# The Web Ontology Language (OWL) and its Applications

*Jorge Cardoso*

*University of Coimbra, Portugal; KSRI/Karlsruhe Institute of Technology, Germany*

*Alexandre Miguel Pinto*

*University of Coimbra, Portugal*

## Introduction

Initially, the WWW was primarily composed of documents written in HTML (hyper text markup language), a language that is useful for visual presentation. HTML is a set of "markup" symbols contained in a Web page intended for display on a Web browser. At a later stage the distinct aspects of web page presentation style became separated from the contents, the former being specified in CSS (cascading style sheets). Still, most of the information on the Web is designed only for human consumption. Humans can read Web pages and understand them, but their inherent meaning is not shown in a way that allows their interpretation by computers (Cardoso & Sheth, 2006).

Since this setting does not allow computers to understand the meaning of Web pages (Cardoso, 2007), the W3C (World Wide Web Consortium) started to work on the concept of Semantic Web (SW) with the objective of developing approaches and solutions for data integration and interoperability purpose. The goal was to develop ways to allow computers to understand Web information.

One of the corner stones of the SW is the Web Ontology Language (OWL). OWL can be used by applications that need to understand and to reason over the meaning of information instead of just parsing data for display purposes. The aim of this chapter is to present OWL, which can be used to develop Semantic Web applications that understand information and data on the Web. This language was proposed by the W3C and was designed for publishing, sharing data, and automating data understood by computers using ontologies. To fully comprehend OWL we need first to study its origin, its logic (semantic) foundations, and the basic blocks of the language. Therefore, we start by briefly introducing XML (eXtensible Markup Language), RDF (Resource Description Framework), and RDF Schema (RDFS). These concepts are important since one of the main syntaxes OWL takes is the XML format, and also because conceptually OWL is an extension of RDF and RDFS.

## Background: The Semantic Web Stack

The Semantic Web identifies a set of technologies and standards that form the basic building blocks of an infrastructure that supports the vision of the meaningful Web. Figure 1 illustrates the different parts of the SW architecture. It starts with the foundation of URI (universal resource identifier) and Unicode. URI is a formatted string that serves as a means of identifying abstract or physical resources. For example, http://eden.dei.uc.pt/~jcardoso/ identifies the location from where a Web page can be retrieved and urn:isbn:3-540-24328-3 identifies a book using its ISBN. Unicode provides a unique number for every character, independent of the underlying platform, program, or language.

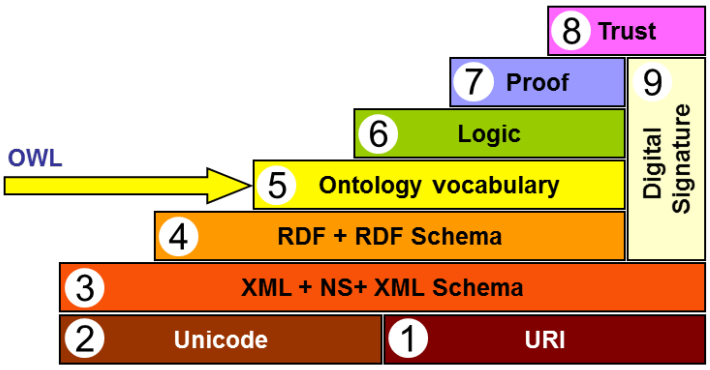


Figure 1. Semantic Web layered architecture (Berners-Lee et al., 2001)

Directly above URI and Unicode we find the syntactic interoperability layer in the form of XML, which in turn underlies RDF and RDFS. Web ontology languages are built on top of RDF and RDFS. The last three layers are logic, proof, and trust, of which the top one has not been significantly explored yet. Some of the layers rely on the digital signature component to ensure security.

Next we briefly describe the most basic layers (XML, RDF, and RDFS). While the notions presented have been simplified, they give a reasonable conceptualization of some of the simplest components of the SW.

### XML

The extensible markup language (XML) (XML, 2007) was originally pictured as a language for defining new document formats for the WWW. An important feature of this language is the separation of content from presentation, which makes it easier to select and/or reformat data. XML is a text-based format that provides mechanisms for describing document structures using markup tags (words surrounded by '<' and '>'). Both HTML and XML representations use tags such as <h1> or <name>, and information between those tags, referred to as the content of the tag. However, there are significant differences between HTML and XML. XML is case sensitive while HTML is not; and HTML has predefined elements and attributes whose behavior is well specified, while XML does not. Instead, users can create their own XML vocabularies that are specific to their application or business needs.

The following example shows an XML document identifying a 'Contact' resource. The document includes various metadata markup tags, such as <first_name>, <last_name>, and <e-mail>, which provides various details about a contact.

```
<Contact contact_id="1234">
    <first_name> Jorge </first_name>
    <last_name> Cardoso </last_name>
    <organization> University of Coimbra </organization>
    <email> jcardoso@dei.uc.pt </email>
    <phone> +351 239 790 051 </phone>
</Contact>
```

While XML has gained awareness, XML is simply a way of standardizing data formats. But from the point of view of semantic interoperability, XML has restrictions. E.g., there is no way to recognize the semantics of a particular domain because XML aims at a document structure and enforces no common interpretation of the data. Although XML is simply a data-format standard, it is part of a set of technologies that constitute the foundations of the SW.

**RDF**

The Resource Description Framework (RDF) (RDF, 2002) was developed by the W3C to provide a common way to describe information so it could be read and understood by computer applications. RDF was designed using XML as the underlying syntax. It provides a model for describing resources on the Web. A resource is an element (document, Web page, printer, user, etc.) on the Web that is uniquely identifiable by a URI. The RDF model is based upon the idea of making statements about resources in the form of a subject-predicate-object expression, a 'triple' in RDF terminology.

- *Subject* is the resource, the thing that is being described;
- *Predicates* are aspects about a resource, and expresses the relationship between the subject and the *object*;
- *Object* is the value that is assigned to the predicate.

RDF has a very limited set of syntactic constructs – only triples are allowed. Every RDF document is equivalent to an unordered set of triples. E.g., the RDF triple that describes the statement:

*"The creator of the page named Research is Jorge Cardoso."*

is

st = (http://eden.dei.uc.pt/~jcardoso/Research/research.html, Creator, JorgeCardoso)

Here, 'http://eden.dei.uc.pt/~jcardoso/Research/research.html' is a resource with the property 'Creator,' having the value 'JorgeCardoso'.

The statement can also be graphically represented as in Figure 2.

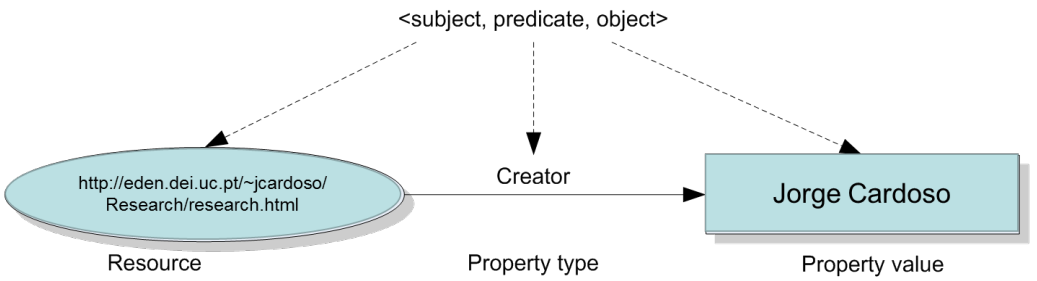


Figure 2. RDF graph

One way to represent the statement in Figure 2 using RDF is:

```
<? xml version="1.0" ?>
<RDF xmlns = "http://w3.org/TR/1999/PR-rdf-syntax-19990105#"
       xmlns:DC = «http://dublincore.org/2003/03/24/dces#»>
       <Description about =
          "http://eden.dei.uc.pt/~jcardoso/Research/research.html">
       <DC:Creator>Jorge Cardoso</DC:Creator>
   </Description>
</RDF>
```

Here, the first lines use namespaces to explicitly define the meaning of the notions that are used. The first namespace xmlns:rdf="http://w3.org/TR/1999/PR-rdf-syntax-19990105#" refers to the

document describing the syntax of RDF. The second "http://dublincore.org/2003/03/24/dces#" refers to the description of the Dublin Core (DC) (DC, 2005), a basic ontology about authors and publications.

### RDF Schema

RDF Schema (RDFS) is semantically richer when compared to RDF. RDFS describes the resources with classes, properties, and values. RDFS associates the resources in classes, states the relations between these classes, declares properties and specifies the domain and range of these properties.

Classes in RDFS are much like classes in object oriented programming languages. These allow resources to be defined as instances of classes, and subclasses of classes. Properties can be seen as attributes that are used to describe the resources by assigning values to them. RDF is used to assert property-related statements about objects, and RDFS can extend this capability by defining the class domain and the class range of such properties (however, RDFS has some limitations but these have been resolved with the introduction of OWL).

## THE WEB ONTOLOGY LANGUAGE

The Web Ontology Language (OWL) (OWL 2, 2012) is one of the most important ontology languages. In this respect, it is more expressive than XML, RDF or RDFS by providing additional vocabulary along with formal semantics.

### OWL Flavors

Since 2009 there is a second version of OWL, dubbed "OWL 2", which supersedes the previous "OWL 1" — in 2012 OWL 2 became the official recommendation by W3C. OWL 1 exists in different sublanguages: OWL Lite, OWL 1 DL, and OWL Full. An important feature of each sublanguage is its expressiveness. OWL Lite is the least expressive and the OWL Full is the most expressive sublanguage. OWL 1 DL is more expressive that OWL Lite but less expressive than OWL Full. This entails that every legal OWL Lite ontology/conclusion is a legal OWL DL ontology/conclusion; every legal OWL DL ontology/conclusion is a legal OWL Full ontology/conclusion. OWL 2 has a similar gradation of sublanguages, including OWL 2 DL (which is more expressive than OWL 1 DL), but instead of OWL 2 Lite there are three distinct profiles (OWL 2 profiles) OWL 2 QL, OWL 2 RL, and OWL 2 EL.

**OWL Full** is the most expressive of the OWL sublanguages and it uses the entire OWL language primitives. It is intended to be used in situations where very high expressiveness is more important than the guarantee of decidability or computational completeness of the language. This sublanguage is meant for users who want maximum expressiveness and the syntactic freedom of RDF, but with no computational guarantees.

**OWL 2 DL** is a sublanguage of OWL Full that restricts the application of OWL and RDF constructors. OWL 2 DL (DL stands for description logics) is not compatible with RDF, in the same way that not every RDF document is a legal OWL DL document, although every legal OWL 2 DL document is a legal RDF document. This sublanguage supports those users who want the maximum expressiveness without losing computational completeness and decidability. OWL 1 DL is a subset of OWL 2 DL.

**OWL 2 EL** is designed as a subset of OWL 2 DL that is particularly suitable for applications employing ontologies that define very large numbers of classes and/or properties, captures the expressive power used by many such ontologies, and for which ontology consistency, class expression subsumption, and instance checking can be decided efficiently (in polynomial time).

**OWL 2 QL** contains the intersection of RDFS and OWL 2 DL. It is designed so that data that is stored in a standard relational database system can be queried through an ontology via a simple rewriting mechanism, i.e., by rewriting the query into an SQL query that is then answered by the RDBMS system,

without any changes to the data. OWL 2 QL is based on the DL-Lite family of description logics and it was designed so that sound and complete query answering is computationally efficient.

**OWL 2 RL** is aimed at applications that require scalable reasoning without sacrificing too much expressive power. It is designed to accommodate both OWL 2 DL applications that can trade the full expressivity of the language for efficiency, and RDF(S) applications that need some added expressivity from OWL 2. The design of OWL 2 RL was inspired by Description Logic Programs and its expressiveness was achieved by defining a syntactic subset of OWL 2 which is amenable to implementation using rule-based technologies.

**OWL 1 Lite** is the simplest sublanguage, intended to be used in situations where only a simple class hierarchy and constraints are needed.

Every OWL 1 Lite, OWL 2 EL/QL/RL ontology or conclusion is a legal OWL 2 DL ontology or conclusion, but not the inverse, and so on for OWL 2 DL and OWL Full, as showed in Figure 3:

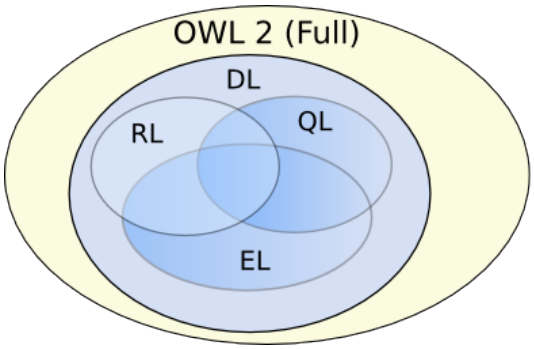


Figure 3. OWL sublanguages

The choice between OWL 2 EL/QL/RL and OWL 2 DL depends on whether the simple constructs of OWL 2 EL/QL/RL are sufficient. The choice between OWL 2 DL and OWL Full depends on how important it is to be able to carry out automated reasoning on the ontology versus to use highly expressive and powerful modeling facilities.

## OWL Syntax and Semantics

In this section we describe the syntax and semantics of the main constructs of OWL. We illustrate step-by-step how to build an ontology using OWL. We also explain how to define the header of an ontology, its classes, properties, and relationships. After reading this section the reader should be able to recognize an ontology written in OWL and identify some of its components.

The semantics of ontologies written in OWL is grounded in First Order Logic (FOL). Indeed, the various flavors of OWL, and most Description Logics, are but syntactic variations over fragments of FOL where objects, sets (or classes) of objects and their relationships can be formally described along with the corresponding constraints.

### *Header*

The first element in an OWL document is an rdf:RDF element which specifies a set of XML namespace's that provide a means to unambiguously interpret identifiers and make the rest of the ontology presentation more readable. E.g.,

```
<rdf:RDF
    xmlns="http://eden.dei.uc.pt/~jcardoso/owl/RUD.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xml:base="http://eden.dei.uc.pt/~jcardoso/owl/RUD.owl">
```

### *Information Version*

After the namespace declaration, an OWL document specifies a collection of assertions that are grouped under an owl:ontology element and offers details about the ontology:

- **owl:versionInfo:** Provides information about the current ontology.
- **owl:priorVersion:** Indicates an earlier version of the current one.
- **owl:backwardCompatibleWith:** Contains a reference to an ontology that is a prior version of the containing ontology that is backward compatible with it.
- **owl:incompatibleWith:** Indicates the containing ontology is not backward compatible with the referenced ontology.
- **owl:imports:** Only this assertion has a formal meaning to the ontology and represents a set of other ontologies that are considered to be part of the current ontology.

A simple example:

```
<rdf:RDF>
...
<owl:Ontology rdf:about="">
    <rdfs:comment> University Ontology </rdfs:comment>
    <owl:versionInfo> v.1 2013-11-18 </owl:versionInfo>
    <owl:priorVersion>
    <owl:Ontology rdf:about = "http://eden.dei.uc.pt/~jcardoso/owl/RUD.owl"/>
    </owl:priorVersion>
    <rdfs:label> University Ontology </rdfs:label>
</owl:Ontology>
...
</rdf:RDF>
```

### *Class Element*

Classes are collections of individuals. They are defined in an OWL document with the owl:Class element. The formal semantics of an OWL class is a set in FOL and they correspond to predicates with arity 1. E.g., the class "Teacher" can be defined as

```
<owl:Class rdf:ID="Teacher">
    <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
```

The rdf:ID element defines the name of the class. If we want to make reference to a class we use the rdf:resource element. An OWL ontology can represent the hierarchy between classes using the element owl:subClassOf. E.g., the class "Teacher" is a subclass of "Person." The semantics of this statement is $\forall x\ Teacher(x) \Rightarrow Person(x)$ in FOL.

It is possible to establish relations between two classes, e.g., using owl:equivalentClass and owl:disjointWith elements. The assertion owl:equivalentClass applied to classes A and B, represents that class A has the same individuals as class B. E.g., the class "faculty" is equivalent to the "academicStaffMember" class:

```
<owl:Class rdf:ID="faculty">
   <owl:equivalentClass rdf:resource="#academicStaffMember"/>
</owl:Class>
```

The owl:disjointWith element applied on two classes A and B specifies that class A and B are disjoint, i.e., classes A and B have no member in common. E.g., a "Full Professor" cannot be an "Associate Professor" at the same time.

```
<owl:Class rdf:about="#AssociateProfessor">
   <owl:disjointWith rdf:resource="#FullProfessor"/>
</owl:Class>
```

### *Complex Class*

Another way to create (more complex) classes in OWL is to combine simple classes using Boolean operators (union, intersection, and complement). The owl:unionOf element applied on classes A and B creates a new class that contains all members from class A and B. For example, the combination of the class "staff members" and the class "student" creates the new class "peopleAtUni."

```
<owl:Class rdf:ID="peopleAtUni">
   <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#staffMember"/>
      <owl:Class rdf:about="#student"/>
   </owl:unionOf>
</owl:Class>
```

The owl:intersectionOf element creates a new class from the two classes A and B which has elements that were both in class A and class B:

```
   <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#faculty"/>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#belongsTo"/>
         <owl:hasValue rdf:resource="#CS_Department"/>
      </owl:Restriction>
   </owl:intersectionOf>
</owl:Class>
```

Individuals of the new class created above are those that are members of both the classes "faculty" and the anonymous class created by the restriction on the property "belongsTo."

The owl:complement element selects all individuals from the domain that do not belong to a certain class:

```
<owl:Class rdf:about="#course">
   <rdfs:subClassOf>
      <owl:Class>
```

```
            <owl:complementOf rdf:resource="#staffMember"/>
        </owl:Class>
    </rdfs:subClassOf>
</owl:Class>
```

The class "course" above has as its members all individuals that do not belong to the "staffMember" class.

### *Property*

Properties let us describe a kind of relationship between members of classes. They correspond to predicates with arity 2 in FOL. In an OWL document two types of properties are distinguished:

- Object properties, which relate objects to other objects, i.e. instances of a class with instances of another class. E.g., the object property "isTaughtBy" relates the class "course" with the class "academicsStaffMember." I.e., a "course" "isTaughtBy" an instance of the "academicStaffMember" class.

```
<owl:ObjectProperty rdf:ID="isTaughtBy">
    <owl:domain rdf:resource="#course"/>
    <owl:range rdf:resource="#academicStaffMember"/>
</owl:ObjectProperty>
```

- Datatype properties, which relate objects to data type values. OWL does not have predefined data types, but it allows one to use the XML Schema data types. E.g., the year in which a person was born is specified using the "http://www.w3.org/2001/XLMSchema#nonNegativeInteger" data type from the XML Schema.

```
<owl:DatatypeProperty rdf:ID="ageYear">
    <rdfs:domain rdf:resource="#Person"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XLMSchema#nonNegativeInteger"/>
</owl:DatatypeProperty>
```

Both kinds of properties can use the rdfs:domain and rdfs:range element to restrict the relation.

### *Property Restrictions*

More elaborate boundaries can be made by applying restrictions to a property, this results in the subclasses of individuals that satisfy that condition. There are two kinds of restrictions: values constraints and cardinality constraints. Examples of values constraints include owl:allValuesFrom, owl:someValuesFrom, and owl:hasValues.

**owl:allValuesFrom**: Defines the set of individuals, for which **all the values** of the restricted property are instances of a certain class:

```
<owl:Class rdf:about="#firstYearCourse">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#isTaughtBy"/>
            <owl:allValuesFrom rdf:resource="#professor"/>
        </owl:Restriction>
    </rdfs:subClassOf>
```

```
</owl:Class>
```

Here, the individuals that are members of the class “firstYearCourse” are all the courses that have the property “isTaughtBy” assigned to a “professor”

**owl:someValuesFrom**: Defines the set of individuals that have **at least one** relation with an instance of a certain class, e.g.:

```
<owl:Class rdf:about=”#academicStaffMember”>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource=”#teaches”/>
         <owl:someValuesFrom rdf:resource=”#undergraduateCourse”/>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

**owl:hasValues**: Defines a set of individuals for which the value of the restricted property is equal to a certain instance. E.g., the individuals of the class “mathCourse” can be characterized as those that are taught by the professor “949352:”

```
<owl:Class rdf:about=”#mathCourse”>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource=”#isTaughtBy”/>
         <owl:hasValues rdf:resource=”#949352”/>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

Cardinality constraints point out how many times the property can be used on an instance. Examples include owl:maxCardinality, owl:minCardinality, and owl:cardinality.

**owl:maxCardinality**: Defines the set of individuals that have **at most** N distinct values of the property concerned. E.g., we can specify that the class “department” has at most 30 members:

```
<owl:Class rdf:about=”#department”>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource=”#hasMember”/>
            <owl:maxCardinality rdf:datatype=”&xsd;nonNegativeInteger”>30
         </owl:maxCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

**owl:minCardinality**: Defines the set of individuals that have **at least** N distinct values of the property concerned. E.g., a course must be taught at least by one teacher:

```
<owl:Class rdf:about=”#course”>
   <rdfs:subClassOf>
```

```
        <owl:Restriction>
            <owl:onProperty rdf:resource="#isTaughtBy"/>
                <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1
            </owl:minCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

**owl:cardinality:** Defines the set of individuals that have **an exact** number of distinct values of the property concerned. This element is used to specify a precise number, i.e., to express that a property has a minimum cardinality which is equal to the maximum cardinality.

### *Properties and their Characteristics*

Properties' characteristics add more expressivity to OWL. owl:equivalentProperty and owl:inverseOf elements are some examples. owl:equivalentProperty states a property that relates the same subject and object pairs as another property. E.g., the property "lectures" is equivalent to "teaches" and in OWL this can be represented as:

```
<owl:ObjectProperty rdf:ID="lectures">
    <owl:equivalentProperty rdf:resource="#teaches"/>
</owl:ObjectProperty>
```

The owl:inverseOf element is used to define an inverse relation between properties. If property P' is the inverse of property P, then X is related to Y by P, iff Y is related to X by the P'. E.g., "teacher teaches a course" is the inverse of "a course is taught by a teacher." In OWL:

```
<owl:ObjectProperty rdf:resource="#teaches">
    <owl:inverseOf rdf:resource="#isTaughtBy"/>
</owl:ObjectProperty>
```

Properties have some characteristics that can be defined directly:

- **Functional property:** Functional properties (owl:FunctionProperty) have at most one value for each instance.
- **InverseFuncionalProperty:** Inverse functional properties (owl:InverseFuncionalProperty) are properties whose inverse property is functional. E.g., a property such as "#isMotherOf" is inverse functional.
- **Transitive property:** The transitive property is understood as: if the pair (x, y) is an instance of the transitive property P, and the pair (y, z) is an instance of P, we can infer the pair (x, z) is also an instance of P.
- **Symmetric property:** The symmetric property (owl:SymmetricProperty) is interpreted as follows: the pair (x, y) is an instance of A iff the pair (y, x) is also an instance of A.

The following example illustrates the owl:SymmetricProperty and owl:TransitiveProperty elements.

```
<owl:ObjectProperty rdf:ID="hasSameGradeAs">
    <rdf:type rdf:resource="&owl;TransitiveProperty"/>
    <rdf:type rdf:resource="&owl;SymmetricProperty"/>
    <rdfs:domain rdf:resource="#student"/>
```

```
    <rdfs:range rdf:resource="#student"/>
</owl:ObjectProperty>
```

## Tools

Special ontology editors, like Protégé (Protégé 2013) and TopBraid Composer (TopBraid Composer 2013), have been developed and are used both in the industry and academic environments. These editors ease the burden of writing the complex syntax of OWL allowing the user to create and edit ontologies in a visual manner.

Given the logical statement nature of the knowledge represented with ontologies, traditional relational databases are not the ideal storage and query platform for OWL, and indeed RDFS. Knowledge in OWL is represented as sets of <subject, predicate, object> triples and these are most efficiently stored and accessed in dedicated triple stores, such as Jena TDB (Jena TDB 2013), and AllegroGraph (AllegroGraph, 2013). Likewise, querying of triple stores is done via specific query languages: the current standard language for querying RDF(S) is SPARQL (SPARQL 1.1, 2013), and this is also the current standard for querying OWL, although SPARQL cannot take full advantage of the expressivity of OWL. Other languages, like SQWRL (SQWRL, 2013), have been recently proposed for querying OWL. SWQRL takes advantage of OWL's expressive power and also of semantic web rule-based languages such as SWRL (SWRL, 2004), but is not yet a standard.

## Applications

Nowadays, several companies and projects rely on semantics to implement more advanced applications. Examples include semantic wikis, semantic social networks, semantic blogs, and semantic product descriptions. One particularly interesting application was the use of semantics to develop the language called Linked USDL (Unified Service Description Language) (Cardoso et al, 2012) which enables to describe services (e.g. from healthcare, education, and cloud computing). It was the emergence of services in society that triggered the development of the language to automate, improve, and support the delivery of digital services. It embraces the use of formal ontology representations to capture the semantics of services in such a way that they are amenable for automated reasoning.

Linked USDL was developed for describing business, software, or real world services using computer-readable and computer-understandable ontologies to make services tradable on the Internet. It provides a comprehensive business envelope to foster the effective commercialization of more complex services. The following example provides an extract of a cloud service described using Linked USDL.

```
:bentity_SugarCRM_Inc a gr:BusinessEntity ;
    foaf:homepage <http://www.sugarcrm.com/> ;
    foaf:logo <http://sugarcrm.com/images/...> ;
    gr:hasISICv4 "620"^^xsd:string ;
    gr:hasNAICS "511210"^^xsd:string ;
    gr:legalName "SugarCRM Inc." ;
    gr:taxID "77123/12345"^^xsd:string ;
    ...
:model_SugarCRM a usdl:ServiceModel ;
    dcterms:title "SugarCRM Model for Software"@en .
    ...
```

```
:offering_SugarCRM a usdl:ServiceOffering ;
    usdl:includes <#service_SugarCRM> .
    ...
    foaf:page <http://www.sugarcrm.com/products> ;
    gr:name "SugarCRM Systems";
    gr:description """SugarCRM is a CRM systems provided as a SaaS solutions.""";
    usdl:price :price_SugarCRM ;
    usdl:legal :legal_SugarCRM ;
    usdl:includes :service_SugarCRM .
    ...
```

The extract starts by describing the business entity which provides the service. In this case, it is a company called SurgarCRM. The codes for the type of service provided are set by using NAICS and ISIC, two widely used classification schemas for business activities. Afterwards, a service offering associates a price structure and legal constraints with a service to provide a concrete offering to customers. Compared to previous efforts, Linked USDL enables to provide a business-orientation on services by modeling co-creation, pricing, legal aspects, and security issues which are all elements that must be part of service descriptions. Once services are modeled, as shown in this example, they are typically made available in marketplaces. Customers can, then, use automated tools to browse, query, and search for the most appropriate services based on their preferences and requirements.

## Future trends

Many researchers worldwide have recognized that the Semantic Web (or Web3.0) is the key to develop a new generation of information systems. The number of international conferences organized every year on this topic clearly shows the interest and importance of this technology. In this context, OWL is the most widespread language to develop Semantic Web-based applications. OWL has been used in many areas. Gartner names Semantic Technologies to its top technology trends impacting information infrastructure in 2013. It estimates that the sales of electronic data discovery software, possible due to semantic technologies, will reach $2.9 billion by 2017.

One of today's hottest IT topics is information integration. This requires merging and mapping information from different data sources and preparing them in a way that end users can use it. But this isn't quite easy. Ontologies and logic enable and support these approaches. By using mappings, the data structures are linked to a uniform ontology, providing a common, holistic view on the data. Combining ontologies with existing business logic makes new business insights available. Information integration makes processes easier, faster, less expensive and more understandable for end users.

Moreover, programming the Semantic Web with OWL can reduce and eliminate terminological and conceptual confusion by defining a shared understanding, that is, a unifying framework enabling communication and cooperation amongst people in reaching a better inter-enterprise organization. Presently, one of the most important roles ontology plays in communication is that it provides unambiguous definitions for terms used in a software system, but semantics needs to be applied rapidly to human integration to enable communication, coordination, and cooperation. The use of ontologies for improving communication has already been shown to work in practice.

## Conclusion

The Semantic Web is no longer a future vision, but increasingly a current reality of the Web where information has a precise meaning. This enables computers and people to work better in cooperation. To make possible the creation of the Semantic Web it is important to have a language that allows the

description of concepts of a given domain and the creation of ontologies. One of the most prominent ontology languages to achieve those two characteristics is OWL (Web Ontology Language) which can be used to develop Semantic Web applications. These applications constitute a new wave of enhanced systems that understand the domain in which they are working and with which they interact. Thus, OWL enables the Web to be a global infrastructure for sharing both documents and data, which makes searching and reusing information easier and more reliable as well. OWL can be used by applications to improve search engines on the Web and tools to manage knowledge. In this chapter we have laid out the foundations of the Semantic Web, its associated languages and standards. These elements are the basic building blocks of any Semantic Web application.

## References


AllegroGraph (2013). AllegroGraph. Retrieved October 29, 2013, from http://www.franz.com/agraph/allegrograph/.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. Scientific American, 279(5), 34–43.

Campanini, S.E., Castagna, P. and Tazzoli, R. (2004). Platypus wiki: a semantic wiki wiki web. In semantic Web Applications and Perspectives. In proceedings of 1st Italian semantic Web Workshop. Cardoso, J. (2007). Semantic Web Services: Theory, Tools and Applications. New York, NY, USA, IGI Global, ISBN:978-1-59904-045-5.

Cardoso, J., Pedrinaci, C., Leidig, T., Rupino, P. & Leenheer, P. D. (2012). Open semantic service networks. In The International Symposium on Services Science (ISSS 2012), pages 1-15, Leipzig, Germany, 2012.

Cardoso, J., & Sheth, A. (2005). Semantic Web Process: powering next generation of processes with semantics and Web services. Heidelberg, Lecture Notes in Computer Science, Springer-Verlag, Vol. 3387.

Cardoso, J., & Sheth, A. (2006). Semantic web services, processes and applications. Springer.

Cayzer, S., & Shabajee, P. (2003). Semantic Blogging and Bibliography Management. Blogtalk the First European Conference on Weblogs (Blogtalk 2003) Vienna, Austria.

DC. (2005). The Dublin Core Metadata Initiative. Retrieved May 9, 2007, from http://dublincore.org/.

HTML. (2007). Hyper Text Markup Language. Retrieved May 9, 2007, from http://www.w3.org/html/.

Jena TDB. (2013). Jena TDB. Retrieved October 29, 2013, from http://jena.apache.org/documentation/tdb/index.html.

OWL 2. (2012). Web Ontology Language (OWL). Retrieved October 28, 2013, from http://www.w3.org/TR/owl2-overview/-

OWL 2 profiles. (2012). Web Ontology Language (OWL). Retrieved October 28, 2013, from http://www.w3.org/TR/owl2-profiles/

Protégé. (2013). Protégé. Retrieved October 29, 2013, from http://protege.stanford.edu/.

RDF. (2002). Resource Description Framework (RDF). Retrieved May 9, 2007, from http://www.w3.org/RDF/.

SPARQL 1.1. (2013). SPARQL 1.1 Overview. Retrieved October 29, 2013, from http://www.w3.org/TR/sparql11-overview/.

SQWRL (2013). SQWRL. Retrieved October 29, 2013, from http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL.

SWRL (2004). SWRL. Retrieved October 29, 2013, from http://www.w3.org/Submission/SWRL/.

TopBraid Composer (2013). TopBraid Composer. Retrieved October 29, 2013, from http://www.topquadrant.com/products/TB_Composer.html.

XML. (2007). Extensible Markup Language (XML). Retrieved May 9, 2007, from http://www.w3.org/XML/.

XMLSchema. (2005). XML Schema. Retrieved May 9, 2007, from http://www.w3.org/XML/Schema.

## Key Terms

**Linked USDL:** It is a master schema constructed using semantic web technologies and linked data principles. It is used to describe typical services such as the ones found in healthcare, education, and cloud computing environments.

**Ontology:** A description of concepts and relationships that can be used by people or software agents who want to share information within a domain.

**OWL:** A markup language for publishing and sharing data using ontologies on the Internet. OWL is a vocabulary extension of the RDF and is derived from the DAML+OIL Web Ontology Language.

**RDF:** Resource description framework is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata model using XML but which has come to be used as a general method of modeling knowledge, through a variety of syntax formats.

**RDFS:** RDF schema is an extensible knowledge representation language, providing basic elements for the definition of ontologies, otherwise called RDF vocabularies, intended to structure RDF resources.

**Semantic Web:** SW provides a common framework that allows data to be shared and reused across applications, enterprises, and community boundaries. It is a collaborative effort led by W3C with the participation of a large number of researchers and industrial partners.

**XML:** Extensible markup language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). XML is a standard for data interchanged on the Web, allowing for the structuring of data but without meaning.