the Semantic Web.

4.1 Enterprise Information Discovery and Integration

Ontology based data integration frameworks will significantly reduce integration costs. Seen from a system architecture perspective, a stronger decoupling of data and applications will become possible. Data could become an independent resource, used by several applications. Semantic Web technologies could also play a major role in the Web Service description, discovery, and composition context. Ontology based service description frameworks could push the possibilities of automatically combining services offered by different service providers another step forward (Tolksdorf, Bizer et al. 2004).

4.2 E-Commerce

The development of XML-based e-commerce standards has led to the same problems EDI initiatives ran into in the past: A wide range of competing standards has been developed and is used by different trading communities. Today suppliers often have to maintain their product catalogs in several XML formats for different market places and customers. Semantic technologies could help to solve these problems by offering a framework for standard mapping and to identify entities like products and traders across different standards. Using RDF's URI identification mechanism, the integration of different product descriptions, ratings, and offers from multiple independent sources by a shopping portal or single user agent might become possible. This would enlarge the data basis of shopping and comparison agents and enhance their usefulness (Tolksdorf, Bizer et al. 2004). Seen from the market perspective, this could lead to an increase in market transparency and make the business models of a range of trading intermediaries and market places obsolete.

4.3 Knowledge Management

Ontology based search engines will offer a direct access to information sources in corporate intranets and on the global Web, which will reduce search costs. In addition, adaptive Web sites could enable a dynamic reconfiguration according to user profiles, and make precisely the information needed available in the personally desired format and on the preferred device. Because semantic knowledge networks are based on language independent ontological concepts, it could be even possible to render a large amount of Web content in the user's preferred language (Tolksdorf, Bizer et al. 2004).

Conclusion

The aim of a Semantic Web is to permit computer programs to aid humans by doing semantic analysis for them on a very large information sets and improve the user experience on data mining. When we are saying that the programs aid humans (users) doesn't mean that computer programs substitute them rather they automatically search the web for them, simulating their poor semantic analysis capability by creating a technology infrastructure that allows web content owners to semantically markup their information and create algorithmically certain ways for computer programs to perform specific semantic queries on very large datasets.

Instead of specifying rows and tables as our relational databases have been performing for decades, we can search if we start to use the Semantic Web for some text included in specific contexts (Sure 2003). And also, rather than matching keywords like the current engine design of Google does, the Semantic Web, thanks to its domain specific ontologies, the machines can now reason automatically by analyzing the contents of the web information and make suitable conclusions.

If the Semantic Web is being standardized, then finding and correlating information about everything including people will become much easier. The fact that

these information gets into the wrong hand might have undesirable effects. Therefore, the security issue must be taken seriously as the Semantic Web development is evolving in time.

III Practical part: Ontologies for the semantic web

1 Usage of ontology

People publish ontologies and put them out on the web for the other people to reuse and learn from them. We can explore and learn even if we don't know the domain very well. The purpose is that getting people the ontology; letting them open it, make some judgment about it and extract information for their own benefit.

While never pretending to duplicate exactly the workings of the human imagination or experience, ontologies attempt to capture conceptually the rational building blocks of the mind by modeling our knowledge of reality. The whole purpose of this is to give the computer humanlike, albeit modest, thinking ability, by providing an explicit vocabulary for things, ideas, actions, relations, and approved behaviors. Ontologies with the expressive power that provides these capabilities are generally termed formal ontologies.

A formal ontology seeks to capture the essence of selected aspects of existence by stating explicitly and formulaically the concepts of the various constituents of the domain being modeled and the relationships that pertain among them. Ontologies are said to be "formal" or "formalized" when they are capable of being rendered into a computer programming language. Probably the single most famous definition of ontology is offered by (Gruber 1993) who defines ontology simply as "an explicit specification of a conceptualization." Concepts, Gruber notes, are abstract, "simplified view[s] of the world" that become the models for the objects and ideas of some part of the world as we know it. (Guarino and Giaretta 1995), emphasizing that purpose determines how these concepts are specified, note that an ontology can give only a "partial account of a conceptualization". Knowledge, after all, is in the mind of the beholder, and ontology will necessarily represent only the point of view of the ontology builder. Ontology, in short, will never be omniscient nor all-encompassing.

Ontologies define the kind of things that exist in the world and, possibly, in an application domain. In other words, ontology provides an explicit conceptualization

which describes semantics of data, providing a shared and common understanding of a domain (Lytras and Naeve 2006).

One of the first steps in ontology creation is choosing domains and categories, which allow the correct representation. In particular, philosophers have tried to define very general and universal categories, which are supposedly able to describe the real world. The main idea is to develop an understandable, complete, and sharable system of categories, labels, and relations, which represent, in an objective way, the real world (Lytras and Naeve 2006).

Ontologies are used to allow communication among people and heterogeneous and widely spread application systems. They are implied in projects as a conceptual model, to enable a content-base access on corporate knowledge memories, knowledge bases, and archives. They allow agents to understand each other when they need to interact, communicate, and negotiate meanings; and refer to a common piece of information and share common understanding of the information structure (Lytras and Naeve 2006).

Ontologies are generally high level (general constraints) and highly declarative, working at the conceptual level and they have high expressivity in a certain sense. They are computational artifact, so that after a little manipulation we expect to run and see something as a result.

1.1 Good ontology outcome

In order to have a good usage out of an ontology what we need to do is to make a proper surveying; opening and analyzing, in other words navigating our ways through. Things about the ontology as a whole are very large structure and have a huge expressivity. So finding the interesting parts out of the ontology in general is not an easy task.

We need to understand axioms which are the logical statements (assertions). Understanding axioms is as if understanding ways of saying the same thing in

different ways. And the purpose of axioms is to express machine readable meanings (Parsia 2006).

We also need to understand classes, how they relate to other classes and how they relate to their members or other members from another class. Classes in the ontology are characterized with axioms

While surveying, making ontology classes unsatisfiable and trying to debug these unsatisfied classes helps us to understand how it works. What matters is seeing the relationship between classes, axioms and how they relate and so on.

Ontologies are typically domain specific and thus, in many cases, impossible to fully understand without expertise. But when the aspects are well explained expertise isn't required for learning from them.

In the coming sections we will have a look how we can survey an ontology and learn from it by navigating ourselves through each elements thanks to the hyperlinks which link almost every aspect of the ontology. In order to do this we will use "Swoop" an ontology editor and browser which has an access to reasoned services.

2 Swoop – OWL Ontology editor

Most existing ontology development toolkits provide an integrated environment to build and edit ontologies, check for errors and inconsistencies (using a reasoner), browse multiple ontologies, and share and reuse existing data by establishing mappings among different ontological entities. However, their User Interface (UI) design (look & feel) and usage style are inspired by traditional Knowledge Relationship (KR)-based paradigms, whose constrained and methodical framework have steep-learning curves, making it cumbersome to use for the average web user (Mindswap 2006).

On the other hand, consider a hypermedia inspired ontology editor that employs a web-browser metaphor for its design and usage. Such a tool would be more effective (in terms of acceptance and use) for the average web user by presenting a simpler, consistent and familiar framework for dealing with entities on the Semantic Web. Based on this hypothesis, we present the ontology editor - SWOOP, meant for rapid and easy browsing and development of OWL ontologies (Mindswap 2006).

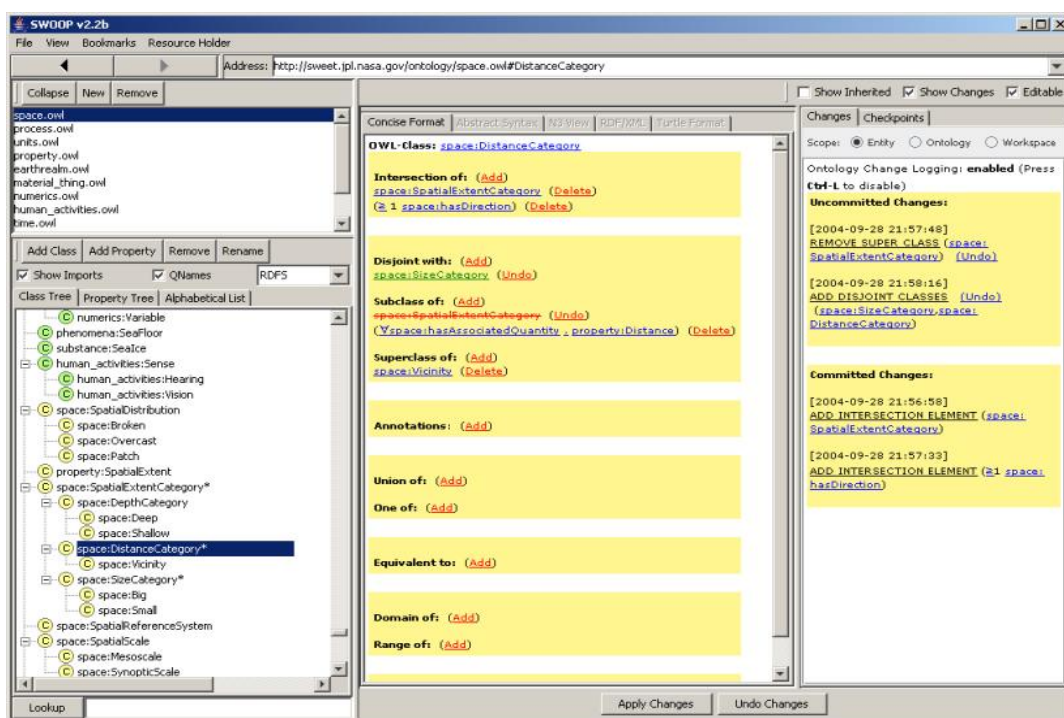


Figure 3: Swoop screenshot (Mindswap 2006)

Features (Mindswap 2006)

- Web Browser like look & feel: (see sample screenshots above)
 - An address bar where the URI of the ontology (or class/property/individual) can be entered directly for loading; hyperlink based navigation across ontological entities (address bar URL changes accordingly); history buttons (Back, Next etc) for traversal; and bookmarks that can be saved for later reference
- Inline Editing
 - All ontology editing in SWOOP is done inline with the HTML renderer, using different color codes and font styles to emphasize ontology changes, eg. different representations for added axioms vs. deleted axioms vs. inferred axioms. Undo/redo options are provided with an ontology change log and a rollback option.
- Designed for OWL Rec.
 - Species validation
 - Uses Manchester OWL API
 - Presentation syntax tabs
 - Concise Format View
 - Abstract Syntax (from OWL API)
 - RDF/XML (from BBN HyperDAML)
 - Turtle/N3
 - (plug-in architecture for new tabs)
 - Multiple Ontology Support
 - Browsing, Mapping, Comparison
- Advanced Features
 - Run "sound and complete" conjunctive ABox queries (written in RDQL) on an ontology using Pellet

- Partition Ontologies automatically by transforming them into an E-connection
- Debug mode: By exposing the internal workflow of a Description Logic Tableaux reasoner (Pellet) in a meaningful and readable manner, explanations are provided to help users understand the cause for (and remove) inconsistencies detected in ontologies.
- Collaborative Annotation Support: An Annotea plugin in SWOOP allows users to write and share annotations on any ontological entity (class/ property/ individual). Additionally, ontology change sets can be attached to the annotation messages for sharing.

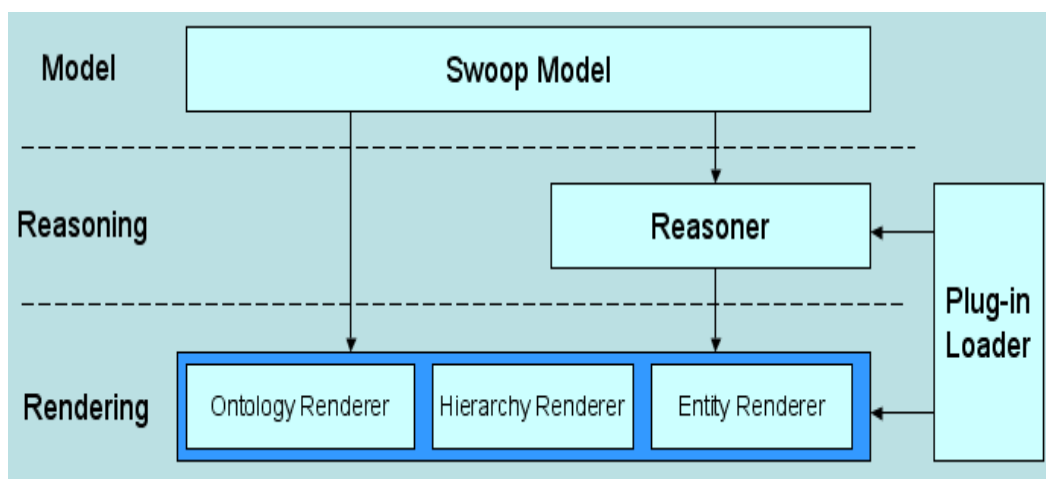


Figure 4: SWOOP Architecture (Mindswap 2006)

- Based on the Model-View-Controller (MVC) paradigm.
- Model
 - Loaded ontologies as seen by a reasoner
- Multiple Views
 - Ontology, entity, and hierarchy views
 - Support rendering + editing
- Plugin based system
 - Loads new reasoners and renderers automatically

2.1 Reasons to use the ontology tool SWOOP

At the beginning of this Bachelor thesis we had used an ontology tool which is called “Protégé”. Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies (Protégé 2007). At its core, Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats. Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data. The Protégé-OWL editor enables users to build ontologies for the Semantic Web, in particular in the W3C's Web Ontology Language (OWL). Further, Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-based tools and applications.

Beyond all these functionalities we had to abandon using it because of the fact that, on its own, Protégé does not provide a reasoning engine: it cannot check logical consistency, determine inferred types of individuals, etc. It can, however, connect to reasoning servers that use the DIG Interface protocol (over HTTP). But using the reasoning servers needs too much time for configuration.

The ontology editing tool Swoop is equipped with the most famous reasoning engine called “Pellet”. After finding this ontology toolkit, we decided to shift our ontology for the semantic web analysis from Protégé to Swoop.

As we moved from Protégé to Swoop because of the integrated reasoner, we will try to see in the coming section what a reasoner is, its different types and how they are different.

2.1 Description Logic Reasoners

We describe reasoning as the process needed for using logic. Efficiently performing this process is a prerequisite for using logic to present information in a declarative way and to construct models of reality (Cardoso 2007).

The name *description logic* refers, on the one hand, to concept descriptions used to describe a domain and, on the other hand, to the logic-based semantics which can be given by a translation into first-order predicate logic. Description logic was designed as an extension to frames and semantic networks, which were not equipped with formal logic-based semantics (Wikipedia 2007).

The description logic reasoners are increasing from day to day because of the fact that we still do not have standardized description logic reasoners. Among the reasoners we will see the description of the capabilities of some of them:

- **CEL** is the first reasoner for the description logic, Extensions of the description logics (EL+), supporting as its main reasoning task the computation of the subsumption hierarchy induced by EL+ ontologies. The most distinguishing feature of CEL is that, unlike other modern DL reasoners, it implements a polynomial-time algorithm. The supported description logic EL+ offers a selected set of expressive means that are tailored towards the formulation of medical and biological ontologies (Suntisrivaraporn 2007).
- **Cerebra Engine** is a commercial C++-based reasoner. It implements a tableau-based decision procedure for general TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, tree-conjunctive query answering using a XQuery-like syntax). It supports the OWL-API and comes with numerous other features (Sattler 2006).
- **FaCT++** is the new generation of the well-known FaCT OWL-DL reasoner. FaCT++ uses the established FaCT algorithms, but with a different internal architecture. Additionally, FaCT++ is implemented using C++ in order to

create a more efficient software tool, and to maximise portability (Tsarkov and Horrocks 2007).

- **fuzzyDL** is a Description Logic Reasoner supporting Fuzzy Logic reasoning. The fuzzyDL system includes a reasoner for fuzzy SHIF with concrete fuzzy concepts (ALC augmented with transitive roles, a role hierarchy, inverse roles, functional roles, and explicit definition of fuzzy sets). fuzzyDL's most interesting features are; it extends the classical Description Logic SHIF to the fuzzy case, it allows the explicit definition of fuzzy concepts with left-shoulder, right-shoulder, triangular and trapezoidal membership functions, it supports concept modifiers in terms of linear hedges, it supports General Inclusion Axioms, it supports "Zadeh semantics" and Lukasiewicz Logic, it is backward compatible, i.e. it supports classical description logic reasoning (Straccia 2007).
- **KAON2** is a free (free for non-commercial usage) Java reasoner for SHIQ extended with the DL-safe fragment of Semantic Web Rule Language (SWRL). It implements a resolution-based decision procedure for general TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, conjunctive query answering). It comes with its own, Java-based interface, and supports the DIG-API (Motik 2005).
- **MSPASS** is a free open-source C reasoner for numerous description logics. It implements a resolution-based decision procedure for extensions of ALB (which is ALC with inverse and Boolean operators on roles) with general TBoxes (satisfiability, subsumption) and ABoxes (instance checking, retrieval). It is an extension of the theorem prover SPASS, and can thus also be used to reason about arbitrary first-order statements (Schmidt 2007).
- **Pellet** is an open source, OWL DL reasoner in Java that is developed, and commercially supported. Based on the tableaux algorithms for expressive

Description Logics (DL), Pellet supports the full expressivity of OWL DL, including reasoning about nominals (enumerated classes). As of version 1.4, Pellet supports all the features proposed in OWL 1.1, with the exception of n-ary datatypes. Pellet provides standard and cutting-edge reasoning services. It also incorporates various optimization techniques described in the DL literature and contains several novel optimizations for nominals, conjunctive query answering, and incremental reasoning. And it is the reasoner that the ontology tool Swoop that we use for this Bachelor thesis uses for its reasoning purpose (Parsia 2006).

- **QuOnto** (Querying Ontologies) is a free (for non-commercial use) Java-based reasoner for DL-lite with GCI. It implements a query rewriting algorithm for both consistency checking and query answering for unions of conjunctive queries over DL-Lite knowledge bases, whose ABox is managed through relational database technology. It comes with its own Java-based interface (Poggi and Ruzzi 2007).
- **RacerPro** is a commercial (free trials and research licenses are available) lisp-based reasoner for SHIQ with simple datatypes (i.e., for OWL-DL with qualified number restrictions, but without nominals). It implements a tableau-based decision procedure for general TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, nRQL query answering). It supports the OWL-API and the DIG-API and comes with numerous other features (Co. 2007).

.
. .
.

The statements that the ontology models see depend on both the asserted statements in the underlying RDF graph, and the statements that can be inferred by the reasoner being used in our case the “Pelette reasoner” (see Figure 5 below).

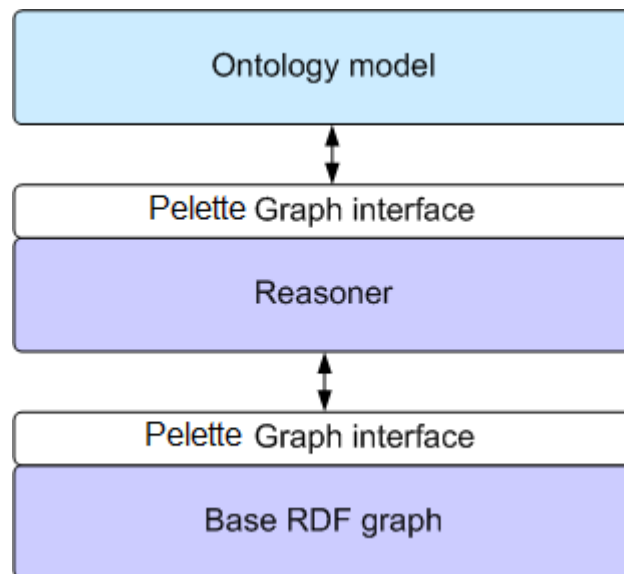


Figure 5: The statements seen by the OntModel (cf. [Jena 2007])

The asserted statements are held in the base graph. This presents a simple internal interface, Graph. The reasoner, or inference engine, can use the contents of the base graph and the semantic rules of the language, to show a more complete set of statements - i.e. including those that are *entailed* by the base assertions. This is also presented via the Graph interface, so the model works only with that interface. This allows us to build models with no reasoner, or with one of a variety of different reasoners, without changing the ontology model (Jena 2007). It also means that the base graph can be an in-memory store, a database-backed persistent store, or some other storage structure altogether without affecting the ontology model.

The reasoning we do in the coming sections deals with organizing Classes, Individuals and Properties. The associated logic is called description logic (DL) as we saw at the beginning of this section, which has become popular as the basis for the Web Ontology Language (OWL) which for our purposes can be considered as an extension of RDF.

3 Navigating an ontology using Swoop

Swoop is like a web-browser it means that we have the address bar at the top where we can put or paste an ontology URL so that it can download the ontology and display on its main page (see Figure 6 below).

We need to download an ontology either from the bookmarks or by typing a URL on the address bar in order to analyze the ontology and try to make a use out of it. For this paper we will download an ontology which is called “koala.owl” from the bookmarks because we thought that it is somehow easy to analyze and understand its concepts.

Understanding what is going on in the existing ontology helps us to figure out what information we need (for example) to get it better. So the major emphasis of Swoop is getting people comfortable with opening an ontology, looking at it and making some sort of judgment about it and of course extracts some information from it for their own benefit. The very advantage of ontology is that nowadays people publish their ontologies on the web and we can make a use out of it by using ontology tools like in this case Swoop.

The screenshot shows the Swoop 2.3beta4 interface. The address bar contains the URL `http://protege.stanford.edu/plugins/owl/owl-library/koala.owl`. The main display area shows the following statistics:

OWL Ontology: koala.owl

Annotations:

- Total Number of Classes: 20 (Defined: 20, Imported: 0)
- Total Number of Datatype Properties: 1 (Defined: 1, Imported: 0)
- Total Number of Object Properties: 4 (Defined: 4, Imported: 0)
- Total Number of Annotation Properties: 2 (Defined: 2, Imported: 0)
- Total Number of Individuals: 6 (Defined: 6, Imported: 0)

Advanced Ontology Statistics:

General Statistics	Property Tree Statistics	Satisfiable Class Tree Statistics
DL Expressivity: <u>ALCON(D)</u>		Classes with Multiple Inheritance: <u>0</u>
No. of GCI's: <u>0</u>		Max. Depth of Class Tree: <u>3</u>
No. of Sub-classes: <u>11</u>		Min. Depth of Class Tree: <u>1</u>
No. of Disjoint Axioms: <u>1</u>		Avg. Depth of Class Tree: <u>1.9</u>
No. of Functional Properties: <u>2</u>	Properties with Multiple Inheritance: <u>0</u>	Max. Branching Factor of Class Tree: <u>0</u>
No. of Inverse Functional Properties: <u>0</u>	Max. Depth of Property Tree: <u>0</u>	Min. Branching Factor of Class Tree: <u>1</u>
No. of Transitive Properties: <u>0</u>	Min. Depth of Property Tree: <u>0</u>	Avg. Branching Factor of Class Tree: <u>3.3</u>
No. of Symmetric Properties: <u>0</u>	Avg. Depth of Property Tree: <u>0.0</u>	
No. of Inverse Properties: <u>0</u>		

Figure 6: Swoop's overview

The advanced ontology statistics table which is found in the main frame of Swoop (see Figure 6 above) shows us the most precise account of what is going on in the expressivity of the ontology. In the first column, we find “General Statistics” of classes and properties with the DL expressivity of ALCON(D) and if we want to know the meaning of ALCON(D), all we need to do is click the link to see its definition (see Figure 7 below).

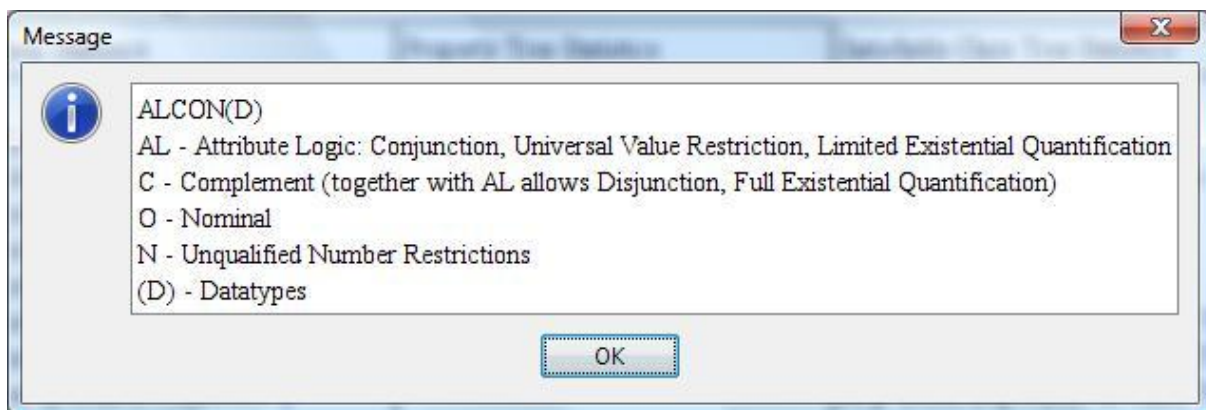


Figure 7: Definition of ALCON(D)

As we can see, the basic description logic is the Attribute Logic (see Figure 7 above) which contains the conjunction, universal value restriction and limited existential qualification. Complement or the negation is the core of very expressive description logic because the Description Logic that contains ALC is then the opposition logic. Nominal and unqualified numbers of restrictions of the datatypes are part of it as well.

At first let's try to see the general structure of the downloaded ontology in our case “koala.owl” from the bookmarks. So what we will do is switching on and off the reasoner which is in our case “Pelette” to see what difference it gives with and without the reasoner (see Figure 8 below).



Figure 8: List of Reasoners

In the first glance (with no Reasoner) we do not have a Multiple Inheritance (see Figure 9 below); Multiple Inheritance refers in which a class can inherit behaviors and features from more than one superclass. This contrasts with single inheritance, where a class may inherit from only one superclass (Wikipedia 2006).

Advanced Ontology Statistics:

General Statistics	Property Tree Statistics	Satisfiable Class Tree Statistics
DL Expressivity: <u>ALCON(D)</u> No. of GCI's: 0 No. of Sub-classes: 11 No. of Disjoint Axioms: <u>1</u> No. of Functional Properties: <u>2</u> No. of Inverse Functional Properties: <u>0</u> No. of Transitive Properties: <u>0</u> No. of Symmetric Properties: <u>0</u> No. of Inverse Properties: <u>0</u>	Properties with Multiple Inheritance: <u>0</u> Max. Depth of Property Tree: <u>0</u> Min. Depth of Property Tree: <u>0</u> Avg. Depth of Property Tree: 0.0	<u>Classes with Multiple Inheritance: 0</u> Max. Depth of Class Tree: <u>3</u> Min. Depth of Class Tree: <u>1</u> Avg. Depth of Class Tree: 1.9 Max. Branching Factor of Class Tree: <u>0</u> Min. Branching Factor of Class Tree: <u>1</u> Avg. Branching Factor of Class Tree: 3.3

Figure 9: Multiple Inheritance with no Reasoner

By putting a reasoner (inference strategy) here in our case “Pelette”, we can see that we now have one Multiple Inheritance (see Figure 10 below) instead of zero Multiple Inheritance in the above case (see Figure 9 above). Computing and maintaining the Multiple Inheritance is the job of the reasoner. The technique is called Ontology Normalization.

Advanced Ontology Statistics:

General Statistics	Property Tree Statistics	Satisfiable Class Tree Statistics
No. of Unsatisfiable Classes: 3 DL Expressivity: ALCON(D) No. of <i>GCI</i> s: 0 No. of <i>Sub-classes</i> : 18 No. of <i>Disjoint Axioms</i> : 1 No. of <i>Functional Properties</i> : 2 No. of <i>Inverse Functional Properties</i> : 0 No. of <i>Transitive Properties</i> : 0 No. of <i>Symmetric Properties</i> : 0 No. of <i>Inverse Properties</i> : 0	Properties with <i>Multiple Inheritance</i> : 0 <i>Max. Depth</i> of Property Tree: 0 <i>Min. Depth</i> of Property Tree: 0 <i>Avg. Depth</i> of Property Tree: 0.0	Classes with Multiple Inheritance: 1 <i>Max. Depth</i> of Class Tree: 4 <i>Min. Depth</i> of Class Tree: 1 <i>Avg. Depth</i> of Class Tree: 2.6 <i>Max. Branching Factor</i> of Class Tree: 5 <i>Min. Branching Factor</i> of Class Tree: 1 <i>Avg. Branching Factor</i> of Class Tree: 2.2

Figure 10: Multiple Inheritance with the reasoner “Pelette”

We can look the classes involved in the multiple inheritance by simply clicking the number aside (Classes with Multiple Inheritance) and see the classes who share the many superclasses. In our example it is the class “MaleStudentWith3Daughters”. Let’s see the Natural Language and the Concise Format tab to have a clear picture of the superclasses of the Multiple Inheritance class (see Figure 11& Figure 12 below).

Concise Format	Abstract Syntax	Natural Language	RDF/XML	Turtle
Definition: (Necessary and Sufficient Conditions) If a MaleStudentWith3Daughters has a children, then that children: - is a Female a MaleStudentWith3Daughters is a Student and is an Animal that: -- has exactly 3 children -- has male gender				

Figure 11: Natural Language tab view

We can also navigate through the classes found by clicking the link thanks to the hyper media heavy interface because there are hyperlinks everywhere which permit us to move around freely.

Concise Format Abstract Syntax RDF/XML Turtle

OWL-Class: MaleStudentWith3Daughters

Intersection of:

- $(\exists \text{hasGender} . \{ \text{male} \})$
- $(\forall \text{hasChildren} . \text{Female})$
- $(= 3 \text{ hasChildren})$
- Student

Subclass of:

- Male (Why?)
- Parent (Why?)
- Student (Why?)

Axiom of the Multiple Inheritance Class

Superclasses of the Multiple Inheritance Class

Figure 12: Concise Format for the Multiple Inheritance Class

The Axiom and the RDF/XML view of the class Multiple Inheritance (see Figure 13 below) allow us to see and analyze how the axiom and the RDF/XML code are related.

<rdf:RDF xml:base="http://protege.stanford.edu/plugins/owl/owl-library/koala.owl" xmlns:koala="skola:" xmlns:owl="owl:"

Axiom for MaleStudentWith3Daughters

$(\text{MaleStudentWith3Daughters} \sqsubseteq ((\exists \text{hasGender} . \{ \text{male} \}) \sqcap (\forall \text{hasChildren} . \text{Female}) \sqcap (= 3 \text{ hasChildren}) \sqcap \text{Student}))$

```

<owl:Class rdf:about="#MaleStudentWith3Daughters">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Student"/>
    <owl:Restriction>
      <owl:hasValue>
        <koala:Gender rdf:about="#male"/>
      </owl:hasValue>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasGender"/>
      </owl:onProperty>
    </owl:Restriction>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:Class rdf:about="#Female"/>
      </owl:allValuesFrom>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasChildren"/>
      </owl:onProperty>
    </owl:Restriction>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="xsd:nonNegativeInteger">3</owl:cardinality>
      <owl:onProperty rdf:resource="#hasChildren"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
<owl:Class rdf:about="#Gender"/>
</rdf:RDF>

```

Figure 13: Axiom vs. RDF/XML (Stanford 2002)

And by clicking the link “Why?” just aside the superclasses of the Multiple Inheritance class (see Figure 12 above) we can see the axiom of the chosen superclass comparing to its subclass in our case the class “MaleStudentWith3Daughters” (see Figure 14 below)



Figure 14: Axioms of superclass “Male” vs. subclass “MaleStudentWith3Daughters”

In order to analyze the redundant or irrelevant parts of the axioms while comparing the super and subclasses, we check the box “Strike out irrelevant parts of axioms” and it overlines those parts which are irrelevant for the reasoner with red line (see Figure 15 below).

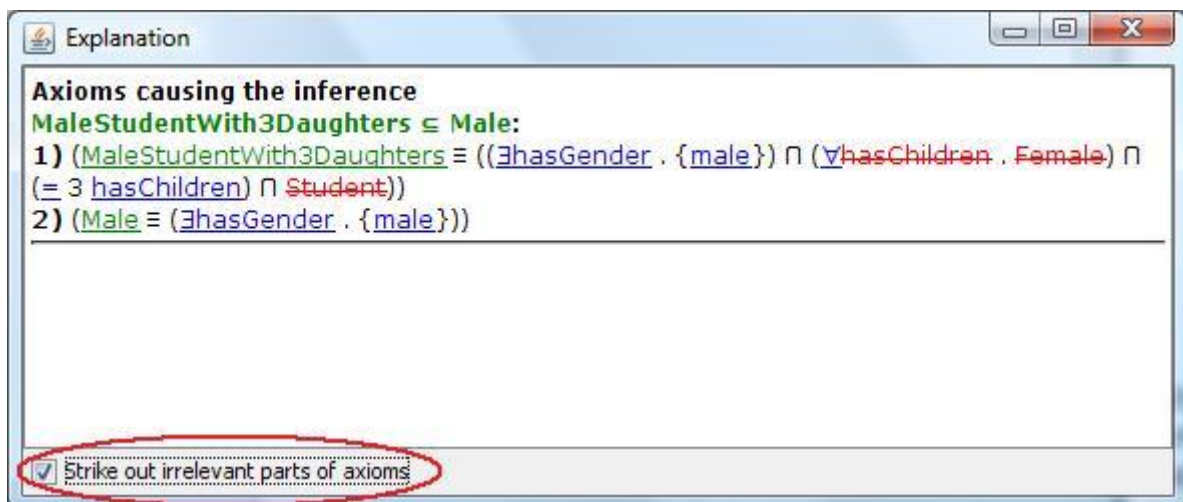


Figure 15: Irrelevant parts of axioms for the super and subclasses

The number of Disjoint Axioms in the first column of the Advanced Ontology Statistics of the main frame of Swoop (see Figure 16 below) shows the possibility of the unsatisfiability or inconsistency of the classes due to some negations. Because if we do not have negations we cannot get a contradiction, so in our case it is essential. If we see no number of restrictions or no disjoint axioms, it means that we have a trivially uncontradictory ontology or we have not ruled out anything essential or say something interesting by using negation and this is a very interesting thing to recognize. It is impossible to rule out anything in the RDF and RDF Schema. This turns to inference, if we compare inference and validation in the XML Schema, it is easy to get a contradiction between the instances and the Schema because it is making a very strong assumption about what we mean when we write something, because here in our example we are being very liberal, so it is quite hard.

Advanced Ontology Statistics:

General Statistics	Property Tree Statistics	Satisfiable Class Tree Statistics
DL Expressivity: ALCON(D) No. of GCI's: 0 No. of Sub-classes: 11 No. of Disjoint Axioms: <u>1</u> No. of Functional Properties: <u>2</u> No. of Inverse Functional Properties: <u>0</u> No. of Transitive Properties: <u>0</u> No. of Symmetric Properties: <u>0</u> No. of Inverse Properties: <u>0</u>	Properties with Multiple Inheritance: <u>0</u> Max. Depth of Property Tree: <u>0</u> Min. Depth of Property Tree: <u>0</u> Avg. Depth of Property Tree: 0.0	Classes with Multiple Inheritance: <u>0</u> Max. Depth of Class Tree: <u>3</u> Min. Depth of Class Tree: <u>1</u> Avg. Depth of Class Tree: 1.9 Max. Branching Factor of Class Tree: <u>9</u> Min. Branching Factor of Class Tree: <u>1</u> Avg. Branching Factor of Class Tree: 3.3

Figure 16: Disjoint Axioms

When we turn on the reasoning we see that the “Max. Branching Factor of Class Tree” in the third column of the Advanced Ontology Statistics table changes from 9 to 5 (see Figure 17 below). In the top level the branch classes that have no explicitly asserted super classes moved and that is a good thing. Because moving means that our definition is what is structuring our classes and what is written about the classes describe its numbers that is what is moving not the specific certain relationships between classes.

Advanced Ontology Statistics:

General Statistics	Property Tree Statistics	Satisfiable Class Tree Statistics
No. of Unsatisfiable Classes: 3 DL Expressivity: <u>ALCON(D)</u> No. of <i>GCI</i> s: 0 No. of <i>Sub-classes</i> : 18 No. of <i>Disjoint Axioms</i> : 1 No. of <i>Functional Properties</i> : 2 No. of <i>Inverse Functional Properties</i> : 0 No. of <i>Transitive Properties</i> : 0 No. of <i>Symmetric Properties</i> : 0 No. of <i>Inverse Properties</i> : 0	Properties with <i>Multiple Inheritance</i> : 0 <i>Max. Depth</i> of Property Tree: 0 <i>Min. Depth</i> of Property Tree: 0 <i>Avg. Depth</i> of Property Tree: 0.0	Classes with <i>Multiple Inheritance</i> : 1 <i>Max. Depth</i> of Class Tree: 4 <i>Min. Depth</i> of Class Tree: 1 <i>Avg. Depth</i> of Class Tree: 2.6 <i>Max. Branching Factor</i> of Class Tree: 5 <i>Min. Branching Factor</i> of Class Tree: 1 <i>Avg. Branching Factor</i> of Class Tree: 2.2

Figure 17: Max. Branching Factor of Class Tree with the reasoned switched on

If we fix certain relations between classes then we lose the opportunity to discover their relationships even if we got it right to emerge from the reasoner discovering them. Therefore, we should get the subclass relationships in order to make the reasoner do its job and that is the goal. Analyzing the ontology until now is a bit abstract, so let's see the visualization.

To see the more grained or detailed view of the ontology structure, the ontology editor and browser Swoop gives us the opportunity to visualize how the ontology structure could look like. By clicking the advanced menu where we find the "Fly The MotherShip" list, we can see the ontology with its asserted subclass tree relationship (see Figure 18 below). With no Reasoner as we saw it before, there are no asserted subclasses. There are two possibilities for this effect; the ontology has no description in it what so ever or the ontology describes the fine things without making too many assertions about the relations between classes.

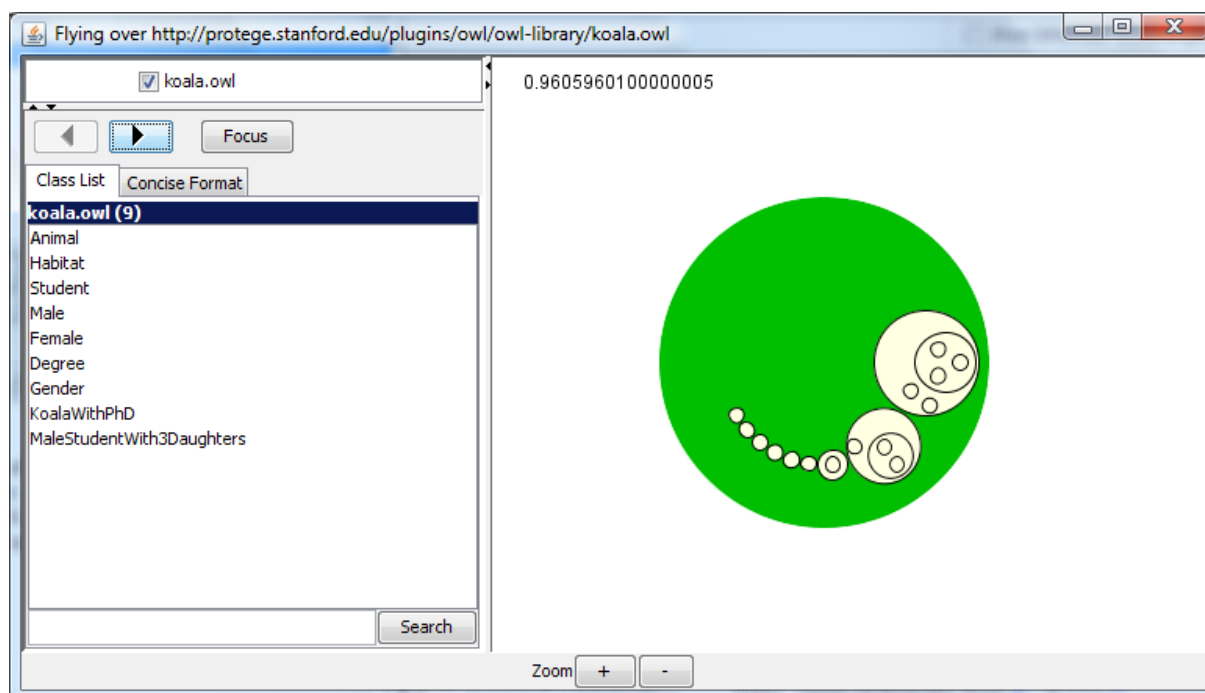


Figure 18: Visualization with no Reasoner

Each circle represents a class, which can be a subclass (child) of another class and so on as we can see in the Figure 18 above. It seems as if we see a smiley face, a sign of “everything is fine” of course we have not switched on the reasoner, without reasoner things (concepts) seem to be working in harmony. But to see whether what we see here is true with the reasoner or not, we need to switch back on our famous reasoner “Pelette”, and see what happens.

By collapsing all the classes in the normal view we can have the same effect as if we are on the visualization view (see Figure 19 below) with no Reasoner. But the advantage of visualization (task specification) is that we can see the sub structure immediately without having to open the classes.

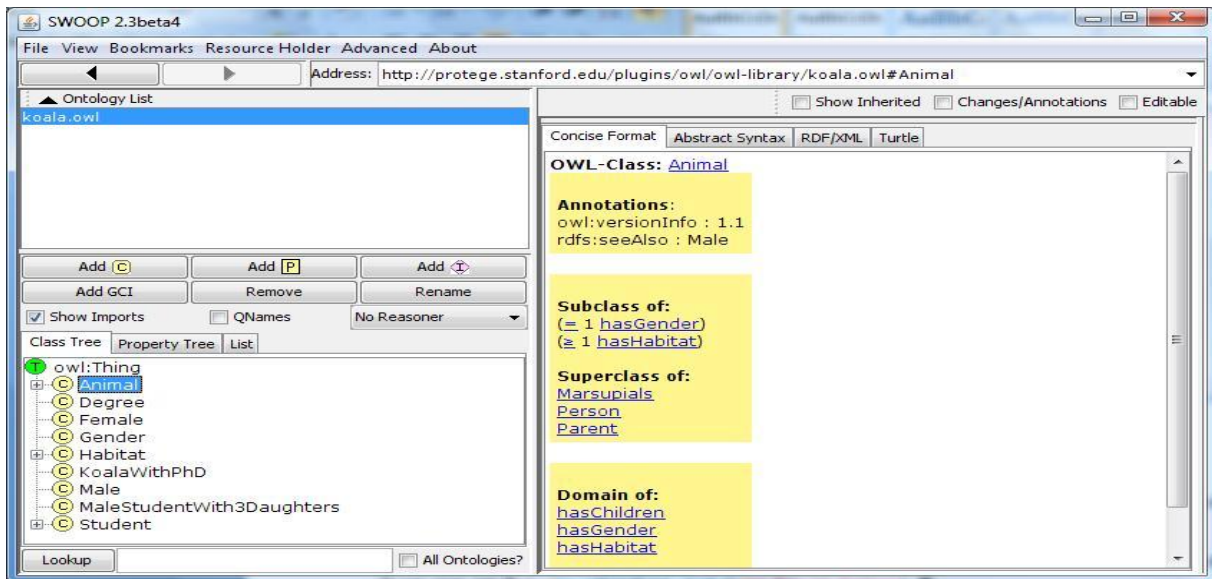


Figure 19: Collapsed class view

When we turn on the reasoner, the visualization collapsed (see Figure 20 below) in the similar was as we mentioned in the “Max. Branching Factory Class Tree” which was shrinking from 9 children to 5 children. So that the difference, that means the four classes had to move somewhere, they couldn't disappear from nature by simply switching the reasoner. We can visualize this effect thanks to visualization. This time with the reasoning switched on the visualization looks different to the Figure 18 above with no Reasoner.

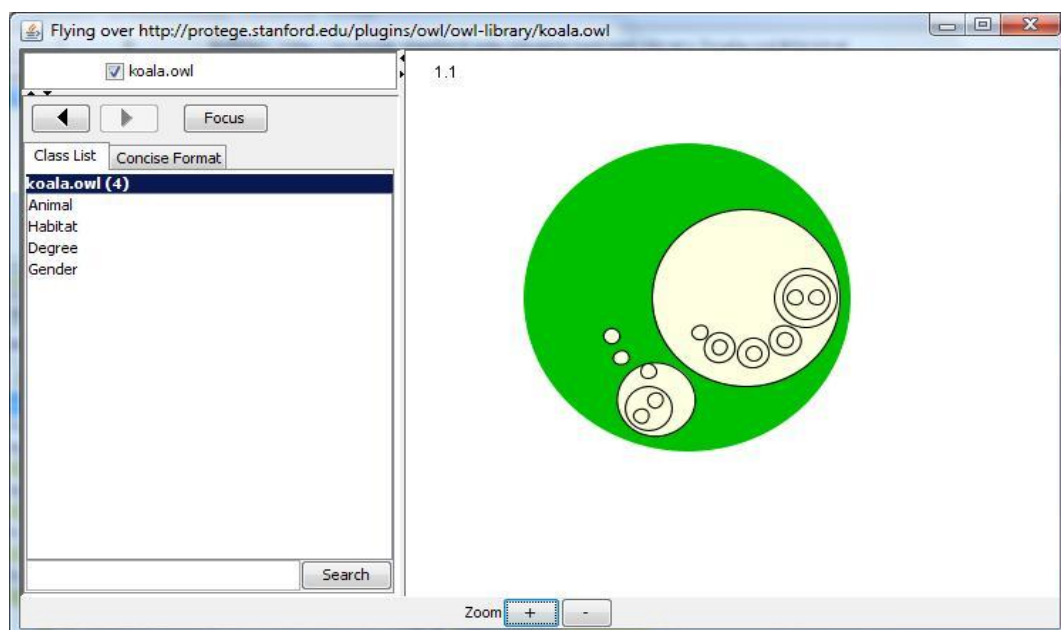


Figure 20: Visualization with reasoner switched on

Switching the reasoning permits the reasoner go grab all the means in the ontology for all the classes and asks each pair in each direction “is this class a subclass of that class?” and tries to figure that out. As we saw in the Figure 20 above with reasoning switched on, not only the classes move in the visualization view but the substructure has changes as well but the general structure remains the same. As we saw in both views above the dominant class was the class “Animal”.

We can also see the Multiple Inheritance that we have seen above by using the visualization view (see Figure 21 below) and clicking one of the classes of the “MaleStudentWith3Daughters”

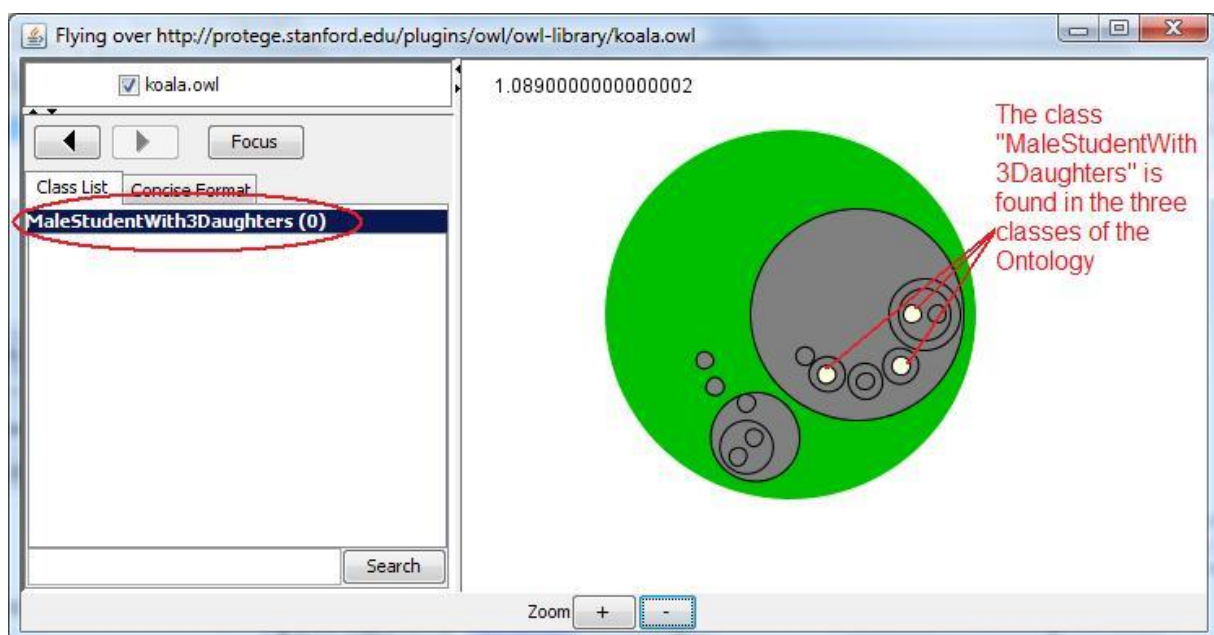


Figure 21: Visualizing the Multiple Inheritance

We can easily observe the parent classes of the class “MaleStudentWith3Daughters” when we overpass the small white dots we see in Figure 21 above which signify the class “MaleStudentWith3Daughters”. The circles just above them are the superclasses; here in our example the superclasses are the class “Student”, the class “Male” and the class “Parent”.

If we want to have a global view of our downloaded ontology we can observe the “Species Validation” the second tab of the main frame of the browser/editor Swoop (see Figure 22 below).

Ontology Info Species Validation

Level: OWL-DL

OWL-DL features:

- Disjoint Classes axiom found: DisjointClasses([Marsupials](#) [Person](#))
- Individual Value: restriction([hasGender](#) value ([male](#)))
- Object restriction with non classID filler: restriction([hasDegree](#) someValuesFrom([oneOf](#)([BA](#) [BS](#))))
- Enumeration: [oneOf](#)([BA](#) [BS](#))
- Individual Value: restriction([hasGender](#) value ([female](#)))
- Data Value: restriction([isHardWorking](#) value ("false"^{^^}<[http://www.w3.org/2001/XMLSchema#boolean](#)>))
- Individual Value: restriction([hasDegree](#) value ([PhD](#)))
- Individual Value: restriction([hasGender](#) value ([male](#)))
- Cardinality with > 1: restriction([hasChildren](#) cardinality(3))
- Data Value: restriction([isHardWorking](#) value ("true"^{^^}<[http://www.w3.org/2001/XMLSchema#boolean](#)>))
- Data Value: restriction([isHardWorking](#) value ("true"^{^^}<[http://www.w3.org/2001/XMLSchema#boolean](#)>))

Figure 22: Species Validation view of the downloaded ontology

The Figure 22 above allows us to see the classes, properties as well the individuals. In general, we can see all the features of the OWL-Description Logic which exist in the case of the downloaded ontology.

The relation that the classes have for example in the case of the Disjoint Class axioms that we saw before, in the Figure 23 below, we can see which classes are Disjoint Classes, in our example the class “Marsupials” and the class “Person” are disjoint. If we want to go further and see how these classes are structured all we need is click the link which takes us to the class structure of those classes.



Figure 23: Link to the structure of the Class

If we still want to go further and find out for example the Disjoint classes or Subclasses or Superclasses of this class (see Figure 23 above) all we need is click the link as we did before.

SWOOP 2.3beta4

File View Bookmarks Resource Holder Advanced About

Address: <http://protege.stanford.edu/plugins/owl/owl-library/koala.owl>

Ontology List

- Economy.owl
- food
- GH5Complete.owl
- IEDMv1.0.owl
- koala.owl**
- model
- pizza_20041007.owl
- protege
- teams
- university.owl
- wine

List of Ontologies loaded to the tool Swoop

Ontology Info Species Validation

OWL Ontology: [koala.owl](#)

Annotations:

Total Number of Classes: 20 (Defined: 20, Imported: 0)
Total Number of Datatype Properties: 1 (Defined: 1, Imported: 0)
Total Number of Object Properties: 4 (Defined: 4, Imported: 0)
Total Number of Annotation Properties: 2 (Defined: 2, Imported: 0)
Total Number of Individuals: 6 (Defined: 6, Imported: 0)

Advanced Ontology Statistics:

General Statistics	Property Tree Statistics	Satisfiable Class Tree Statistics
DL Expressivity: ALCON(D)		Classes with <i>Multiple Inheritance</i> : 0
No. of <i>GCI</i> s: 0	Properties with <i>Multiple Inheritance</i> : 0	<i>Max. Depth</i> of Class Tree: 3
No. of <i>Sub-classes</i> : 11	<i>Max. Depth</i> of Property Tree: ?	<i>Min. Depth</i> of Class Tree: 1
No. of <i>Disjoint Axioms</i> : 1	<i>Min. Depth</i> of Property Tree: ?	<i>Avg. Depth</i> of Class Tree: 1.9
No. of <i>Functional Properties</i> : 2	<i>Avg. Depth</i> of Property Tree: ?	<i>Max. Branching Factor</i> of Class Tree: 9
No. of <i>Inverse Functional Properties</i> : 0		<i>Min. Branching Factor</i> of Class Tree: 1
No. of <i>Transitive Properties</i> : 0		<i>Avg. Branching Factor</i> of Class Tree: 3.3
No. of <i>Symmetric Properties</i> : 0		
No. of <i>Inverse Properties</i> : 0		

Figure 24: List of Ontologies loaded

We can also load many ontologies at the same time and survey, analyze, move around them to extract the good use out of it (see Figure 24 above).

4 SWOT analysis: Ontologies for the Semantic Web

The following sections represent a number of Strengths, Weaknesses, Opportunities, and Threats (SWOTs) (Wikipedia 2007) identified as we considered the past, present and future of Ontologies for the Semantic Web.

4.1 Strengths

- W3C has an excellent reputation for creating useful standards (HTML, XML, XML Schema etc). Therefore, the fact that it revealed in 2004 as standards the Resource Definition Framework (RDF) and the OWL Web Ontology Language (OWL) as part of its plan for the "Semantic Web" plays a big role (Boulton 2004).
- Widespread acceptance in academic institutions worldwide. While the work on standards for representing semantics in Web services is evolving as mentioned in the previous point, both the academic and industry communities worldwide have recognized the need to make a business case for articulating the value of semantics in Web services (Cardoso 2007).

4.2 Weaknesses

- Proof, logic and trust layers (Figure 1 above) still in research and development stage (McCreary 2006). While the other layers of the semantic Web stack have received a fair amount of attention, no significant research has been carried out in the context of these layers (Cardoso 2007). For further

information about these layers check section 3.6 above for logic & proof and section 3.7 above for the trust layer.

- RDF perceived as too complex or conflicting with XML (McCreary 2006). RDF is an abstract, conceptual framework for defining and using metadata, independent of any concrete implementation and syntax. However, to write RDF statements we require a concrete means of expression (Daum and Merten 2003). See section 3.2 Extensible Markup Language (XML) and 3.3 Standard Syntax – RDF above for the two Semantic Web standards.
- Perception that web sites need to be published in both human and machine readable versions doubling costs. We can have a machine readable RDF version or a human readable HTML version but not both at the same time (Davis 2007) if we decide to have both, they couldn't share the same URI what so ever and the cost obviously increases.

4.3 Opportunities

- IT departments spend billions each year on integration of Semantic Web (McCreary 2006). We could estimate that the market for products and services stemming from semantic Web technologies at \$50 billion by 2010, up from about \$7 billion today (Copeland 2007). So by seeing all this we could expect a real boom in the near future.
- Automated metadata discovery could become cost-effective. Automatic metadata (data about data) generation in the case of RDF is more efficient, less costly, and more consistent than human-oriented processing (Margaritopoulos, Manitsaris et al. 2007).
- Automated integration requires ontologies. Semantic Integration approach allows for two distinct ontologies to be merged/integrated using a set of semantically similar words. For example, if two concepts have the same set of associated words then the concepts can be said to have the same context and

can therefore said that these concepts are equivalent (Tierney and Jackson 2004).

4.4 Threats

- Incompatible and constantly changing Folksonomies or personal information (tags) with freely chosen keywords. The problem with multiple vocabularies that contain the same terms but apply different meaning to them is that we destroy the author-intended meaning of the information if we attempt to merge the information. That said, it is bad to assume binary compatibility between the meanings expressed in vocabularies(Simmons 2007). There will be a great need for an open, unified vocabulary in the Semantic Web.
- Privacy invasion stems from the issue of reduced anonymity on the Semantic Web. A Web that exposes vast amounts of information about everyone has its drawbacks. One downside to having so much information easily accessible to anyone is there will always be someone ready to abuse that information(Simmons 2007).

5 Future of Information Management with Semantic Web

The future of information management will be based on Semantic Web architecture and applications. The most important issue is which technologies and firms take the immediate leadership to drive the migration, and therefore guide the information architecture of the future (Generation 2006).

- **Migration to XML and RDF Standards:** Applications programs will follow Microsoft's migration to XML standards for the authoring and exchange. XML and RDF standards will become the dominant approach for capturing, understanding, storing and exchanging external document descriptions and document contents.
- **Universal Internet Web Portals:** Information access will migrate to web portals within organizations and with the general population; and web portals based on Semantic Web applications will become the central user application. Operating systems and legacy applications will become transparent under semantic web portals with highly flexible applications.
- **Parallel Legacy Database Integration:** Legacy databases will be extracted into parallel Semantic Web architecture databases to provide access to fragmented sources. Parallel architecture dramatically reduces the costs, risks, and schedules from the Enterprise Resource Planning (ERP) "tear down and rebuild".
- **Global and Language Expansion:** Information sources, users and entities will expand globally and support many languages. Because Semantic Web architectures and applications "learn and think" in the original language, the production and exchange of multi-language information between language domains will increase dramatically. For example interactive Japanese language sources on China in English.
- **Network Access and Distribution:** Networks will get better, faster, cheaper, wireless and distributed. Semantic Web architecture and applications will

expand to link global data sources from main frame servers, desktop workstation and laptops, to hand held PDA and cell phone.

- **Network Transaction and Capacity:** Human transactions will grow slowly; and machine transactions will grow exponentially. The migration from man to machine intelligence transactions will rapidly take over the private and public network. This rapid capacity demand will focus a major increase in network hardware investment and stimulate new value added network services.

Conclusion and possible Future Work

In the broader context of the Semantic Web, applications need to understand not just the human-readable presentation of content but the actual content – what we call the meaning. It addresses ways of representing knowledge so that machines can reason about it to make valid deductions and inferences (Leuf 2006).

This bachelor thesis has presented the state of the art on Semantic Web Languages including the most relevant ones (XML, RDF, RDFS, OWL). The first part of the paper presented a short description of each of these languages, outlining their main capabilities and strengths. In the second part, a survey of ontology thanks to the ontology management tool SWOOP.

In the first part of this bachelor thesis we saw the language XML, the World Wide Web Consortium (W3C) developed a standard for metadata called the Resource Description Framework (RDF). The goal was to add semantics defined on top of XML. As a result, RDF provided a means of describing the relationships among resources in terms of named properties and values. However, users have desired more from RDF, such as data types and a consistent expression for enumerations. As a result, W3C developed RDF Schema (RDFS), an object-oriented type system as a minimal ontology modeling language. Although RDF and RDFS are building blocks for defining a Semantic Web, together they still lack sufficient expressive power. OWL facilitates greater machine readability of Web content than that supported by XML, RDF, and RDFS by providing additional vocabulary along with a formal semantics (Alesso and Smith 2005).

An ontology defines the terms used to describe and represent concepts. Ontologies are used to describe concepts as well as their inter-relationships. Being able to standardize the definition of real-world concepts becomes very powerful as we begin to investigate knowledge that spans across multiple domains.

So bearing in mind all the advantages Ontology could bring for us in order to develop and analyze Semantic Web we tried to analyze it by using the OWL ontology editor SWOOP as a framework in the second part of this paper for automating the

analysis tasks. For each URI we found in the bookmarks of the editor or downloaded from the net, there are two types of statistics we collect. The first set contains the statistics that do not change when a reasoner processes the ontology. We call this set static statistics, and it includes, for example, number of defined classes, what ontologies are imported (if any), or which of the OWL species the document belongs to. The second set, on the other hand, changes depending on whether a reasoning service is present. This set is called the dynamic statistics. For example, the number of concepts that have more than one parent may change when reasoning is applied since new subsumption relationships can be discovered by the reasoner. Because dynamic statistics change, we collected both the told (without reasoning), and the inferred (with reasoning) dynamic statistics (Wang 2006). Our method is to load each URI into SWOOP, collect the static statistics and the told dynamic statistics, then turn on the Pellet reasoner that comes with SWOOP, and collect the inferred dynamic statistics.

By looking and exploring the statistics by altering the alternative reasoners on and off then navigating to the detailed of the classes by clicking on the links, seeing what are the causes of multiple inheritance, what are the causes of involved or not of disjoint axioms, what are the causes of the depth of the classes and so on gives us some sense of the boundaries and structures of the ontologies and gives us the reason to go around and learn about the specific domain.

We could possibly have a future work which would concentrate on the development of the agents by using the Ontology technology on different machines; on the customer side and on the supplier side. And let the agents communicate and exchange their data by defining the same vocabularies (ontologies) on both sides. Thanks to the ontologies defined, the agents could understand the message they exchange to one another. At last the extracted data from both agents could be saved in the database of each agent's machine. The database's structure of each agent might be different, what matters for their communication is them having the same ontology.

References

- Alesso, H. P. and C. F. Smith (2005). Developing Semantic Web Services, A K Peters.
- Altova. (2005). "Altova Semantic Web Tools." Retrieved 15.08.2007, from http://www.altova.com/dev_portal_semanticweb.html.
- Altova. (2006). "What is the Semantic Web? ." Retrieved 15.08.2007, from http://www.altova.com/semantic_web.html.
- Berners-Lee, T. (2000). Semantic Web Stack. <http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html>.
- Boulton, C. (2004). "W3C Wraps Up Semantic Web Standards " Retrieved 02.12.2007, from <http://www.internetnews.com/infra/article.php/3310831>.
- Cardoso, J. (2007). Semantic Web Services: Theory, Tools and Applications, IGI Publishing.
- Co., R. S. G. (2007). "RacerPro is an OWL reasoner and inference server for the Semantic Web." Retrieved 22.11.2007, from <http://www.racer-systems.com/>.
- Copeland, M. V. (2007). "What's next for the Internet." Retrieved 11.12.2007, from http://money.cnn.com/magazines/business2/business2_archive/2007/07/01/100117068/index.htm.
- Daum, B. and U. Merten (2003). System Architecture with XML, Morgan Kaufmann Publishers.
- Davis, I. (2007). "Is the Semantic Web destined to be a shadow?" Retrieved 09.12.2007, from <http://iandavis.com/blog/2007/11/is-the-semantic-web-destined-to-be-a-shadow>.
- Fensel, D., H. Lausen, et al. (2006). Enabling Semantic Web Services: The Web Service Modeling Ontology.
- Generation, N. (2006). "Semantic Web Architecture and Applications." Retrieved 01.12.2007, from http://66.102.9.104/search?q=cache:BoulAjqO5AEJ:www.oss.net/dynamaster/file_archive/040930/210eb91aa7aef064ac1f02d94142eade/Semantic%2520Web%2520Architecture%2520and%2520Applications%2520040924%2520NEW.doc+Semantic+Web+architecture+and+applications+are+the+next+generation+in+information+architecture&hl=en&ct=clnk&cd=1&gl=ch
- Geroimenko, V. (2006). Visualizing the semantic web: XML-based Internet and information visualization 2nd ed. London, Springer corp.
- Gruber, T. R. (1993). "What is an Ontology?" Retrieved 06.11.2007, from <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>.
- Guarino, N. and P. Giaretta. (1995). "Ontologies and Knowledge Bases." Retrieved 06.11.2007, from <http://www.loa-cnr.it/Papers/KBKS95.pdf>.

- hp. (2007). "Introduction to Semantic Web Technologies." Retrieved 01.09.2007, from <http://www.hpl.hp.com/semweb/sw-technology.htm>.
- Jena. (2007). "Jena 2 Ontology API." Retrieved 24.11.2007, from <http://jena.sourceforge.net/ontology/index.html>.
- Leuf, B. (2006). The Semantic Web: Crafting Infrastructure For Agency.
- Lytras, M. D. and A. Naeve (2006). Intelligent Learning Infrastructure for Knowledge Intensive Organizations: A Semantic Web Perspective, IGI Publishing.
- Margaritopoulos, M., A. Manitsaris, et al. (2007). "On the Identification of Inference Rules for Automatic Metadata Generation." Retrieved 12.11.2007, from <http://www.mtsr.ionio.gr/proceedings/margaritopoulos.pdf>.
- Mazzocchi, S. (2000). "Toward the semantic web." Retrieved 01.09.2007, from <http://www.betaversion.org/~stefano/papers/semanticweb.pdf>.
- McCreary, D. (2006). "Promoting the Semantic Web." Retrieved 01.12.2007, from <http://66.102.9.104/search?q=cache:rzhc2INbeyoJ:www.danmccreary.com/presentations/promo-semantic-web.ppt+swot+%2B+Semantic+Web&hl=en&ct=clnk&cd=1>.
- Mindswap. (2006). "SWOOP - A Hypermedia-based Featherweight OWL Ontology Editor." Retrieved 16.11.2007, from <http://www.mindswap.org/2004/SWOOP/>.
- Motik, B. (2005). "KAON2." Retrieved 22.11.2007, from <http://kaon2.semanticweb.org/>.
- Parsia, B. (2006). Ontologies.
- Parsia, C. (2006). "Pellet." Retrieved 22.11.2007, from <http://pellet.owldl.com/>.
- Poggi, A. and M. Ruzzi. (2007). "Querying Ontologies." Retrieved 22.11.2007, from http://www.dis.uniroma1.it/~quonto/quonto_overview.htm.
- Protégé. (2007). "What is protégé?" Retrieved 20.11.2007, from <http://protege.stanford.edu/overview/>.
- Sattler, U. (2006). "DESCRIPTION LOGIC REASONERS." Retrieved 22.11.2007, from <http://www.cs.man.ac.uk/~sattler/reasoners.html>.
- Schmidt, R. A. (2007). "MSPASS." Retrieved 22.11.2007, from <http://www.cs.man.ac.uk/~schmidt/mypass/>.
- Simmons, J. (2007). "Semantic Focus." Retrieved 12.11.2007, from <http://www.semanticfocus.com/blog/entry/title/5-problems-of-the-semantic-web/>.
- Stanford. (2002). "Koala." Retrieved 18.02.2008, from <http://protege.stanford.edu/plugins/owl/owl-library/koala.owl>.
- Straccia, D. U. (2007). "The fuzzyDL System." Retrieved 22.11.2007, from <http://gaia.isti.cnr.it/~straccia/software/fuzzyDL/fuzzyDL.html>.

Suntisrivaraporn, B. (2007). "A polynomial-time Classifier for the description logic EL+." Retrieved 22.11.2007, from <http://lat.inf.tu-dresden.de/systems/cel/>.

Sure, Y. (2003). "A short Tutorial on Semantic Web; ." Retrieved 20.11.2007, from http://videlectures.net/training06_sure_stsw/.

Swartz, A. and J. Hendler. (October, 2001). "A Network of Content for the Digital City." Retrieved 16.08.2007, from <http://blogspace.com/rdf/SwartzHendler>.

Tierney, B. and M. Jackson. (2004). "Contextual Semantic Integration for Ontologies." Retrieved 11.12.2007, from <http://www.macs.hw.ac.uk/BNCOD21/DC/Tierney.pdf>.

Tolksdorf, R., C. Bizer, et al. (2004). "Business to Consumer Markets on the Semantic Web." Retrieved 10.11.2007, from http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/otm2003_Semmarkets.pdf.

Tsarkov, D. and I. Horrocks. (2007). "FaCT++." Retrieved 22.11.2007, from <http://owl.man.ac.uk/factplusplus/>.

W3C. (2001). "Semantic Web." Retrieved 15.08.2007, from <http://www.w3.org/2001/sw/>.

W3C. (2004). "Intersection, union and complement." Retrieved 08.08.2007, from <http://www.w3.org/TR/owl-ref/#intersectionOf-def>.

W3C. (2004). "OWL Web Ontology Language." Retrieved 15.11.2007, from <http://www.w3.org/TR/owl-features/>.

Wang, T. D. (2006). "Gauging Ontologies and Schemas by Numbers." Retrieved 01.12.2007, from <http://km.aifb.uni-karlsruhe.de/ws/eon2006/eon2006wang.pdf>.

Wikipedia. (2006). "Multiple Inheritance." Retrieved 25.11.2007, from http://en.wikipedia.org/wiki/Multiple_inheritance.

Wikipedia. (2007). "Description logic." Retrieved 21.11.2007, from http://en.wikipedia.org/wiki/Description_logic.

Wikipedia. (2007). "SWOT analysis." Retrieved 12.12.2007, from http://en.wikipedia.org/wiki/SWOT_analysis.

Wilshire. (2006). "Semantic Technology Primer." Retrieved 10.08.2007, from <http://www.semantic-conference.com/primer.html>.