

Pengfei Zhang

## **Semantic Web**

The Next Generation Web

# **Semantic Web**

The Next Generation Web

Pengfei Zhang

Bachelor's thesis

Spring 2012

Degree Programme in Information Technology

Oulu University of Applied Sciences

## **PREFACE**

This Bachelor's thesis was done at Oulu University of Applied Sciences during the spring term 2012.

My very special thank to the supervisor of my thesis, Mr. Veikko Tapaninen, who is lecturer of mobile related courses. He has been really patient and helpful, giving suggestions of my project and introducing thesis procedures for me. I would like to thank Mr. Jarmo Karppelin who approved my thesis topic and assigned me a great tutoring teacher. Big thanks belong to Mrs. Kaija Posio, who is responsible for the language checking.

Last but not least, I appreciate the huge support from my family and friends it kept me confident that I can finish this project quickly and efficiently.

Raahe, Finland

March, 2012

## TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu  
Tietotekniikan koulutusohjelma, Tietoteknologia

---

Tekijä: Pengfei Zhang

Opinnäytetyön nimi: Semantic Web

Työn ohjaaja: Veikko Tapaninen

Työn valmistumislukukausi ja -vuosi: Kevät, 2012 Sivumäärä: 52+10

---

Tämä opinnäytetyö esittelee seuraavan sukupolven Webteknologian Semantic Webin yhdistämällä metadataa, ontologiaa ja logiikka tehtäessä Web-toimintoja. Tällä hetkellä ei ole vielä mahdollista ratkaista kaikkia Web-ongelmia täydellisesti käyttämällä ainoastaan tekoälyä, mutta on mahdollista tunnistaa tietynlaisia attribuutteja niin, että koneilla on potentiaalia olla älykkäitä.

Rakensin pieniä näytteitä ja monimutkaisia esimerkkejä käyttämällä meta-kieltä, jota koneet voivat ymmärtää. Ohjelmointiin ja editointiin käytin Nodepad++- ja XML Notepad 2007-, XML- ja HTML-koodien lataukseen ja testaamiseen käytin Firefox 11- ja visuaaliseen esimerkkien tarkkailuun käytin RDF Gravity 1.0 -ohjelmaa.

Työn kehityksen kautta RDF-dokumentit ja ontologiamallit ovat jalostettu siihen pisteeseen, josta niitä voidaan vielä muokata ja/tai liittää suoraan verkkosivujen lähdekoodiin parantamaan tiedonjakoa ja uudelleen käyttökykyä Webissä.

---

Asiasanat: Semantic Web, Web 3.0, XML, RDF, OWL, Ontology.

## **ABSTRACT**

Oulu University of Applied Sciences  
Degree Programme in Information Technology

---

Author: Pengfei Zhang

Title of Bachelor's thesis: Semantic Web

Supervisor: Veikko Tapaninen

Term and year of completion: Spring, 2012      Number of pages: 52+10

---

The aim of this Bachelor's thesis work was to introduce the next generation of Web technology, a semantic Web utilizing metadata, ontology, and logic to carry out Web tasks. It is still not possible to resolve all Web problems completely using only artificial intelligence. However, it is important to recognize certain attributes of machines that point to their potential to have intelligence.

I built both small samples and complex examples using meta-languages that machines could understand. The implementation and testing tools included Notepad++ and XML Notepad 2007 for coding and editing, Firefox 11 to load XML and HTML codes, and RDF gravity 1.0 to give a straightforward, visual representation of the examples. The fields involved were programming, Web technology, and data processing.

During the development of this work, the RDF documents and ontology models were refined to a point where they can still be modified and/or directly included into websites' source code in order to improve the data sharing and reusability on the Web.

---

Keywords: Semantic Web, Web 3.0, XML, RDF, OWL, ontology.

# TABLE OF CONTENTS

1 INTRODUCTION	6
1.1 Limitation of the present Web	7
1.2 Future Web	8
2 THE SEMANTIC WEB	9
2.1 Nature	9
2.2 Characteristics	9
2.3 Layer Cake	11
2.4 Semantic Web in practice	13
2.4.1 Web services	14
2.4.2 Semantic application	14
2.4.3 Semantic search	16
2.4.4 Semantic Mechanism	16
3 RESOURCE DESCRIPTION FRAMEWORK	19
3.1 RDF triple	19
3.2 Elements	20
3.2.1 Syntax	20
3.2.2 Header	20
3.2.3 Namespaces	21
3.2.4 Description	22
3.2.5 Data Types	22
3.3 RDF Schema	23
3.3.1 Core Classes	23
3.3.2 Properties	24
3.4 Example: Cars	25
3.5 RDF graph	28

3.6 FOAF application	29
3.7 RDFa	30
3.8 Limitation of RDF and RDFs	32
4 WEB ONTOLOGY LANGUAGE	33
4.1 OWL offers more than RDFs	33
4.2 Three subs of OWL	34
4.3 The OWL elements	35
4.3.1 Syntax	35
4.3.2 Header	35
4.3.3 Class	36
4.3.4 Property elements	37
4.3.5 Property restrictions	38
4.4 OWL2	39
5 CAR ONTOLOGY PROTOTYPE	41
5.1 Classes and subclasses	41
5.2 Properties	43
5.3 Other elements	45
5.4 Testing	46
6 POSSIBILITIES OF FURTHER DEVELOPMENT	48
7 CONCLUSION	49
LIST OF REFERENCES	50
APPENDICES	53
1. Cars_example.xml	53
2. Cars_example.xml in tree view	54
3. General Notations	55
4. Cars_example.rdf RDF graph	56
5. RDFa.html	57

6. Car ontology source code	58
7. Car_ontology.rdf RDF graph	61



# 1 INTRODUCTION

This thesis topic proposal was suggested during my summer job at Neljämerta Oy in 2011. The main aim of this thesis is to introduce and have a preview of the next generation Web technologies, as well as, a semantic Web programming. The core contents I am explaining here are the semantic Web and its Languages. I use theoretical explanation and support it with sample source code. The end, an ontology prototype was made – a car ontology that I built with the techniques I explained in earlier chapters.

The primary audience for this document is computer science engineers as well as Web developers who want to step into the next generation Web technologies. However, the rest of the audience should be able to acquire and understand the general idea that I am trying to present here.

This topic may look boring or senseless when it is read by people who are not working in technical fields, but I suggest then just keep reading. The more they read and think, the more interesting this topic will be, because the content of this work will be used in the near future and it will change the way how we use internet and make our life and work much easier.

I recommend the reader to have certain knowledge of Web technologies or computer engineering, in order to obtain the most benefit from this work.

The modern world is experiencing the excitement of a rapid change. We may realize ourselves in the midst of an information revolution. It is widely admitted that the technology of today's information age has had a massive impact on global communications and business and that will continue to improve the work productivity. However, the World Wide Web is doing significant contributions to

this progress, yet there are still challenges of further development of Web with intelligence features.

## **1.1 Limitation of present Web**

The Web has changed from a distributed open system to a Web dominated one by portals, such as Google and MSN. While the World Wide Web Consortium (W3C) has developed Web standards, vendors and developers have customized their applications to efficient business. (Alesso H, Simith C. Thinking on Web)

Microsoft's Windows (.NET) and Sun Java (J2EE) frameworks have made the Web moving towards becoming a distributed network. The Extensive Markup Language (XML) was developed and it uses non-predefined tags, which makes XML very flexible and extensible. XML plays the role of an interoperable bridge for data exchanging.

The Web is still based on HyperText Markup Language (HTML) which makes the Web page readable for humans by describing the page layout and information display. Because in HTML, we define the head, title, body and so on, the page layout has been defined in such a way that a human can easily read it. There is no capability for machine understanding and using the Web information that it is why the major information is restricted to keyword search.

Today, we are still facing difficulties in developing complex networks with intelligent features. The problem with performing intelligent tasks like a Web service is that a human operation is needed and data is exchanged inefficiently. The limitation of present Web includes security, data exchange efficiency, automation and search.

## 1.2 Future Web

The new Web is called the semantic Web. Let us take a look at W3C's description:

“The Semantic Web is about two things. It is about common formats for integration and combination of data drawn from diverse sources, where on the original Web mainly concentrated on the interchange of documents. It is also about language for recording how the data relates to real world objects.” (W3C, W3C Semantic Web Activity, 07.11.2011)

In another words, the semantic Web is based on Resource Description Framework (RDF) that allows data to be shared and used across applications, networks and other boundaries. The semantic Web agent can utilize metadata and ontology to master its task. When an agent receives a task, it will search the information from the Web and meanwhile communicate with other Web agents in order to fulfill its task. (Breitman, K., Casanova, M., and Truszkowski, W., Semantic Web, Technologies and applications, Springer, London, 2007)

## **2 THE SEMANTIC WEB**

Web 3.0 is another way to call the semantic Web. As new websites with new features and capabilities are emerging, in some ways, these new technologies will help us do things easier or even do the things could not have been done before.

### **2.1 Nature**

The nature of Web 3.0 is to use a new technology that enable to remix, reuse and repurpose data on the Web in different ways.

Let us see some simple examples to better understand what Web 3.0 can do. What we see, today, on an ordinary web browser is the information from one website or application. Imagine that you are checking your calendar application; you can see your work or school schedules, appointments with different persons, locations of all events, related persons contacts or even pictures of events and comments all this information is gathered together from different sources and ready for you to be used. That's the remix nature of Web 3.0.

### **2.2 Characteristics**

We can understand Web 3.0 as building upon a series of attributes, and it has the following key features (Pollock. 2009. 27-29):

**Ubiquitous Networking:** The data should always be available through any channel or device and no matter what physical location of a device is.

Open: the semantic Web is all about data. Thus, for using the data, most parts of the network should remain open, i.e. open data, open devices, open Web services, open protocols and open identity.

Executable data: The main idea is the information on the Web is more connected and dynamic. It is capable of remixing and reusing on demand. We can say data is always executable when somebody needs it. The data is not owned by a single application or community.

Formatted data: In order to remix and reuse the data from anywhere on the Web, the data formats should be structured. For example, the uses of standards-based query language like SPARQL (an RDF query language) for searching in RDF databases. This new structured data is portable, reassemble and linkable.

Intelligence: It means the Web will understand the human. The semantic Web applications are so smart that they automate the way people interact with it. Intelligence and automation will be the key features of the semantic Web; it will improve the human productivity unprecedentedly.

These entire characteristics make the semantic Web to connect everything, and it uses richer semantics to enable:

- Better search
- More targeted advertisements
- Smarter collaboration
- Deeper integration
- Richer content
- Better personalization

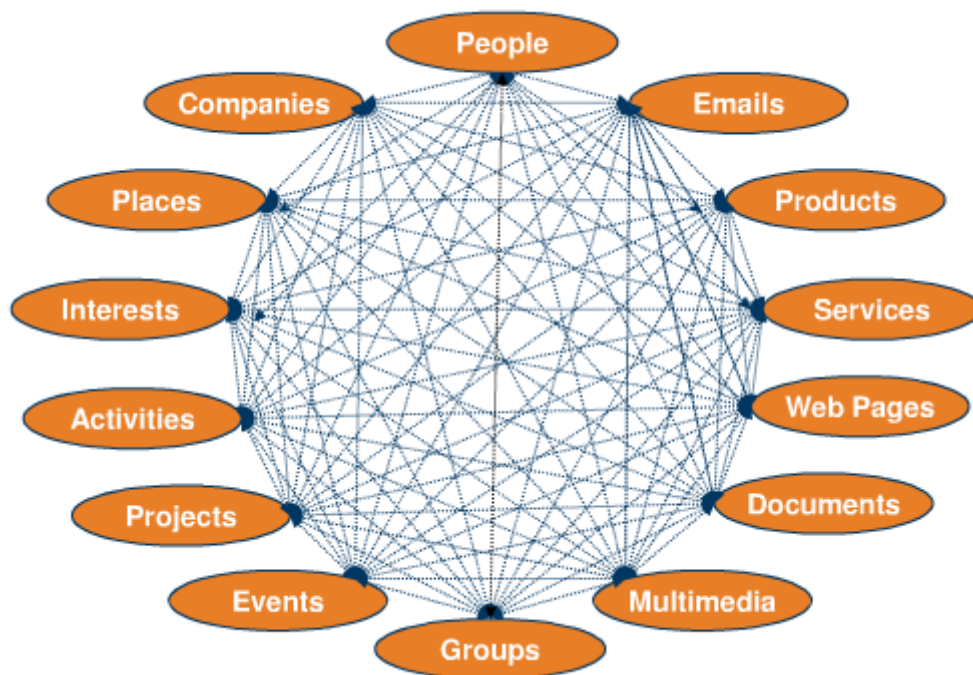


Figure 2.1 Semantic Web connectivity. (BMJ Group, Semantic publishing: how to create richer metadata)

The figure 2.1 above basically demonstrates the semantic Web's connectivity. The data from different places can be shared and reused by others. The linked open data looks like a complex network, therefore, the data and data models are fully accessible from the Web itself. I can publish some data in a model from China, and you can include it directly in your data and data model published in Finland. As long as we both have an Internet connection and use semantic Web we can make it happen.

### 2.3 Layer Cake

The Figure 2.2 below it is called a semantic Web "Layer Cake"; Many people also call it a "Markup Language Pyramid" or a "Semantic Web Stack". The current Web is built on HTML and XML, when describes how information is displayed and laid out for humans to read. The Web has developed as a

medium for humans without the functionality of an automatic data processing. As a result, the machines are unable to process the data automatically.

The semantic Web brings meaning to Web pages' content, so that software agents can carry out tasks across different pages automatically. Basically, the semantic Web can be constructed using RDF and Web Ontology Language (OWL). W3C has developed these languages and data can be defined and linked using RDF and OWL.

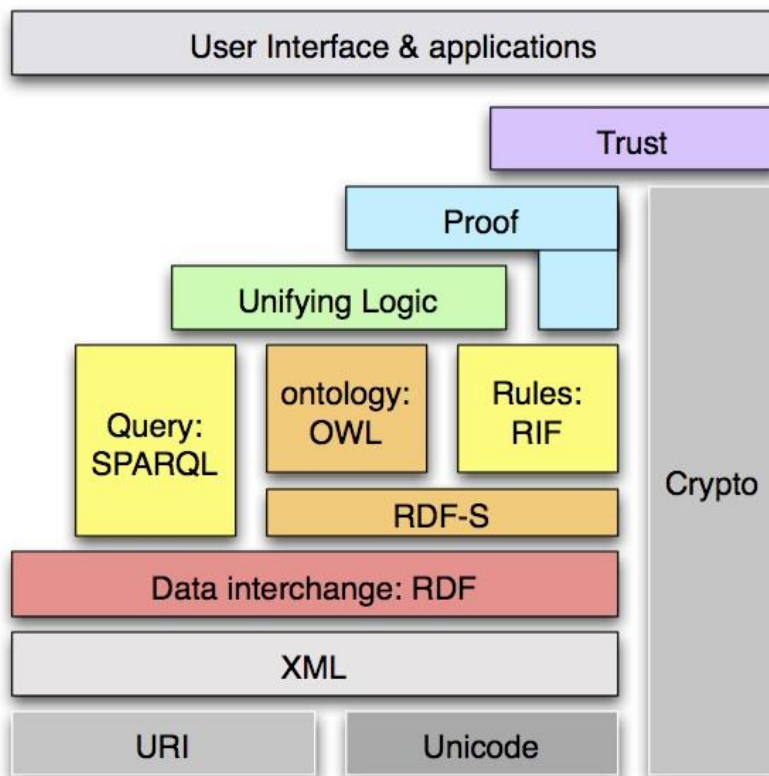


Figure 2.2 Layer cake.  
(SW Arch: Same symbols, multiple languages, 2006)

Figure 2.2 illustrates how semantic Web Languages are built upon XML and climb up the pyramid. These languages are richer than HTML and represent the meaning and structure of the Web content. Therefore the Web contents are understandable for software agents, and this will lead to a new way of processing, retrieving and analyzing of data.

As shown in the semantic Web Stack, the listed languages and technologies are used to create the semantic Web. Currently, the accepted and standardized technologies to build semantic Web applications are from the bottom of the stack up to OWL. It is still not clear how to implement the stack above, research and developing is still undergoing. In order to achieve full visions of the semantic Web, all layers of the stack need to be implemented as a precondition. (Semantic Web Stack, 2011)

The semantic Web research has developed from traditional Artificial Intelligence (AI) and ontology languages. Currently, the most important ontology languages on the Web are XML, XML Schema, RDF, RDF Schema and OWL. They provide a syntax that fits well with Web languages. They are also well balanced between expressive power and computational complexity.

## **2.4 Semantic Web in practice**

The next major way that applications are written will be using RDF, OWL, and SPARQL just as the semantic applications. Differing from JAVA programming and UML (United Modeling Language) modeling features, the semantic application will actually have executable domain models at the core of the application. (Poollock J, Semantic Web for Dummies, 54)

The semantic Web is able to provide more meaningful metadata about web information, by using RDF and OWL documents. This will help in reconstructing or transforming of the current Web into semantic Network. In addition, the more accurate data will be easier to locate by software agents. Also the representation of the Web content's meaning is better and the logical connections are clearly formed between the related objects.



## 2.4.1 Web services

There are several areas where the current technologies for discovery (UDII, Universal Description, Discovery and Integration), binding (WSDL or Web Services Description Language), and messaging (SOAP, Simple Object Access Protocol) could use OWL to provide an ontology for automatic semantic Web services.

A very simple code below demonstrates the way to include an RDF document into an HTML based Website source code. Even though, it does not make any difference on the Website's appearance, but it improves the functionality of data sharing and acquisition. There are certainly other means to use RDF/OWL documents according to different applications.

```
1 <link rel='meta'  
2 type='application/rdf+xml'  
3 title='rdfdoc'  
4 href='http://www.example.com/rdfdoc.rdf' >
```

Figure 2.3 Include RDF document.

## 2.4.2 Semantic application

The use of RDF in addition to XML can be appropriate when information from two sources need to be merged or interchanged. It is possible to combine the files by agreeing on defined terms (some kind of agreement that based on privacy and trust policies) to correspond to the same URIs (Universal Resource Identifier).

Not every semantic web application requires the maximum functionality from the semantic languages. Sometimes a part of the application can benefit from semantic Languages. In my opinion, it is the data acquisition functionality part that will be likely to benefit the most.

Each application may have a local RDF information base, and also the ability to join together with other RDF resources via Web protocols. So, if someone builds an RDF application and keeps the data open on the Web, then I can build my RDF application using his data without much integration efforts, as long as Internet connection is available. As shown in Figure 2.4, this kind of data interoperability could dramatically change the way how software applications act together over the Web.

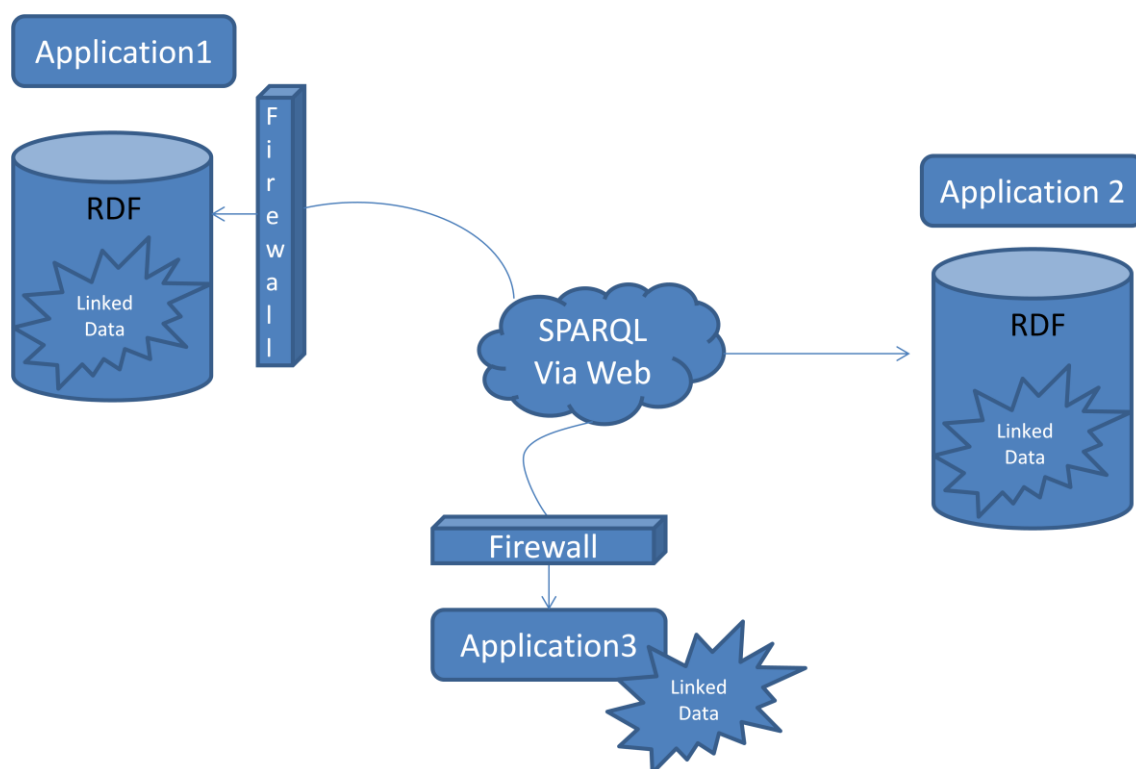


Figure 2.4 Data share with applications.

Each RDF-enabled application could work with local or remote data graphs via a URI naming infrastructure and without much overhead dedicated to transforming data into and out different structures and syntaxes.

### **2.4.3 Semantic search**

As Web ontology becomes more available and popular, the usage of RDF and OWL tags provides opportunities to see a better search result which can truly respond to detailed content requests. That means that the search result may answer directly your question or give the most relevant information and the least irrelevant information. This is the intention of semantic-based search engines and semantic-based search applications. The semantic search seeks to find documents that have a similar concept other than only rely on key words duplication today.

There are two ways to improve search results through semantic methods: 1. LSI (Latent Semantic Index), the one Google is currently developing, (see the news Neontron, Google LSI Launched: Are you ready?). 2. Semantic Web document. LSI organizes the existing HTML pages into semantic structure and can then take advantage of the implicit higher-order associations of the words with texts objects. This is one of the solutions that can be applied immediately with current Web contents (Alesso H, Semantic Search Technology).

### **2.4.4 Semantic Mechanism**

Tons of RDF assertions can be found from different sources according to application criteria, but the application would not know which ones to choose. For example, if there are several RDF documents found and they all are related to an application, either the application was predefined to use certain sources or the user has to do the picking work.

After the appropriate assertions are selected for the application's use, the application needs to decide which set of resources URIs is going to use. For example, in the Figure 2.5 the selected assertions restrict the range down to the

interpretation of URI `http://example#apple`. It knows that the solution must be one of those apples, but it does not specify the exact one.

The Figure 2.5, it illustrates the process of application use RDF semantics of assertions. In step 1, relevant assertions are selected and assigned to the application. This step usually repeats itself, as illustrated by the additional step 1.a: If an RDF document is selected, it may refer to the predefined ontologies from other documents, using an `owl:imports` OWL element, so that the assertions in those documents can be merged with other assertions which have already been selected for this application.

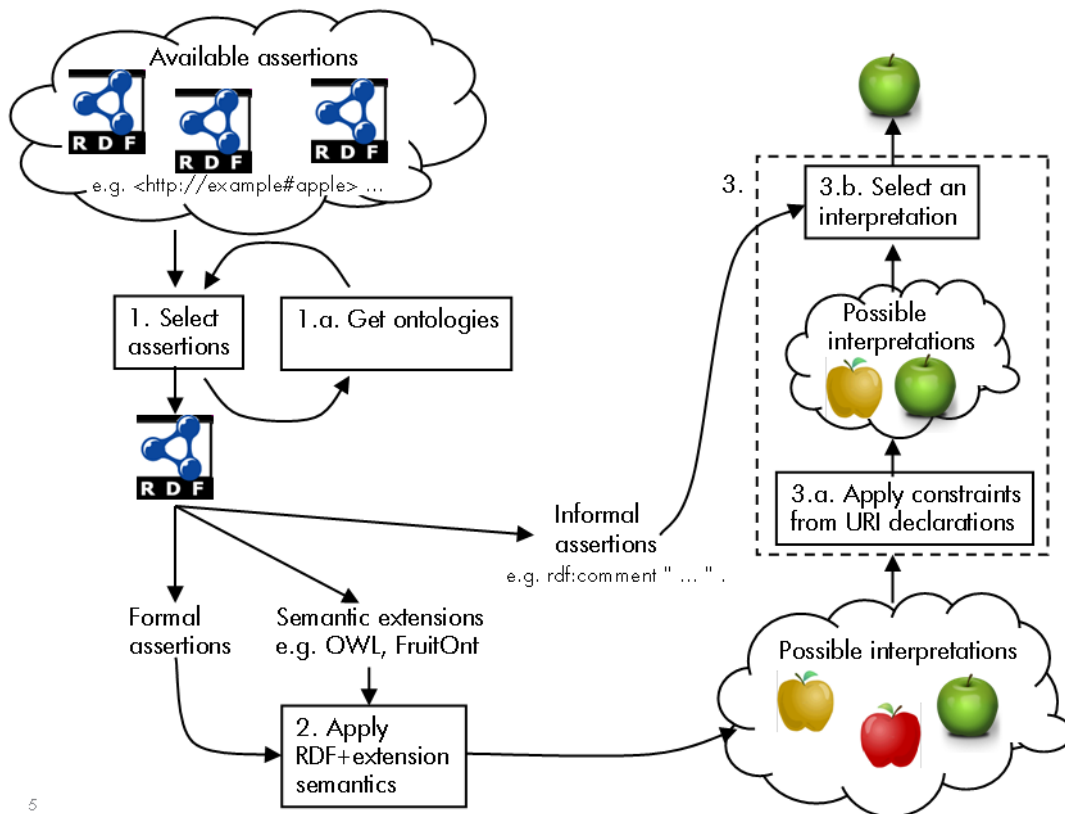


Figure 2.5 RDF semantics in the semantic Web  
(David Booth, Denotation as a Two-Step Mapping in Semantic Web Architecture)

After a set of RDF assertions has been selected, the selected assertions are often used in three ways: (David Booth, Denotation as a Two-Step Mapping in Semantic Web Architecture)

- “By applying RDF semantics, the formal assertions form the RDF graph whose entailments will be determined in step 2.”
- “Specified URIs, usually namespaces, may be recognized and trigger the inclusion of particular semantic extensions in step 2. Although such semantic extensions are often associated with well known vocabularies like OWL, any URI may signal the use of semantic extensions. For example, <http://example#FruitOnt> might signal that some special rules related to fruits should be used.”
- “Embedded informal assertions, such as prose contained in `rdfs:comment` statements, may be used later in step 3 to help the user select the most appropriate interpretation corresponding to a particular URI.”

In step 3, with the assistance of informal assertions, an interpretation will be selected from the several possible interpretations. The selected interpretation links a URI which was used in step 1, to a resource, and finally, the green one is found as the solution.

## 3 RESOURCE DESCRIPTION FRAMEWORK

XML is a universal meta-language for defining a markup (W3C, Extensible Markup Language). It provides a framework for the data exchange, but it does not provide a mechanism to deal with the meaning of the data.

RDF was developed by W3C in order to build and extend XML. RDF is a format for data that uses a simple relational model that allows structured data to be reused and remixed across different applications. (W3C, W3C Semantic Web Frequently Asked Questions).

Below, it is a segment of XML markup tags:

```
<book>  
  <title>Suomi-Kiina Sanakirja</title>  
</book>
```

Figure 3.1 XML tags.

We could understand that the book has a title – Suomi-Kiina Sanakirja. A typical simple sentence is known to contain three parts: subject - book, predicate - has, and, object - Suomi-Kiina Sanakirja. There is no way that a machine could acquire the same information based on XML alone.

To enable machines to do tasks intelligently and automatically, it is necessary to let the machines to know the meaning of the content. This is where RDF can provide new capabilities built upon XML.

### 3.1 RDF triple

The RDF model is based on statements made about resources that can be anything with URI. This basic model produces a triple usually contains three parts: (W3C, RDF Triples)

- “the subject, which is an RDF URI reference or a blank node”
- “the predicate, which is an RDF URI reference”
- “the object, which is an RDF URI reference, a literal or a blank node”

With the model of triple, RDF can express relationships between two sources.

## **3.2 Elements**

Elements include: Syntax, Header, Namespace and Description. In this part of documentation I describe different elements with examples separately. The elements are the basis for creating and implementing RDF documents.

### **3.2.1 Syntax**

An RDF syntax is based on An XML syntax.

RDF provides a reasonable way to describe the relationships of properties and values among resources. Encoding RDF triples in XML makes an object portable across platforms. Because RDF data can be expressed in XML syntax, it can be passed over the Web as a document and it can be parsed using the existing software.

The combination of RDF and XML enables individuals or programs to locate, reuse or store the information from the semantic Website.

### **3.2.2 Header**

You may discover that an RDF document looks pretty much like an XML document in elements, tags and namespaces. However the RDF document starts with a header as an “rdf:RDF” element which also specifies a number of namespaces.

```

1 <?xml version="1.0"?> <!-- XML header -->
2
3 <rtf:RDF <!-- root element tag -->
4   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
6   <!-- XML namespaces -->
7 <rdf:Description rdf:about="book">
8   <dc:has title>"suomi-kiina sanakirlja"</dc:has title>
9   </rdf:Description>
10  <!-- triple -->
11 </rdf:RDF> <!--The end tag -->

```

Figure 3.2 RDF header source code.

Line1: Syntax declaration.

Lin4&5: Namespaces for rdf and dc, as well as the URLs where they are defined.

Line7&8 Triple-subject, predicate, object.

Line11: Indicates the end of the RDF document.

### 3.2.3 Namespaces

RDF is using the XML namespace mechanism. In RDF, external namespaces are expected to define the RDF documents, which are used to import the RDF sources. This allows the reusing of the resources and adding additional features for the resources in order to produce a huge data web. For example:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
```

Figure 3.3 Namespace example source code.

Where,

The prefix for RDF syntax is given as “rdf”.

XMLNS means a namespace.



### 3.2.4 Description

“The `rdf:about` attribute of element `rdf:Description` is equivalent to that of an `ID` attribute, but it is often used to suggest that the object about which a statement is made has already been defined elsewhere.” (Antonou G, Harmelen F, 2004, 71)

The content of `rdf:Description` elements are called property elements. The description can be also defined within other descriptions producing a nested description. From the `rdf:resource` attribute, you can find further definition of `ID` , and the `rdf:type` element brings a structure to the `rdf` document.

### 3.2.5 Data Types

RDF uses a type to identify what kind of thing a resource is. RDF uses two general types, a resource type and a literal type. For example, the resource `sanakirja` can refer to a type of book. The value of the type can be another resource, which would mean that more information can be found in the type itself.

The types can be specified with a triple. For example:

```
<http://www.kirjasto.fi/sanakirja>,  
  rdf:type,<http://www.adlibris.com/fi/book>
```

Figure 3.4 Data type example source code.

The resource `<http://www.adlibris.com/fi/book>` is used to represent a book. The predicate of the triple is `rdf:type`, which is in the RDF namespace, since “type” predicate is built in to RDF.

RDF uses XML data types which include a wide range of data types. In addition, RDF allows any externally defined data typing schema.

### 3.3 RDF Schema

RDF is a universal language that lets users describe resources using their own vocabularies. That it is actually something pretty amazing compared to other universal languages, where you have to use predefined vocabularies. RDF does not make assumptions about any particular application domain, nor define the semantics of any domain. A user can take over and decide what to do in RDF Schema. (Antonou G, Harmelen F, 2004, 80)

“RDF Schema provides modeling primitives for organizing Web objects into hierarchies. Key primitives are classes and properties, subclass and sub property relationships, domain and range restrictions.” (Antonou G, Harmelen F, 2004, 17)

RDF Schema is a not very expressive language for ontology. In addition, it is necessary to use more powerful ontology languages, such as OWL. It will be explained in details in the next chapter. The OWL extends RDF Schema and it can be used to represent the more complicated relationships between objects.

#### 3.3.1 Core Classes

Some of the core classes include:

- `rdfs:Resources`, the class contains all resources.
- `rdfs:Class`, the class of all classes.
- `rdfs:Property`, the class contains all properties.

For example, we can define a class `sanakirja` as below:

```
<rdfs:Class rdf:ID="sanakirja">  
...  
</rdfs:Class>
```

Figure 3.5 RDF class example code.

A class contains a set of elements. All individual objects that belong to a class are instances of the class. The relationship between instances and classes in RDF is expressed by `rdf:type`. The classes are used to sort contents in an RDF document using Schema.

Once the classes are created, the relationships between them must be established through subclasses, super classes and so on.

### 3.3.2 Properties

#### RDFS / RDF Properties

Element	Domain	Range	Description
<code>rdfs:domain</code>	Property	Class	The domain of the resource
<code>rdfs:range</code>	Property	Class	The range of the resource
<code>rdfs:subPropertyOf</code>	Property	Property	The property is a sub property of a property
<code>rdfs:subClassOf</code>	Class	Class	The resource is a subclass of a class
<code>rdfs:comment</code>	Resource	Literal	The human readable description of the resource
<code>rdfs:label</code>	Resource	Literal	The human readable label (name) of the resource
<code>rdfs:isDefinedBy</code>	Resource	Resource	The definition of the resource
<code>rdfs:seeAlso</code>	Resource	Resource	The additional information about the resource
<code>rdfs:member</code>	Resource	Resource	The member of the resource
<code>rdf:first</code>	List	Resource	
<code>rdf:rest</code>	List	List	
<code>rdf:subject</code>	Statement	Resource	The subject of the resource in an RDF Statement
<code>rdf:predicate</code>	Statement	Resource	The predicate of the resource in an RDF Statement
<code>rdf:object</code>	Statement	Resource	The object of the resource in an RDF Statement
<code>rdf:value</code>	Resource	Resource	The property used for values
<code>rdf:type</code>	Resource	Class	The resource is an instance of a class

Figure 3.6 RDF/RDFS Properties Table.

(W3 Schools, RDF Reference)

As illustrated in Figure 3.6, it states the most commonly used RDF/RDFs property elements. The description part in the table explains well where or in which situation each element can be used.

Here is an example I wrote. It states that sanakirja belongs to books only and the property value is always a literal (string).

```
<rdf:Property rdf:ID="sanakirja">  
  <rdfs:domain rdf:resource="#book"/>  
  <rdfs:range rdf:resource="&rdf;Literal"/>  
</rdf:Property>
```

Figure 3.7 RDF property example source code.

### 3.4 Example: Cars

Here we present a simple ontology of car types. The class relationships are shown in the figure below:

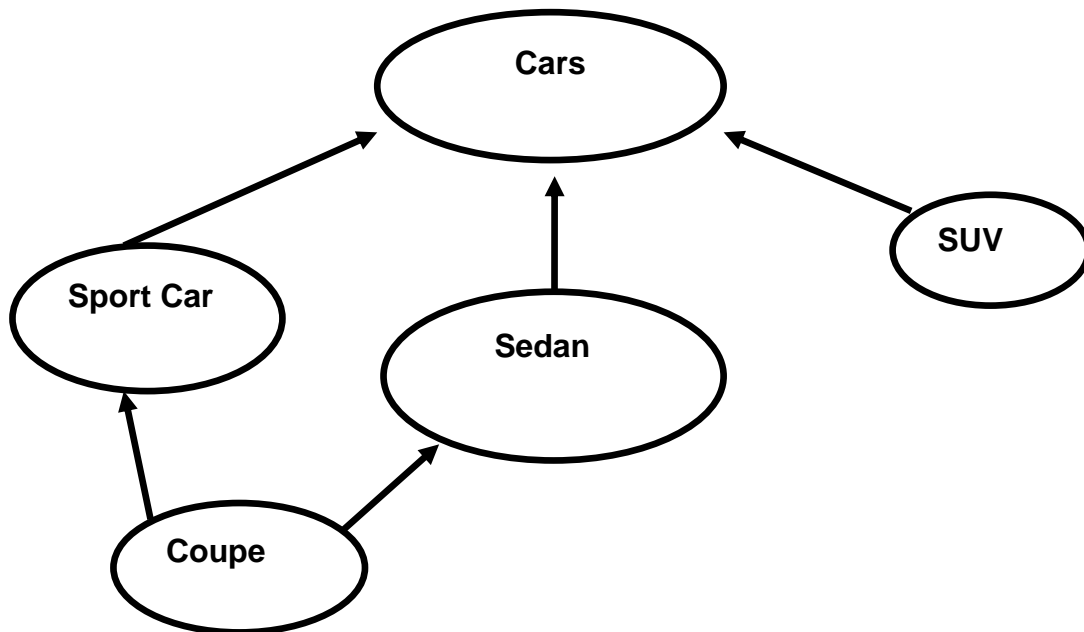


Figure 3.8 An example of a class hierarchy for the cars.

According to the class relationship shown above, a basic prototype can be created by using the RDF and RDF Schema.

To see the full source code of this example, check Appendix 1.Cars\_example.xml. The source code was written in NotePad++ 5.9.8 and saved as an .XML file, and then it can be opened by browser. I use Firefox 11.0, and the result you can see in the Figure 3.9 below.

```
- <rdf:RDF>
  <rdfs:Class rdf:ID="cars" rdfs:comment="The class of Cars"/>
  - <rdfs:Class rdf:ID="sportcar" rdfs:comment="The class of sport car" >
    <rdfs:subClassOf rdf:resource="#cars"/>
  </rdfs:Class>
  - <rdfs:Class rdf:ID="suv" rdfs:comment="The class of SUV">
    <rdfs:subClassOf rdf:resource="#cars"/>
  </rdfs:Class>
  - <rdfs:Class rdf:ID="sedan" rdfs:comment="The class of sedan">
    <rdfs:subClassOf rdf:resource="#cars"/>
  </rdfs:Class>
  - <rdfs:Class rdf:ID="coupe" rdfs:comment="The class of Coupe">
    <rdfs:subClassOf rdf:resource="#sportcar"/>
    <rdfs:subClassOf rdf:resource="#sedan"/>
  </rdfs:Class>
  - <rdf:Property rdf:ID="price" rdfs:comment="It is a property of Cars">
    <rdfs:domain rdf:resource="#cars"/>
  </rdf:Property>
</rdf:RDF>
```

Figure 3.9 Cars\_example.xml in browser.

As you can see, even the source code was successfully loaded by the browser (no error found), but it still appears pretty much like a code and it is not possible for human to understand. That is because RDF and RDFs were created for machine understanding, as I mentioned earlier.

We can also open the Cars\_example.xml source code in an XML notepad 2007 from Microsoft. The version I use is XML Notepad 2007. The XML Notepad provides a tree view (as shown in Figure 3.10) and an XSL output according to

the source code. In addition, it is possible to create a similar .xml file from the tree view.

The tree view gives a very clear view of relationships between the classes and properties. In addition, the XSL output gives pretty much same the result as in the Web browser, since the xml file here does not have a style sheet.

The figure 3.10 below is a partly opened tree view of cars\_example.xml. For the fully open tree view, see Appendix 2.Cars\_example.xml in a tree view.



Figure 3.10 Cars\_example.xml in tree view.

### 3.5 RDF graph

An RDF graph is a set of RDF triples.

“The set of nodes of an RDF graph is the set of subjects and objects of triples in the graph.” (W3C, RDF Semantics, 2004)

There is a couple of RDF Graph Visualization Tools available. The one I chose is called RDF Gravity 1.0. The RDF Graph Visualization Tool (RDF Gravity) is a tool for visualizing directed graphs built in RDF and OWL. “The tool provides a simple yet powerful visualization of RDF graph structures and the ability to filter out and visualize specific parts or fragments of RDF Graphs.” (RDF Gravity, Sunil Goyal, Rupert Westenthaler)

We can change the cars\_example.xml to cars\_example.rdf and bring it into the RDF gravity to have a look. The Figure 3.11 below is the RDF graph laid out by the RDF gravity. The reason I present the RDF graph here is to make RDF and RDFs more visual so that it is easier to understand the data relationships in the example I made.

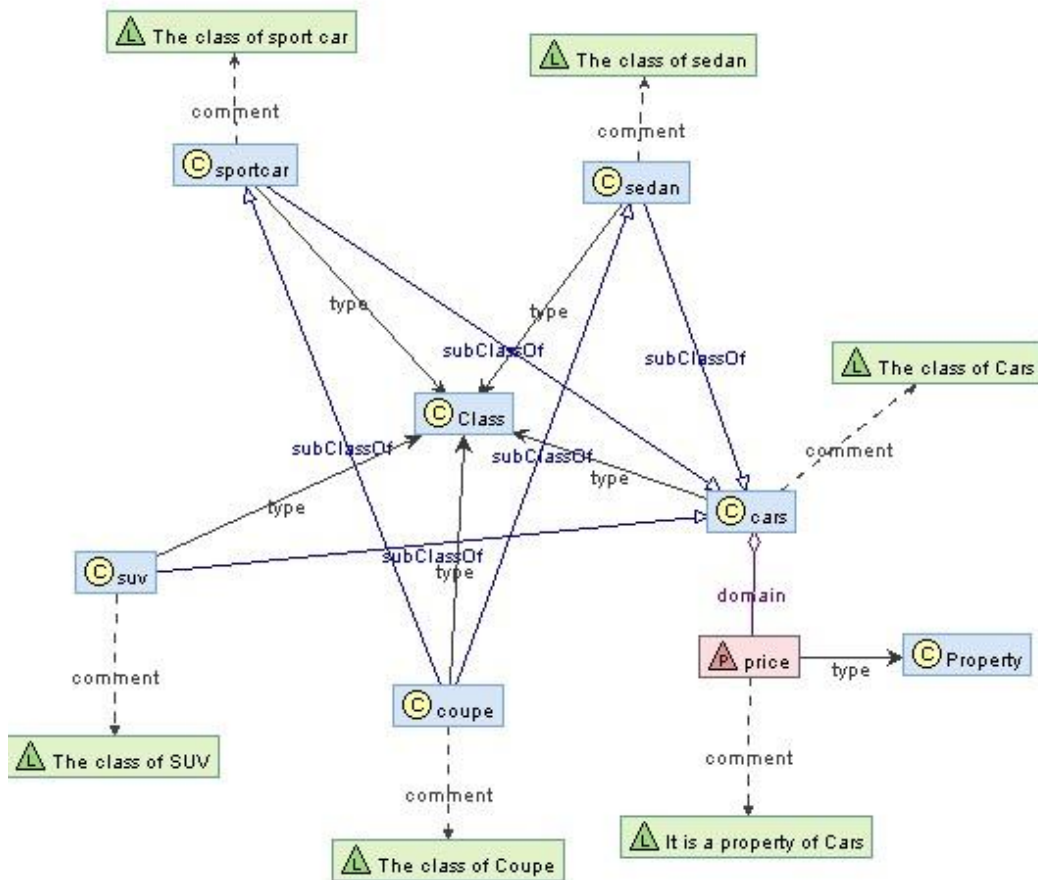


Figure 3.11 A visual graph of a car example.

For the full application interface screenshot, see Appendix 4. Cars\_example.rdf RDF graph.

### 3.6 FOAF application

With the rapid growth of the Internet, many companies and organizations have realized the power of association of the Web. The Friend of a Friend (FOAF) RDF vocabulary, originally invented by Dan Brickley and Libby Miller, gives a simple expression for community membership. (FOFA vocabulary specification 0.9)



FOAF allows the collection and representation of personal information and relationships. Through the associated search engines, individuals can find people with similar interests through FOAF. The system represents a useful building block for creating an information system that supports online communities. (Finding friends with XML and RDF)

FOAF has become a popular application, and it has a potential to be an important managing tool. In addition to providing simple directory services, the information from FOAF is accessible from multiple other ways. It is also one of the important characteristics of the semantic Web.

Using an email address (mailto:) has turned out a good way to identify a person. Even if a person can have many email addresses, it is a reasonable idea. A sample of FOAF description of the author is as follows:

```
1 <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:foaf="http://xmlns/foaf/0.1">
4
5 <foaf:Person>
6   <foaf:name>Pengfei Zhang</foaf:name>
7   <foaf:mbox rdf:resource="r8zhpe00@students.oamk.fi"/>
8 </foaf:Person>
9
10 </rdf:RDF>
```

Figure 3.12 FOAF sample code.

### 3.7 RDFa

RDFa is a proposed set of extensions to XHTML (Extensible HyperText Markup Language). The "a" here stands for attributes. Its intent is to allow the inclusion of metadata in any XML document, but RDFa is primarily used in XHTML. RDFa allows machines to understand and use RDF semantics from within a Web page.

A more relevant object can be found on a Web page like audio, video and image. If you are searching information on the Web, you may find hundreds of pages of the information. Before RDFa, the information was represented in XHTML elements: Only a human can read. With RDFa, there is now a standard-based approach to representing the Web page metadata, just like RDF.

I created a simple page using RDFa about myself and it looks like the Figure 3.13 – a rendering of a basic Web page containing a semantic markup. For the source code, see Appendix 5. RDFa.html.



## Information about Pengfei Zhang



Author of This Thesis - Semantic Web. Student of OAMK.

Email: [r8zhpe00@students.oamk.fi](mailto:r8zhpe00@students.oamk.fi)

Phone: [+358 401234567](tel:+358401234567)

© Clyde Zhang

Figure 3.13 RDFa in Browser.

Other information about different objects on the page can also be included. As you can see, embedding RDF-based data can be an easy and straightforward task with RDFa.

Note that, there is no direct connection between RDFa and FOAF. FOAF is an RDF document that uses machine understandable vocabulary and people can

use it to describe online profiles. However, RDFa can be added into a part of HTML document. RDFa is just a solution to use RDF in attributes in HTML and XHTML Web pages so that data is structured while the page appearance is still kept as HTML.

### **3.8 Limitation of RDF and RDFs**

One of the limitations of RDF is from the use of the properties as special kinds of resources. The properties themselves can be used as an object in an object-attribute-value statement. This flexibility can cause confusion in modeling.

RDF promotes the use of standardized vocabularies, standardized classes and standardized properties. While RDF XML-based syntax is well-suited for machine processing but not user friendly.

## 4 WEB ONTOLOGY LANGUAGE

“The OWL Web Ontology Language is designed for use by applications that need to process the content of language instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDFs) by providing additional vocabulary along with a formal semantics.” OWL has sublanguages: OWL Lite, OWL DL, and OWL Full. (W3C, OWL Web Ontology Language)

### 4.1 OWL offers more than RDFs

RDF and RDFs allows the representation of ontological knowledge. However, a number of other features are missing.

- The `rdfs:range` defines the property range applies to all classes. So we cannot define a range for only few classes.
- Sometimes we want to build a class by using other classes' intersection, aggregation or complement. RDFs does not allow that to happen.
- When we want to place a restriction on the value of the property may have you may find out it is impossible to express it in RDFs.

Thus, we need OWL - the Web Ontology Language. It is richer than RDFs, and offers more features.

The semantic Web is a vision of the next generation Web, where information will be given exact meaning, making it easier for machines to automatically process and integrate the available Web information. The semantic Web will take advantage of XML's ability to define customized tagging schemes and RDF's flexible approach to represent data. As the “Layer Cake” that I presented in Chapter 2.3, the first level on top of RDF required for the semantic Web is an

ontology language. The Web ontology language can formally describe the meaning of terminology used in Web documents. If we expect machines to be able to perform useful reasoning tasks with semantic documents, the language must go beyond the basic semantics of RDF Schema. The upper layer it goes, the more intelligent the Web will be. (W3C, OWL features)

OWL should be an extension of RDFs, on the basis of using an RDF meaning of classes and properties, and offering richer expressive power. The main requirements for ontology languages are: a well defined syntax, expressive, symbolic and efficient reasoning support. OWL has been designed to meet all these requirements as an ontology language.

## **4.2 Three subs of OWL**

“The W3C-endorsed OWL specification includes the definition of three variants of OWL, with different levels of expressiveness.” (Wikipedia, Web Ontology Language)

Three increasingly expressive sub languages are OWL Lite, OWL DL and OWL Full. “Each of them is a syntactic extension of its simpler predecessor. The following set of relations hold. Their inverses do not.”

“Every legal OWL Lite ontology is a legal OWL DL ontology.

Every legal OWL DL ontology is a legal OWL Full ontology.

Every valid OWL Lite conclusion is a valid OWL DL conclusion.

Every valid OWL DL conclusion is a valid OWL Full conclusion.”

(W3C, OWL Web Ontology Language)

## **4.3 The OWL elements**

Like syntax, header, classes and property. I describe different elements separately with examples. Elements are the basis for creating or implementing OWL documents. Some of them may be similar to RDF and RDFs elements.

### **4.3.1 Syntax**

OWL was built upon RDF and RDF Schema and it also uses an XML based syntax which makes OWL easy to learn and use. Also other syntax forms for OWL have been defined:

- An XML-based syntax is more easily read by human users, because it does not follow RDF conventions.
- An abstract syntax used in the language specification document that is more compact and readable than the XML syntax or the RDF/XML syntax.
- A graphic syntax based on the UML (Unified Modeling Language) convention, which is widely used, it is an easy approach to people to become familiar with OWL.

(W3C, OWL Web Ontology Language Semantics and Abstract Syntax)

### **4.3.2 Header**

OWL documents are usually named OWL ontologies and they are one kind of RDF documents. The root element `rdf:RDF` specifies a set of relevant namespaces:

```

1 <rdf:RDF
2   xmlns:owl="http://www.w3.org/2002/07/owl#"
3   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5   xml:xsd="http://www.w3.org/2001/XMLSchema#">

```

Figure 4.1 OWL header.

An OWL document may start with a number of assertions. These assertions are inside an owl:Ontology element, which may contain comments, version info, and other ontologies documents.

### 4.3.3 Class

Classes are defined using an owl:Class element. It is also a subclass of rdfs:Class. For example, we can define a class sanakirja as below:

```

1 <owl:Class rdf:ID="sanakirja">
2   <rdfs:subClassof rdf:resource="#book"/>
3 </owl:Class>

```

Figure 4.2 Class example.

We can also say that the class defined below is disjoint from the tietosanakirja and muistikirja classes by using an owl:disjointWith element. Since sanakirja has no connection with tietosanakirja and muistikirja, even though they all are books, the relationship is disjointed. This is something that cannot be done with RDF Schema. The rdf:about can be used to refer the ID. See example below:

```

1 <owl:Class rdf:ID="sanakirja">
2   <owl:disjointWith rdf:resource="#tietosanakirja"/>
3   <owl:disjointWith rdf:resource="#muistikirja"/>
4 </owl:Class>

```

Figure 4.3 An example of disjointed class.

The dictionary is same as sanakirja but in a different languages. The language difference in their description and instances may be different but in general they

are the same objects. Thus they are equivalent. Equivalent classes can be defined by using an owl:equivalentClass element:

```
1 <owl:Class rdf:ID="dictionary">
2   <owl:equivalentClass rdf:resource="sanakirja"/>
3 </owl:Class>
```

Figure 4.4 An example of equivalent Classes.

There are two predefined classes in OWL, owl:Thing and owl:Nothing. The first one is the most general class; It contains everything, because everything is a thing. The second one is an empty class. It usually remains anonymous.

### 4.3.4 Property elements

In OWL there are two categories of properties:

- Object properties: link objects to other objects. Examples are suomi-kiina and kiina-suomi.
- Data type properties: link objects to data type values. Examples are price, pages and title etc. OWL does not have any predefined data types. However, it allows us to use XML Schema data types.

Here is an example of a data type property:

```
1 <owl:DatatypeProperty rdf:ID="price">
2   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
3 </owl:DatatypeProperty>
```

Figure 4.5 Data type property example.

Usually, user defined data types are collected in an XML schema. Here is an example of an object property:



```

1 <owl:objectProperty rdf:ID="suomi-kiina">
2   <rdfs:domain rdf:resource="#sanakirja"/>
3   <rdfs:range rdf:resource="#books"/>
4   <rdfs:subPropertyOf rdf:resource="#languages"/>
5 </owl:objectProperty>

```

Figure 4.6 An example of object property.

We may declare more than one domain and range. But it is not necessary in most of cases.

### 4.3.5 Property restrictions

With `rdfs:subClassOf` we can specify a class A to be a sub class of B; Then all instances belonging to A are also instances of B.

If we wish to declare, instead, that A fulfills certain requirements, that is, all instances of A fulfill the requirements. In another word, we can say that A is the subclass of B, where B collects all objects that fulfill the requirements. That is exactly how it is done in OWL. In general, B can be anonymous.

The following element requires that the language of sanakirja is suomi-kiina.

```

1 <owl:Class rdf:about="#sanakirja">
2   <rdfs:subClassOf>
3     <owl:Restriction>
4       <owl:onProperty rdf:resource="#suomi-kiina"/>
5       <owl:allValuesFrom rdf:resource="#languages">
6     </owl:Restriction>
7   </rdfs:subClassOf>
8 </owl:Class>

```

Figure 4.7 Property restriction example.

The `owl:allValuesFrom` element is used to specify the class of possible values the property can take, in other words, all values of the property must come from this class. In the example above, the only language required of sanakirja is suomi-kiina.

In general, an owl:Restriction element contains owl:onProperty, datatype elements and restriction declarations. One type of restriction declaration constrains the values that property may have by using owl:allValuesFrom, owl:someValuesFrom and owl:hasValue. Another type defines cardinality restrictions. For example, we can define the sanakirja has to have at least 2 languages.

```
1 <owl:Class rdf:about="#sanakirja">
2   <rdfs:subClassOf>
3     <owl:Restriction>
4       <owl:onProperty rdf:resource="#languages"/>
5       <owl:minCardinality rdf:datatype="&xsd;nonNegativeinteger">
6         2
7       </owl:minCardinality>
8     </owl:Restriction>
9   </rdfs:subClassOf>
10 </owl:Class>
```

Figure4.8 Cardinality restriction example.

We had to specify that the literal “2” is to be interpreted as non-NegativeInteger, and that we used the “xsd” namespace declaration was made in the header element to refer to the XML Schema document.

## 4.4 OWL2

The OWL 2 Web Ontology Language is also a formally defined ontology language for the semantic Web. “OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents.” (W3C, OWL 2 Web Ontology Language, 2009)

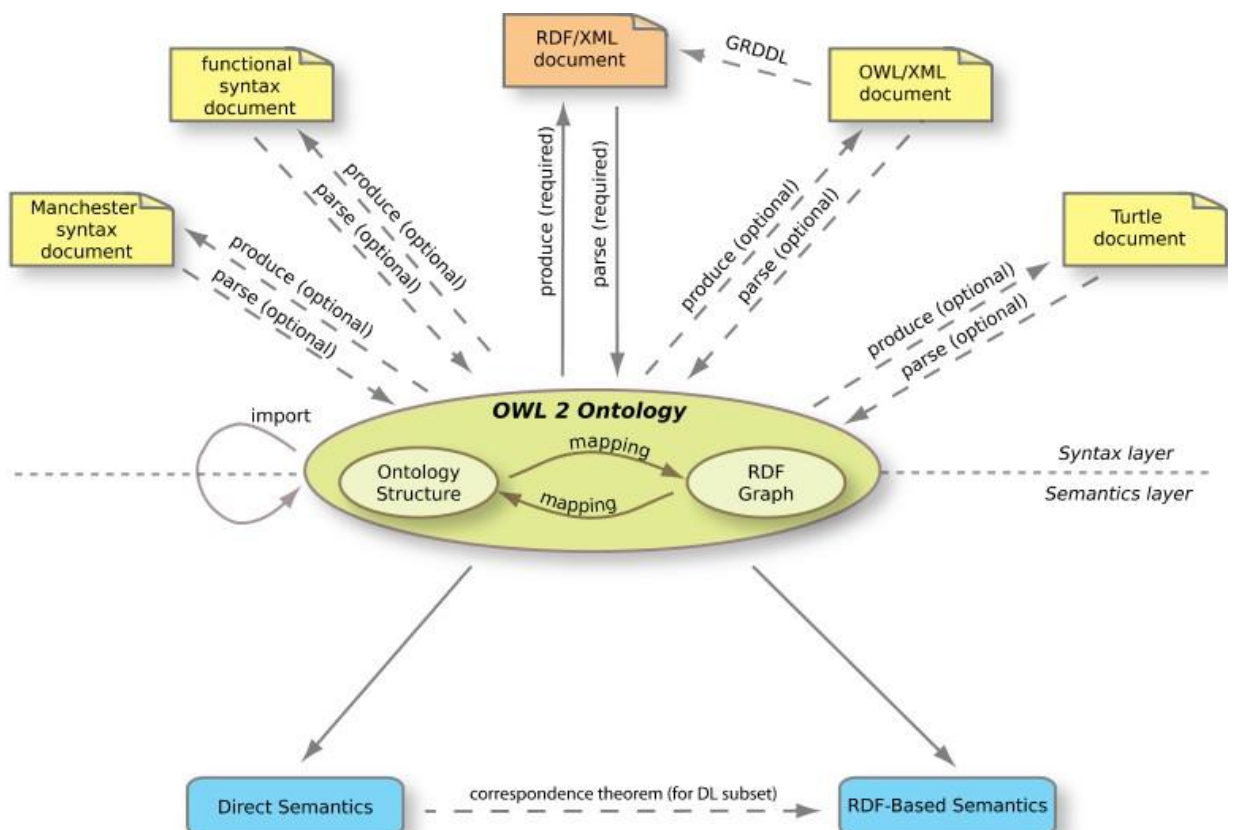


Figure 4.9 The Structure of OWL 2. (W3C, OWL 2 Web Ontology Language, 2009)

## 5 CAR ONTOLOGY PROTOTYPE

The main idea here is to build a car ontology by using OWL. The ontology here means something that helps you to define or utilize the object more efficiently. The fact is that different systems may use different names for the same objects, or they may use the same names for different objects. Ontology contains lots of reticular properties and relationships, and the objects can be better defined even they are named differently.

The same I have declared earlier, OWL is a richer and more expressive language than RDF Schema. However, it has a nontrivial relationship with RDF Schema; it is also made for machines to understand it and not for human reading. Therefore, the output might not make so much sense for a human, but it makes sense for machines.

Figure 5.1 shows the basic relationships about classes and their subclasses. Pay attention to that information of the classes is only simplified on the use of this. The entire graph is much larger and more complicated. I will illustrate the complete RDF graph with the RDF gravity 1.0 later in this chapter. For the completed source code of this prototype, see Appendix 6. Car\_ontology source code.

Note that, all data about Audi in this project is taken from Audi's official website: [www.audi.com](http://www.audi.com).

### 5.1 Classes and subclasses

The class and subclass relationships in this ontology are clearly shown in Figure 5.1. The figure logic follows the rule: top to bottom, general to specific. Each ellipse element stands for a class. Some may be subclasses and some may contain a property, a comment or a label.

The entire ontology is built upon these relationships. It is much easier for us to understand this kind of figure than the ontology RDF graph.

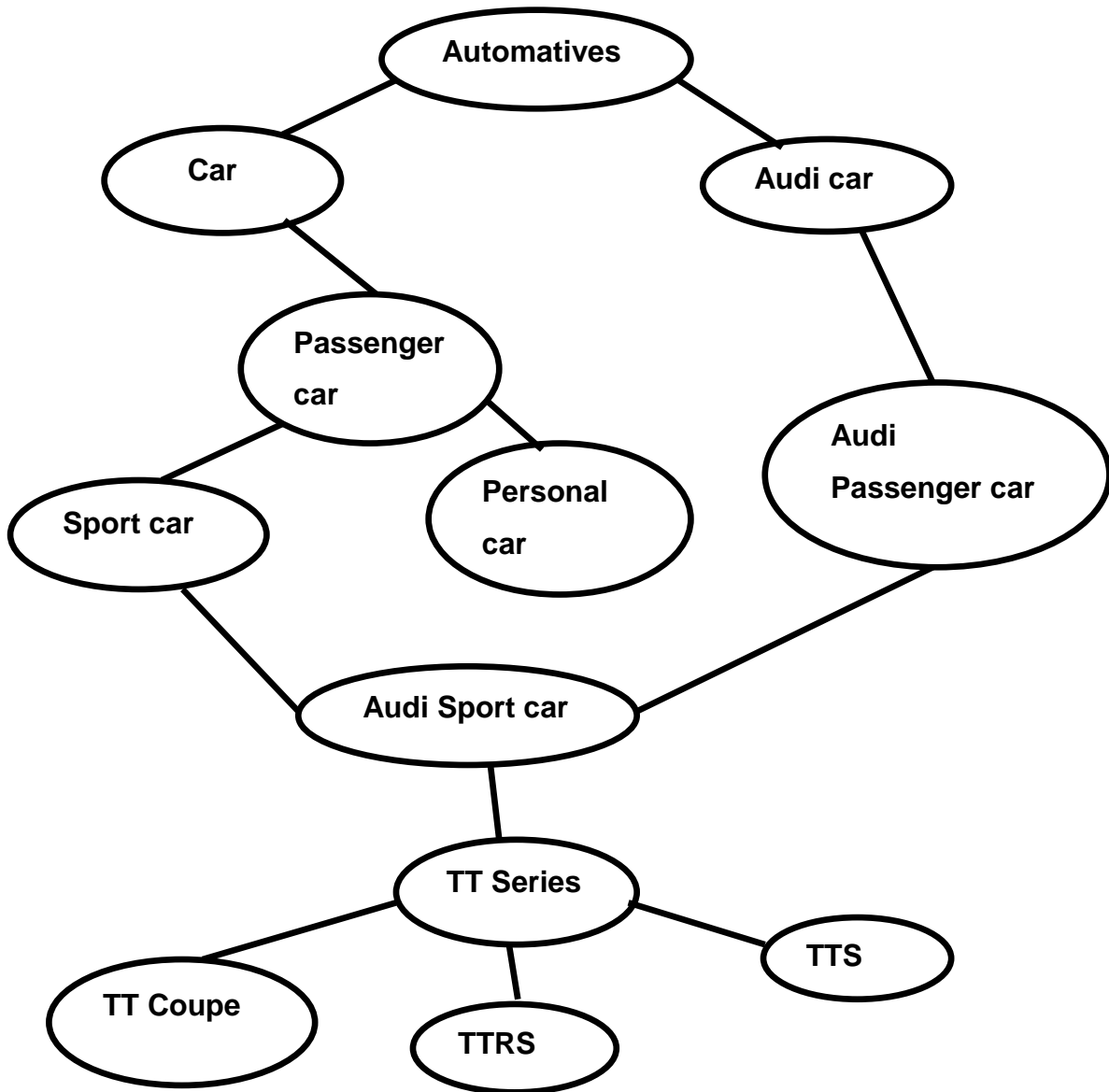


Figure 5.1 Classes and subclasses of the car ontology.

```
15 <owl:Class rdf:ID="Automotive">
16   <rdfs:comment>
17   Automotives form a class.
18   </rdfs:comment>
19 </owl:Class>
```

Figure 5.2 Class.

As shown in Figure 5.2, I used an owl:Class element to build a class and its ID is Automotive. In the element rdfs:comment, it indicates the description I gave for this class. All the other classes in this prototype were built in the same way.

```
21 <owl:Class rdf:ID="Car">
22   <rdfs:comment>
23     Cars form a subclass of Automotive.
24   </rdfs:comment>
25   <rdfs:label>car</rdfs:label>
26   <rdfs:subClassOf rdf:resource="#Automotive" />
27 </owl:Class>
```

Figure 5.3 Subclasses.

The class Car was built and with an rdfs:subClassof element, and now, this class is also a subclass of Automotive. Other Subclasses in this prototype were built in the same way. One element I have to mention here is the rdfs:label. It is an instance of rdf:Property that is used to provide a human understandable version of a resource name.

## 5.2 Properties

```
29 <owl:Class rdf:ID="AudiCar">
30   <rdfs:comment>
31     Audi cars are exactly those automotives
32     that are manufactured in Germany.
33   </rdfs:comment>
34   <owl:intersectionOf rdf:parseType="Collection">
35     <owl:Class rdf:about="#Automotive" />
36     <owl:Restriction>
37       <owl:onProperty rdf:resource="#manufactured-in" />
38       <owl:hasValue>
39         <xsd:string rdf:value="Germany" />
40       </owl:hasValue>
41     </owl:Restriction>
42   </owl:intersectionOf>
43 </owl:Class>
```

Figure 5.4 Class Audicar and its property.

The class Audicar here looks much more complicated than the classes I explained before. That is because I added a property and a value to it.



rdfs:domain is a description which specifies the property Torque. If I had not specified the domain here then any resource could be the subject. The rdfs:range element defines the property's data range. The class of those resources that may contain values, in this case, are strings.

### 5.3 Other elements

```
2 <!DOCTYPE owl [  
3   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >  
4 ]>
```

Figure 5.7 Data type reference.

OWL uses most of the built-in XML Schema data types. There are lots of predefined data types in the references, according to the values. The data types I used are xsd:integer and xsd:string.

```
5 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
6   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"  
7   xmlns:xsd="http://www.w3.org/2001/XMLSchema#"  
8   xmlns:owl="http://www.w3.org/2002/07/owl#">
```

Figure 5.8 namespaces

Different namespaces were used in this document. As you can see from Figure 5.8 four namespaces were included at the beginning of the document.

```
10 <owl:versionInfo>  
11   My example version 1.0, 7 March 2012  
12 </owl:versionInfo>
```

Figure 5.9 Version info.

This part does not effect much on the functionality of thw whole document. In addition, owl:versionInfo provides information for the versioning systems and the object is usually a literal. The reasons why I did it in this document are really simple: 1. demostrating the element. 2. maintaining of the developing a new version in future.



## 5.4 Testing

The RDF graph of a Car ontology is shown below in Figure 5.10. Since RDF Schema and OWL were both designed for a machine to understand it, they would not make any great visual appearance. So I used RDF visualization tool RDF Gravity 1.0 to test my ontology code. Since RDF Schema and OWL are both built upon XML, the code was written in an XML syntax. For testing purpose, I changed the file into .rdf and opened it by RDF Gravity 1.0. The .rdf is actually the appropriate document format when putting it into practice.

What you see in Figure 5.10 is a graph that includes class and property relationships and logics. Unlike in Figure 5.1, I listed at beginning of this Chapter, the logic of the ontology seems easy and straightforward. The actual relationships and logics that are understood by a machine should be like Figure 5.10, complicated for us but understandable for machines. Machines can use ontology to locate and define exact data.

As you can see this graph contains a set of classes and subclasses, properties, labels, URIs etc. They are all connected to different lines, which indicate different relationships between two objects. For more notation explanations, see Appendix 3. General Notations, and the full screen shot of RDF gravity interface see Appendix 7. Car\_ontology.rdf RDF graph.

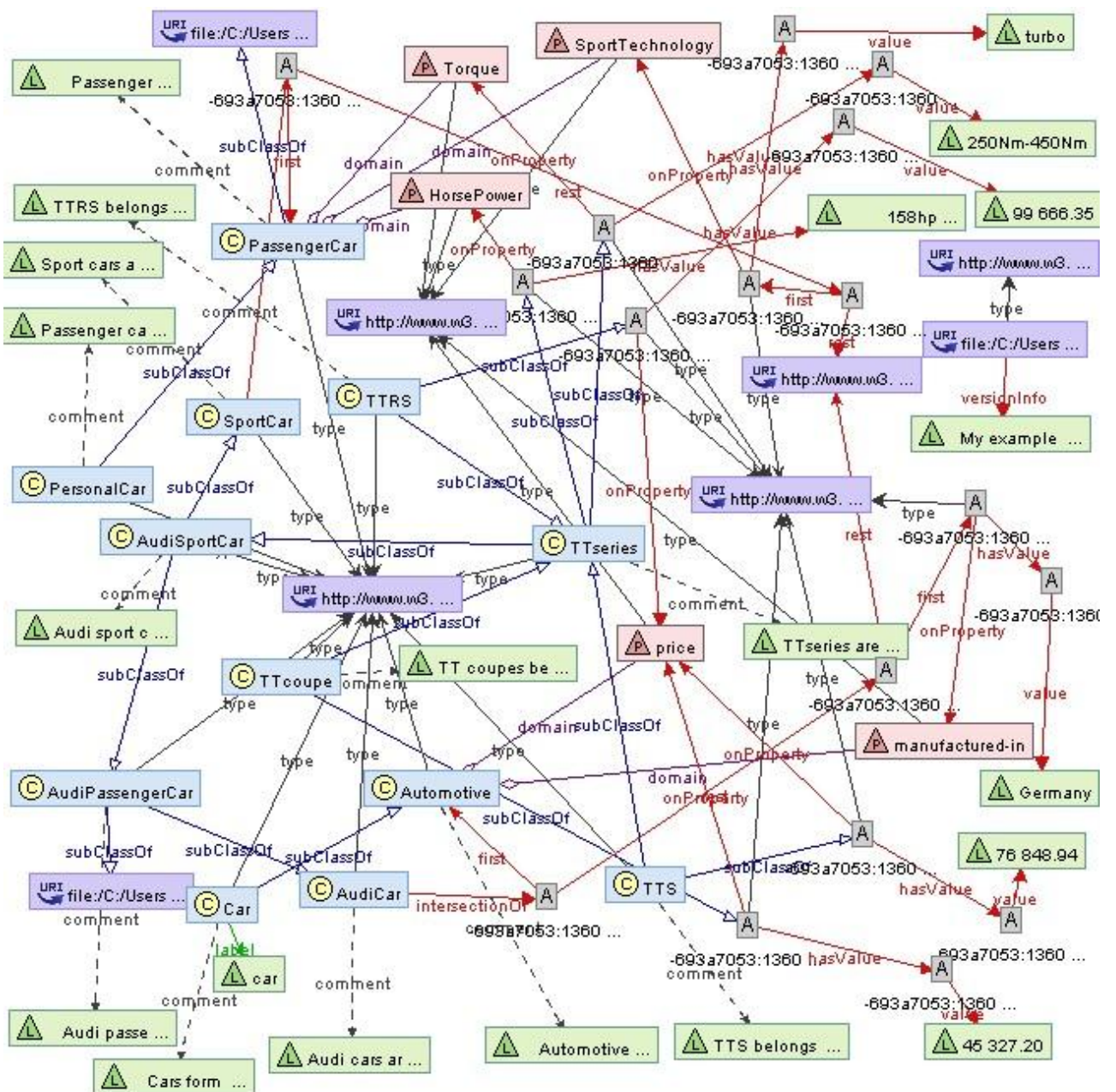


Figure 5.10 RDF graph.

## **6 POSSIBILITIES OF FURTHER DEVELOPMENT**

This thesis was completed better than I expected. I did not plan to implement such a complex ontology as presented in the previous chapter. However, the ontology prototype has not been developed to the point that it can be put into practice directly.

The semantic Web services or applications need a huge open database or knowledge base as a precondition. In the other words, a semantic network is needed, because they will not work independently. For example, in my ontology prototype, I had to manually input all the information and data, because I did not have an open data network as a backbone so that the information could have been acquired automatically.

Even though the semantic languages – RDF, RDF Schema and OWL have advantages compare with the currently Web languages, there are also obvious limitations. With the leading effort of the W3C community, greater languages will be developed for the semantic Web, just like the emerging semantic language OWL2. Therefore, my ontology prototype may be done in a completely different language but with the same functionality.

I strongly believe that in the near future, more and more developers and communities will realize the importance and convenience of the semantic Web. When more and more Web pages and data are constructed in the semantic languages, the implementation of semantic applications will be easier and more efficient.

## 7 CONCLUSION

I did not have much Web developing knowledge and experience, so this thesis was quite challenging for me, especially, so was the topic semantic Web. We often hear people discussing it, but do not really do much to develop and practice it.

Through this thesis I gained the general knowledge about Web architecture and technologies. Also, building the examples and the final prototype improved my Web programming skills.

I am quite satisfied with what I have done in this project. I have met some difficulties, but I managed to conquer them. At the beginning I knew only little about the semantic Web and now I am able to implement an ontology prototype using RDFs and OWL languages. In order to motivate myself I chose automotives as the subject of final prototype. As automotive is my favorite hobby, it was actually my interest that drove me to improve the ontology prototype better and carried me this far.

During this thesis, I learnt a new Web technology which will be the main stream in future, and I also enhanced my understanding of the XML language. I also got familiar with the procedures of carrying out a technical documentation.

Most important, researching the semantic Web really bordered my vision and made me to think how the Internet Webs have developed and changed our lives.

## LIST OF REFERENCES

1. Alesso H, Simith C, Thinking on Web, USA, 2008, 14
2. Antonou G, Harmelen F, Semantic Web Primer, MIT Press, USA, 2004, 17, 71, 80
3. BMJ Group, Semantic publishing: how to create richer metadata, <http://blogs.bmj.com/bmj-journals-development-blog/category/the-semantic-Web/>, Date of retrieval: 20.02.2012.
4. Breitman K, Casanova M, and Truszkowski W, Semantic Web, Technologies and applications, Springer, London, 2007
5. David Booth, Denotation as a Two-Step Mapping in Semantic Web Architecture, <http://dbooth.org/2009/denotation/>, Date of retrieval: 16.03.2012
6. FOFA Vocabulary Specification 0.9, Brickley, D., Miller, L., <http://smlns/foaf/o.1/>, Date of retrieval: 03.03.2012
7. IBM, XML Watch: Finding friends with XML and RDF, <http://www.ibm.com/developerworks/xml/library/x-foaf/index.html>, Date of retrieval: 03.03.2012
8. Pollock.J, Semantic Web for Dummies, Wiley, Indiana, 2009, 27-29
9. Resource Description Framework, <http://xml.coverpage.org/rdf.html>, Data of retrieval: 29.02.2012

10. Semantic Web Stack, [http://en.wikipedia.org/wiki/Semantic\\_Web\\_Stack](http://en.wikipedia.org/wiki/Semantic_Web_Stack),  
Date of retrieval: 27.02.2012
11. SW Arch: Same symbols, multiple languages,  
<http://www.w3.org/2006/Talks/0718-aaai-tbl/Overview.html#%2814%29>,  
Date of retrieval: 20.02.2012
12. W3 Schools, RDF Reference,  
[http://www.w3schools.com/rdf/rdf\\_reference.asp](http://www.w3schools.com/rdf/rdf_reference.asp), Date of retrieval:  
02.03.2012
13. W3C Semantic Web Frequently Asked Questions,  
<http://www.w3.org/RDF/FAQ>, Date of retrieval: 29.02.2012
14. W3C, Extensible Markup Language, <http://www.w3.org/XML>, Date of  
retrieval: 29.02.2012
15. W3C, OWL 2 Web Ontology Language, 2009, <http://www.w3.org/TR/owl-overview/#Introduction>, Date of retrieval: 13.03.2
16. W3C, OWL Web Ontology Language Semantics and Abstract Syntax,  
<http://www.w3.org/TR/owl-semantics/>, Date of retrieval: 06.03.2012
17. W3C, OWL Web Ontology Language, <http://www.w3.org/TR/owl-features/>, Date of retrieval: 04.03.2012
18. W3C, RDF Semantics, 2004, <http://www.w3.org/TR/rdf-mt/>, Date of  
retrieval: 10.03.2012.
19. W3C, RDF Triples, <http://www.w3.org/TR/rdf-concepts/#section-triples>,  
Date of retrieval: 01.03.2012

20.W3C, Semantic Web. <http://www.w3.org/2001/sw/>. Date of retrieval:  
27.01.2012

21.Wikipedia, Web Ontology Language,  
[http://en.wikipedia.org/wiki/Web\\_Ontology\\_Language](http://en.wikipedia.org/wiki/Web_Ontology_Language), Date of retrieval:  
05.03.2012

# APPENDICES

## 1. Cars\_example.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2
3 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5
6   <rdfs:Class rdf:ID="cars" rdfs:comment="The class of Cars" />
7
8   <rdfs:Class rdf:ID="sportcar" rdfs:comment="The class of sport car">
9     <rdfs:subClassOf rdf:resource="#cars" />
10  </rdfs:Class>
11
12  <rdfs:Class rdf:ID="suv" rdfs:comment="The class of SUV">
13    <rdfs:subClassOf rdf:resource="#cars" />
14  </rdfs:Class>
15
16  <rdfs:Class rdf:ID="sedan" rdfs:comment="The class of sedan">
17    <rdfs:subClassOf rdf:resource="#cars" />
18  </rdfs:Class>
19
20  <rdfs:Class rdf:ID="coupe" rdfs:comment="The class of Coupe">
21    <rdfs:subClassOf rdf:resource="#sportcar" />
22    <rdfs:subClassOf rdf:resource="#sedan" />
23  </rdfs:Class>
24
25  <rdfs:Property rdf:ID="price" rdfs:comment="It is a property of Cars">
26    <rdfs:domain rdf:resource="#cars" />
27  </rdfs:Property>
28
29 </rdf:RDF>
```

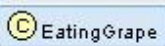
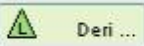




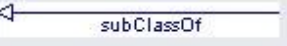
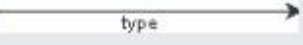
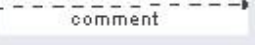
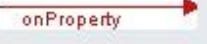


## 2. Cars\_example.xml in tree view

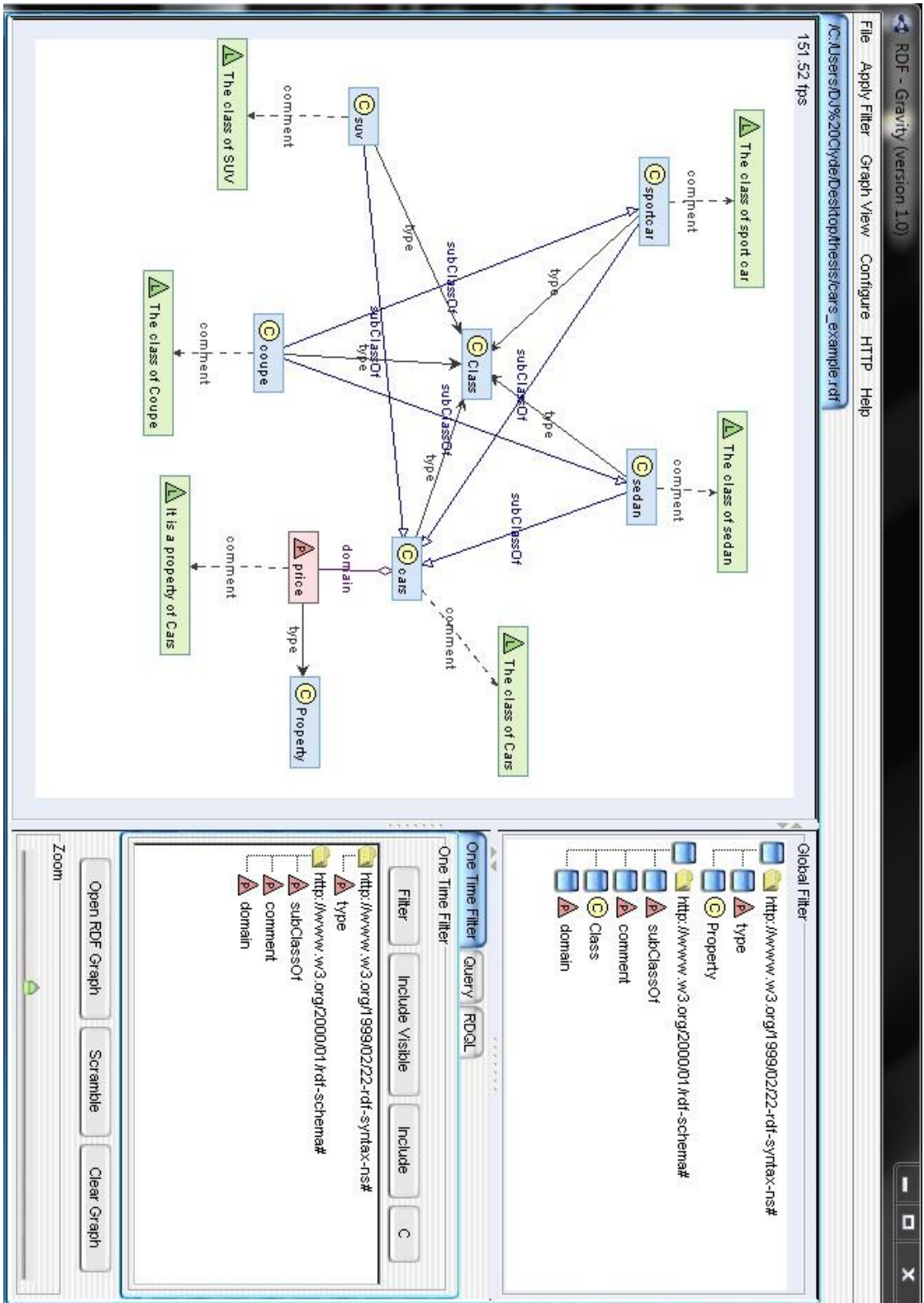
Tree View	XSL Output
[-] rdf:RDF	
xmlns:rdf	<a href="http://www.w3.org/1999/02/22-rdf-syntax-ns#">http://www.w3.org/1999/02/22-rdf-syntax-ns#</a>
xmlns:rdfs	<a href="http://www.w3.org/2000/01/rdf-schema#">http://www.w3.org/2000/01/rdf-schema#</a>
[-] rdfs:Class	
rdf:ID	cars
rdfs:comment	The class of Cars
[-] rdfs:Class	
rdf:ID	sportcar
rdfs:comment	The class of sport car
[-] rdfs:subClassOf	
rdf:resource	#cars
[-] rdfs:Class	
rdf:ID	suv
rdfs:comment	The class of SUV
[-] rdfs:subClassOf	
rdf:resource	#cars
[-] rdfs:Class	
rdf:ID	sedan
rdfs:comment	The class of sedan
[-] rdfs:subClassOf	
rdf:resource	#cars
[-] rdfs:Class	
rdf:ID	coupe
rdfs:comment	The class of Coupe
[-] rdfs:subClassOf	
rdf:resource	#sportcar
[-] rdfs:subClassOf	
rdf:resource	#sedan
[-] rdf:Property	
rdf:ID	price
rdfs:comment	It is a property of Cars
[-] rdfs:domain	
rdf:resource	#cars

### 3. General Notations

The table below provides the notations used the RDF Gravity tool for denoting RDF resources and properties.

	Refers to concepts or Classes. Here "EatingGrape" is a concept.
	Refers to a literal value (string, integer) etc.
	Refers to Anonymous nodes
	This refers to instances. Any anonymous node which has rdf:type property associated to a Class is also shown as an instance.
	Refers to Properties
	Refers to URI strings which cannot be identified as any of the above items
	Blue edges refer to rdfs:subClassOf property.
	Black edges refer to the rdf:type property.
	Dotted Edges refer to the rdf:comment property.
	Rest all other properties are marked as red in color.

#### 4. Cars\_example.rdf RDF graph



## 5. RDFa.html

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 TRANSITIONAL//EN"
2   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <head>
6   <meta http-equiv="content-type"
7     content="text/html; charset=iso-8859-1" />
8   <title>Author</title>
9 </head>
10 <body>
11
12   <div class="content"
13     about="http://localhost/RDFa.html"
14     instanceof="foaf:person">
15
16     <h1>
17       Infomation about
18       <span property="thesis:foaf_firstName">Pengfei</span>
19       <span property="foaf:family_name">Zhang</span>
20     </h1>
21
22     
27
28     <p>Author of This Thesis - Semantic Web. Student of OAMK.</p>
29     <p>
30       Email: <a href="mailto:r8zhpe00@students.oamk.fi">r8zhpe00@students.oamk.fi</a>
31     </p>
32     <p>
33       Phone: <a href="tel:+358401234567">+358 401234567</a>
34     </p>
35     <p>&copy; Clyde Zhang</p>
36   </div>
37 </body>
38 </html>
```

Hyper Text Markup Language file

## 6. Car ontology source code

```
*C:\Users\DJ Clyde\Desktop\thesis\Car_ontology.xml - Notepad++
Car_ontology.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE owl [
3    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
4  ]>
5  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
7        xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
8        xmlns:owl="http://www.w3.org/2002/07/owl#">
9
10     <owl:Ontology rdf:about="">
11       <owl:versionInfo>
12         My example version 1.0, 7 March 2012
13       </owl:versionInfo>
14     </owl:Ontology>
15
16     <owl:Class rdf:ID="Automotive">
17       <rdfs:comment>
18         Automotives form a class.
19       </rdfs:comment>
20     </owl:Class>
21
22     <owl:Class rdf:ID="Car">
23       <rdfs:comment>
24         Cars form a subclass of Automotive.
25       </rdfs:comment>
26       <rdfs:label>car</rdfs:label>
27       <rdfs:subClassOf rdf:resource="#Automotive" />
28     </owl:Class>
29
30     <owl:Class rdf:ID="AudiCar">
31       <rdfs:comment>
32         Audi cars are exactly those automotives
33         that are manufactured in Germany.
34       </rdfs:comment>
35       <owl:intersectionOf rdf:parseType="Collection">
36         <owl:Class rdf:about="#Automotive" />
37         <owl:Restriction>
38           <owl:onProperty rdf:resource="#manufactured-in" />
39           <owl:hasValue>
40             <xsd:string rdf:value="Germany" />
41           </owl:hasValue>
42         </owl:Restriction>
43       </owl:intersectionOf>
44     </owl:Class>
```

```

45 <owl:Class rdf:ID="PassengerCar">
46   <rdfs:comment>
47   Passenger cars are cars.
48   </rdfs:comment>
49   <rdfs:subClassOf rdf:resource="#car" />
50 </owl:Class>
51
52 <owl:Class rdf:ID="PersonalCar">
53   <rdfs:comment>
54   Passenger cars for personal use form
55   a subclass of Passenger Cars.
56   </rdfs:comment>
57   <rdfs:subClassOf rdf:resource="#PassengerCar" />
58 </owl:Class>
59
60 <owl:Class rdf:ID="AudiPassengerCar">
61   <rdfs:comment>
62   Audi passenger cars are Audi cars and passenger cars.
63   </rdfs:comment>
64   <rdfs:subClassOf rdf:resource="#PassengerCar" />
65   <rdfs:subClassOf rdf:resource="#AudiCar" />
66 </owl:Class>
67
68 <owl:Class rdf:ID="SportCar">
69   <rdfs:comment>
70   Sport cars are exactly those passenger cars
71   that use sport technology.
72   </rdfs:comment>
73   <owl:intersectionOf rdf:parseType="Collection">
74     <owl:Class rdf:about="#PassengerCar" />
75     <owl:Restriction>
76       <owl:onProperty rdf:resource="#SportTechnology" />
77       <owl:hasValue>
78         <xsd:string rdf:value="turbo" />
79       </owl:hasValue>
80     </owl:Restriction>
81   </owl:intersectionOf>
82 </owl:Class>
83
84 <owl:Class rdf:ID="AudiSportCar">
85   <rdfs:comment>
86   Audi sport cars are Audi passenger cars
87   and sport cars.

```

```

88     </rdfs:comment>
89     <rdfs:subClassOf rdf:resource="#SportCar" />
90     <rdfs:subClassOf rdf:resource="#AudiPassengerCar" />
91 </owl:Class>
92
93 <owl:Class rdf:ID="TTseries">
94     <rdfs:comment>
95     TTseries are Audi sport cars with 158hp to 335hp
96     horse power and 250Nm to 450Nm torque.
97     </rdfs:comment>
98     <rdfs:subClassOf rdf:resource="#AudiSportCar" />
99     <rdfs:subClassOf>
100         <owl:Restriction>
101             <owl:onProperty rdf:resource="#HorsePower" />
102             <owl:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
103                 158hp-335hp
104             </owl:hasValue>
105         </owl:Restriction>
106     </rdfs:subClassOf>
107     <rdfs:subClassOf>
108         <owl:Restriction>
109             <owl:onProperty rdf:resource="#Torque" />
110             <owl:hasValue>
111                 <xsd:string rdf:value="250Nm-450Nm" />
112             </owl:hasValue>
113         </owl:Restriction>
114     </rdfs:subClassOf>
115 </owl:Class>
116
117 <owl:Class rdf:ID="TTcoupe">
118     <rdfs:comment>
119     TT coupes belongs to the TT series
120     and price start with 45 327.20e.
121     </rdfs:comment>
122     <rdfs:subClassOf rdf:resource="#TTseries" />
123     <rdfs:subClassOf>
124         <owl:Restriction>
125             <owl:onProperty rdf:resource="#price" />
126             <owl:hasValue>
127                 <xsd:integer rdf:value="45 327.20" />
128             </owl:hasValue>
129         </owl:Restriction>
130     </rdfs:subClassOf>
131 </owl:Class>

```

```

133 <owl:Class rdf:ID="TTS">
134   <rdfs:comment>
135     TTS belongs to the TTs series
136     and price start with 76 848.94e.
137   </rdfs:comment>
138   <rdfs:subClassOf rdf:resource="#TTseries" />
139   <rdfs:subClassOf>
140     <owl:Restriction>
141       <owl:onProperty rdf:resource="#price" />
142       <owl:hasValue>
143         <xsd:integer rdf:value="76 848.94" />
144       </owl:hasValue>
145     </owl:Restriction>
146   </rdfs:subClassOf>
147 </owl:Class>
148
149 <owl:Class rdf:ID="TTRS">
150   <rdfs:comment>
151     TTRS belongs to the TT series
152     and price start with 99 666.35e.
153   </rdfs:comment>
154   <rdfs:subClassOf rdf:resource="#TTseries" />
155   <rdfs:subClassOf>
156     <owl:Restriction>
157       <owl:onProperty rdf:resource="#price" />
158       <owl:hasValue>
159         <xsd:integer rdf:value="99 666.35" />
160       </owl:hasValue>
161     </owl:Restriction>
162   </rdfs:subClassOf>
163 </owl:Class>
164
165 <owl:DatatypeProperty rdf:ID="manufactured-in">
166   <rdfs:domain rdf:resource="#Automotive" />
167   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
168 </owl:DatatypeProperty>
169
170 <owl:DatatypeProperty rdf:ID="price">
171   <rdfs:domain rdf:resource="#Automotive" />
172   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#nonNegativeInteger" />
173 </owl:DatatypeProperty>
174
175 <owl:DatatypeProperty rdf:ID="SportTechnology">
176   <rdfs:domain rdf:resource="#PassengerCar" />
177   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
178 </owl:DatatypeProperty>
179
180 <owl:DatatypeProperty rdf:ID="HorsePower">
181   <rdfs:domain rdf:resource="#PassengerCar" />
182   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
183 </owl:DatatypeProperty>
184
185 <owl:DatatypeProperty rdf:ID="Torque">
186   <rdfs:domain rdf:resource="#PassengerCar" />
187   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
188 </owl:DatatypeProperty>
189
190 </rdf:RDF>

```



# 7. Car\_ontology.rdf RDF graph

The screenshot displays an RDF graph viewer window titled "RDF - Gravity (version 1.0)". The main area shows a dense network of nodes and edges representing the Car\_ontology.rdf. Nodes are represented by colored boxes with labels such as "PassengerCar", "SportCar", "PersonalCar", "TTS", "Automotive", "Germany", and "TTS belongs to". Edges represent properties like "type", "subClassOf", "hasValue", and "onProperty".

At the bottom of the window, there are two filter panels:

- Global Filter:** A list of nodes with checkboxes, including "first", "value", "rest", "http://www.w3.org/1999/02/22-rdf-syntax-ns#nil", "http://www.w3.org/2001/XMLSchema#", "http://www.w3.org/2002/07/owl#Restriction", "http://www.w3.org/2002/07/owl#", "hasValue", "versionInfo", "intersectionOf", and "http://www.w3.org/2002/07/owl#Class".
- One Time Filter:** A list of nodes with checkboxes, including "type", "first", "value", "rest", "http://www.w3.org/2001/XMLSchema#", "http://www.w3.org/2002/07/owl#", "hasValue", "versionInfo", "intersectionOf", "onProperty", "http://www.w3.org/2000/01/rdf-schema#", "subClassOf", and "comment".

Navigation buttons at the bottom include "Open RDF Graph", "Scramble", "Clear Graph", and "Zoom".