

Esperanto Services
IST-2001-34373



Deliverable

D21

State of the art on Semantic Web Languages

Ying Ding
Sinuhé Arroyo

Institut für Informatics
University of Innsbruck

{ying.ding, sinuhe.arroyo} @uibk.ac.at

22-01-2003

Executive Summary

This deliverable delineates the state of the art of Semantic Web languages, which include SHOE, Ontobroker, XML(s), RDF(s), OIL, DAML+OIL, OWL and other related languages (DAML-S and Topic Maps) and technology.







During the running of the project, this work package will be responsible for providing updated information about the evolution of these languages and technologies.

Document Information

IST Project Number	IST-2001-34373	Acronym	Esperanto Services
Full title	Application Service Provision of Semantic Annotation, Aggregation, Indexing and Routing of Textual, Multimedia, and Multilingual Web Content		
Project URL	www.esperanto.net		
Document URL			
EU Project officer	Brian Macklin		

Deliverable	Number	21	Name	State of the art in Semantic Web Languages			
Task	Number		Name				
Work package	Number	2					
Date of delivery	Contractual	30-10-2002	Actual	22-01-2003			
Code name			Status	draft <input type="checkbox"/>	final <input checked="" type="checkbox"/>		
Nature	Prototype <input type="checkbox"/> Report <input checked="" type="checkbox"/> Specification <input type="checkbox"/> Tool <input type="checkbox"/> Other						
Distribution Type	Public <input checked="" type="checkbox"/> Restricted <input type="checkbox"/> Consortium						
Authors (Partner)	Ying Ding (IFI), Dieter Fensel (IFI), Sinuhé Arroyo (IFI)						
Contact Person	Ying Ding						
	Email	Ying.ding@uibk.ac.at	Phone	+43 512 507 6112	Fax	+43 512 507 9872	
Abstract (for dissemination)	This deliverable describes the state of the art on Semantic Web Languages developed within the Esperanto project.						
Keywords	OWL, RDF, RDFS Semantic Languages, OIL, DAML+OIL						
Version log/Date	Change			Author			
30-08-2002	First Version			Dieter Fensel			
20-10-02	Second revision			Ying Ding			
22-01-2003	Third revision			Sinuhé Arroyo			
17-02-2003	Internal Quality Proof			Sinuhé Arroyo			

Project Information

Partner	Acronym	Contact
Intelligent Software Components S.A. (Coordinator)	iSOCO 	Dr. V. Richard Benjamins c/ Francisca Delgado 11, 2 nd floor 28108 Madrid (Alcobendas), Spain #e richard@isoco.com #t +34-91-334-97-97, #f +34-91-334-97-99
Universidad Politécnica de Madrid	UPM 	Dr. Asunción Gómez-Pérez Campus de Montegancedo, sn Boadilla del Monte, 28660, Spain #e asun@fi.upm.es #t +34-91 336-7439, #f +34-91 352-4819
Institut für Informatik, Leopold-Franzens Universität Innsbruck	IFI 	Dr. Ying Ding Institute of computer science University of Innsbruck Technikerstr. 25 A-6020 Innsbruck, Austria #e dieter.fensel@uibk.ac.at #t +43 512 507 6486
Universität des Saarlandes	UdS 	Dr. Hans Uszkoreit Universitaet des Saarlandes Computerlinguistik D-66041 Saarbruecken, Germany #e uszkoreit@coli.uni-sb.de #t + 49 681 302-4115, #f: + 49 681 302-4700
The University of Liverpool	UniLiv 	Dr. Valentina A.M. Tamma Department of Computer Science, University of Liverpool Room 1.11, Chadwick Building Peach Street Liverpool L69 7ZF, UK #e valli@csc.liv.ac.uk #t +44 151 794 6797, #f +44 151 794 3715
Fundación Residencia de Estudiantes	Residencia  Residencia de Estudiantes	Mr Carlos Wert Fundación Residencia de Estudiantes Pinar, 23 28006 Madrid, Spain #e cwert@fundacionginer.org #t +34-91-446 01 97, #f +34-91-4468068
Centré d'Innovació i Desenvolupament Empreserial	CIDEM (Centré d'Innovació i Desenvolupament Empreserial)	Carlos Gómara Centré d'Innovació i Desenvolupament Empreserial Provença, 339 08037 Barcelona, Spain

Esperanto Services

	 Generalitat de Catalunya CIDEM 	#e cgomara@cidem.gencat.es #t +34-93-4767305, #f +34-93-4767303
Biovista	Biovista 	Dr. Andreas Persidis 34 Rodopoleos Street Ellinikon Athens 16777, HELLAS #e biovista@ath.forthnet.gr #t +30.1.9629848, #f +30.1.9647606

Table of Contents

1. INTRODUCTION	7
2. SEMANTIC WEB LANGAUES	9
2.1 EARLY AGE: SHOE AND ONTOBROKER.....	9
2.1.1 <i>SHOE</i>	10
2.1.2 <i>Ontobroker</i>	10
2.1.3 <i>Summary</i>	11
2.2 XML AND ITS FAMILY	12
2.2.1 <i>XML</i>	12
2.2.2 <i>XML Schema (XMLs)</i>	12
2.2.3 <i>XML Family</i>	13
2.3 RDF AND ITS FAMILY	14
2.3.1 <i>RDF</i>	14
2.3.2 <i>RDF Schema (RDFS)</i>	15
2.4 OIL, DAM+OIL AND OWL.....	17
2.4.1 <i>OIL</i>	17
2.4.2 <i>DAML+OIL</i>	20
2.4.3 <i>OWL</i>	21
2.5 <i>RULEML</i>	26
2.6 TOPIC MAP AND ITS FAMILY	26
2.7 <i>DAML-S</i>	27
2.8 SUPPORTING TOOLS	28
3. COMPARISON	29
3.1 GENERAL COMPARISON OF MODELING PRIMITIVES	29
<i>Factual knowledge: Data Models</i>	29
<i>Terminological knowledge: ontologies</i>	29
<i>Inference Knowledge</i>	30
3.2 SPECIFIC COMPARISON BETWEEN TWO LANGUAGES.....	31
<i>XMLs vs. DTD</i>	31
<i>RDF vs. XML</i>	31
<i>OIL vs. XML</i>	32
<i>OIL vs. XMLs</i>	32
<i>RDFS vs. XMLs</i>	33
<i>OIL vs. RDF(s)</i>	33
<i>DAML+OIL vs. RDFS</i>	34
<i>DAML+OIL vs. OIL</i>	34
<i>DAML+OIL vs. OWL</i>	35
4. ANALYSIS.....	36
5. CONCLUSION AND FUTURE PLAN	44
REFERENCES	47
APPENDIX 1.....	49

1. INTRODUCTION

The current World Wide Web (WWW) is, by its function, the syntactic web where structure of the content has been presented while the content itself is difficult to access to computers. Although the WWW has resulted in a revolution in information exchange among computer applications, it still cannot fulfill the interoperability among various applications without some pre-existing, human-created agreements somewhere in-house or outside of the web.

The next generation of the Web aims to alleviate such problem. The Web resources will be much easier and more readily accessible by both human and computers with the added semantic information in a machine-understandable and machine-processible fashion (Berners-Lee, 1999). "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation," wrote by Tim Berners-Lee, James Hendler and Ora Lassila in their Scientific American article "The Semantic Web".

How to make the Semantic Web possible in a way that computer could understand the semantic meaning of the information presented on the Web. Ontology here is the magic-maker. It plays a pivotal role by providing a source of shared and precisely defined terms that can be understood and processed by machines. A typical ontology consists of a hierarchical description of important concepts and their relations in a domain, task or service. The degree of formality employed in capturing these descriptions can be quite variable, ranging from natural language to logical formalisms, but increased formality and regularity clearly facilitates machine understanding. Therefore a decent ontology language which can help in the formality on the web is the most wanted thing in the Semantic Web. Refer to D1.1 for further information.

Various wish lists for the requirement of such Web ontology languages have been flung around the Web. To name but not limited, such as: it should be well designed for the intuition of human users without losing the adequate expressive power; it should be well defined with clear specified syntax and formal semantics; it should be compatible with existing web standards, etc. The concrete requirement report on Web ontology language design is well-documented by Heflin, Volz and Dale (2002).

In this survey, we intent to have the broader coverage for various existing web ontology languages, starting from SHOE and Ontobroker, and following the historical line, XML(s), RDF(s), OIL, DAML+OIL, and OWL. We also bring some attentions for DAML-S and TopicMaps. The structure of this survey is planned as follows: In Section 2 – Semantic Web Languages, we give the introduction for each language family. In Section 3, the comparison has been conducted among and between these languages. Section 4 gives the summarized comparison result and analysis. Section 5 contains the final summary and future plan.

2. SEMANTIC WEB LANGAUES

As Tim Bernes-Lee described, the Semantic Web Languages are like the language cake (see Figure 1). The layered tower is the dreamed vehicle to bring the Semantic Web to its full potential. The recognition of the importance of ontologies for the Semantic Web has led to the revolution and extension of the current web markup languages, e.g., XML Schema, RDF (Resource Description Framework), and RDF Schema, furthermore, OIL, DAML+OIL, and OWL.

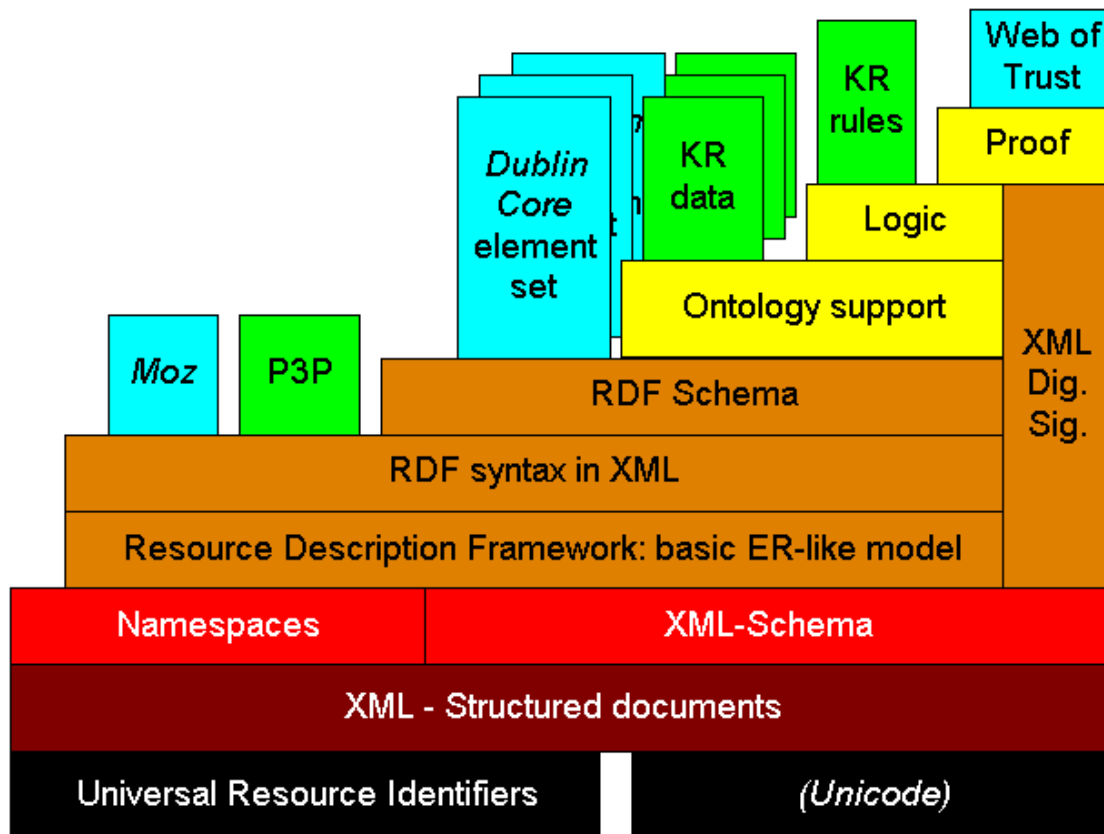


Figure 1: Semantic Web Language cake (layered tower)

The structure of this Section is planned as following the history of the web language development. First, we introduce the origin of the first-try (SHOE and Ontobroker), then main stream of the language development: XML(s), RDF(s), OIL, DAML+OIL, and OWL. At the end, We will bring your attentions with DAML-S, TopicMap and other related initiative world-wide.

2.1 Early age: SHOE and Ontobroker

In the early 90s, when the current Web just popped out and changed the whole world overnight like a tornado. When people are still indulged by the magic and realize that they need to spend much more time and effort to get used to the changes to their

normal daily life brought by the current Web, some prescient researchers already foresee the limits of the current Web. Therefore the early initiatives starts almost parallely together with the development of the current web. These initiatives are SHOE¹ and Ontobroker².

2.1.1 SHOE

SHOE has been developed in the University of Maryland (USA) in early 90s. It creates an extension of HTML by adding tags that are necessary to embed semantic data into web pages. This extension contains two categories: tags for constructing ontologies and tags for annotating web documents (Heflin and Hendler, 2001 and 2000; Heflin, Hendler and Luke, 1999; Luke, Spector, Rager and Hendler, 1997).

HTML <META>-tags is the first attempt at representing semantics inside Web-documents. Their intended use is limited to stating global properties that apply to the entire document. The anchors in META-tags are not standardized, which can be exploited in software if one wishes to, but cannot be interpreted by standard Web-browsers and search-engines.

SHOE proposes an extension of the HTML <META>-tag concept which can occur both in <HEAD> and <BODY> of a document. These SHOE expressions are separate from the contents of a document, and can be applied to the entire document, whereas HTML <META>-tags are limited to attribute-value pairs. SHOE expressions also include binary relations between instances. SHOE allows representing concepts, their taxonomies, n-ary relations, instances and deduction rules, which are used by its inference engine to obtain new knowledge.

Is important to notice that this language is no longer being maintained.

2.1.2 Ontobroker

Ontobroker has been developed in the University of Karlsruhe (Germany) during the mid 90s. It applies Artificial Intelligence techniques to improve access to heterogeneous, scattered and semi-structured information sources. It relies on the use of *ontologies* to annotate web pages, formulate queries, and derive answers. Ontobroker provides a broker architecture with three core elements: a query interface for formulating queries, an inference engine used to derive answers, and a webcrawler used to collect the required knowledge from the Web. The gist of Ontobroker is to create a methodology to define an ontology and use it to annotate/structure/wrap the web documents, and furthermore to make use of its advanced query and inference services (Fensel et al., 1999).

¹ <http://www.cs.umd.edu/projects/plus/SHOE/>

² <http://www.aifb.uni-karlsruhe.de/www-broker>

The Languages

Ontobroker defines three interleaved languages: an *annotation* language to enrich web documents with ontological information, a *representation* language to formulate ontologies, a *query language* (which is the subset of the representation language) to formulate queries.

The *annotation* language provided by Ontobroker is called HTML_A, which enables the annotation of HTML documents with machine-processable semantics. It extends the anchor tag with additional attribute, called the *onto* attribute, to annotate the web document based on three primitives – object, value, relationship. The same piece of data that is rendered by a browser is now having a semantic meaning defined by HTML_A.

A *representation* language is used to formulate an ontology, which defines the terminology (i.e., signature) and rules (i.e., axioms) that allow the derivation of additional facts. This language is based on *Frame logic* (Kifer et al., 1995) and introduces the terminology that is used by the annotation language to define the factual knowledge provided the Web, such as, class definition, attribute definition, is-a relationship, is-element-of relationship and rules.

The *query* language is defined as a subset of the representation language. The elementary expression is written in Frame logic: $x[\textit{attribute} \rightarrow v] : c$. Complex expressions can be built by combing these elementary expressions with the usual logical connectives.

The Tools

Ontobroker is implemented into two tools: a *webcrawler* and an *inference engine*. The *webcrawler* collects web pages from the Web, extracts their annotations, and parses them into the internal format of Ontobroker. The *inference engine* takes these facts together with the terminology and axioms of the ontology, and derives the answers to user queries. A hyperbolic presentation of the ontology and a tabular interface improve the accessibility of Ontobroker. Ontobroker was presented as a means to improve access to information provided in intranets and in the Internet (Fensel et al., 1999).

2.1.3 Summary

The ontology language provided by SHOE and Ontobroker are based on the current HTML with proper extension. Therefore the normal web browser can still interpret

the document by understanding additional embedded semantics. Especially the Ontobroker can be treated as the earliest Semantic Web prototype.

There are two main differences between SHOE and Ontobroker. First, the annotation language is not used to annotate existing information on the Web, but to add additional information and annotate them. That is, in SHOE, information must be repeated and this redundancy may cause significant maintenance problems. Ontobroker uses the annotations to directly add semantics to textual information that is also rendered by a browser. A second difference is the use of inference techniques and axioms to infer additional knowledge. SHOE can rely on frame-based systems in order to deal with inheritance. Ontobroker uses an inference engine to answer queries. Therefore, it can make use of rules that provide additional information.

2.2 XML and its family

With the exponentially increased web information, HTML is considered as too simple to present the document. The limited tags failed to provide some essential information on the Web, which limits HTML to only represent the layout of the information. Therefore the new language has to be designed to cater the new requirements emerging from web applications. XML³ is a tag-based language for describing tree structures with a linear syntax. It offers the facilities for users to define their own tags, which are needed for describing the structure of the documents. In this part, we discuss XML and the important members in its family.

2.2.1.XML

XML provides seven different means for presenting information: document, element, attribute, text, namespace, processing instruction, and comment. A DTD consists of three elements: *Element* declaration that define composed tags and value ranges for elementary tags, *attribute* declaration that define attributes of tags, and finally *entity* declaration. For more details, please visit <http://www.w3.org/XML>.

2.2.2 XML Schema (XMLs)

XML schemas are means for defining constraints on valid XML documents. They have the same purpose as DTDs but provide several significant improvements:

- definitions are itself XML documents. The clear advantage is that all tools developed for XML (e.g., validation or rendering tools) can be immediately applied to XML schema definitions.
- a rich set of datatypes that can be used to define the values of elementary tags.
- much richer means for defining nested tags (i.e., tags with sub-tags).

³ <http://www.w3.org/XML>

- the namespace mechanism to combine XML documents with heterogeneous vocabulary.

2.2.3 XML Family

As XML has been quickly adopted by academy and industry due to the easy use and implementation, various XML-like languages are designed for specific purposes. Here we list some important XML family members.

XSL

In XML, users can define their own tags therefore additional style sheet information is required for a browser to render such XML documents. XSL is mainly designed for this purpose to express the format information for XML documents. Furthermore, XSL allows defining views that can manipulate structure and elements of a document before they are rendered. XSL even enables the translation of one XML document into another one by using a different DTD. This is important in cases, where different user may wish to have different *views* on the information captured in a XML document.

XML is a standard language for defining tagged languages. However, XML does not provide standard DTDs, i.e., each user can/may/must define his own DTD. For exchanging data between different users relying on different DTDs one has to map different DTDs onto each other. XSL can be used to translate XML documents using DTD₁ in an XML document using DTD₂.

Query Languages for XML

The need for a query language for XML becomes obvious when comparing the WWW to a database. Query languages for XML provide query answering service which also applicable to semi-structured data. Currently, there exist still a number of proposals (XQL, XQuery, XPath). Most of them can be found in (<http://www.w3.org/XML/Query>).

Resources

<http://www.w3.org/XML/>
<http://www.w3.org/XML/Core/>
<http://www.w3.org/XML/Linking>
<http://www.w3.org/XML/Query>

<http://www.w3.org/XML/Schema>

<http://www.w3.org/Style/XSL/>

Summary

The first push towards more semantic structure on the Web has been the development of the XML. It allows the Web-page creators to define their own tags. In essence, XML allows structuring Web-pages as labeled trees, where the labels can be chosen to reflect as much of the document semantics as required. In general, XML is used for two purposes: for the markup of individual pieces of data, and as serialization syntax for other languages (eg SMIL, or RDF). In the first case, XML itself is used as a language to model meta-data. In the second use, XML is used as a language definition vehicle to define another language, which is in turn used to model meta-data.

XML provides semantic information as a by-product of defining the structure of the document. XML prescribes a tree structure for documents and the different leaves of the tree have a well-defined tag. The structure and semantics of document are interwoven. However, important aspects are lacking of rules and constraints (i.e., class definitions). They are often a significant part of the knowledge provided by an ontology.

2.3 RDF and its family

Although XML provides much more space for users to define their own tags, it fails to define the semantics in the machine understandable and processable way. RDF comes to fill up the hole. In this section, we will discuss RDF and its important family members.

2.3.1 RDF

The advantage that Semantic Web brings is that the computer can understand and process the semantics of the information on the current Web. The Resource Description Framework (RDF)⁴ is an important step toward that direction. It provides means for adding semantics to a document without making any assumption about the structure of the document. RDF is an infrastructure that enables the encoding, exchange and reuse of structured metadata. Search engines, intelligent agents, information broker, browsers and human user can make use of semantic information. RDF is an XML application (i.e., its syntax is defined in XML) customized for adding meta information to Web documents. Basically, RDF defines a data model for describing machine processable semantics of data, which consists of three object types:

⁴ <http://www.w3.org/RDF>

- **Resources:** A resource may be an entire Web page; a part of a Web page; a whole collection of pages; or an object that is not directly accessible via the Web; e.g. a printed book. Resources are always named by URIs.
- **Properties:** A property is a specific aspect, characteristic, attribute, or relation used to describe a resource.
- **Statements:** A specific resource together with a named property plus the value of that property for that resource is a RDF statement.

These three individual parts of a statement are called, respectively, the *subject*, the *predicate*, and the *object*. In a nutshell, RDF defines *object-property-value-triples* as basic modeling primitives and introduces a standard syntax for them. As RDF statements are also resources, statements can be recursively applied to statements allowing their nesting.

RDF emphasizes the facilities to enable automated processing of Web resources. The broad goal of RDF is to define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain. The definition of the mechanism should be domain neutral, and the mechanism should be suitable for describing information about any domain.

The foundation of RDF is a model for representing named properties and property values. The RDF data model⁵ is a syntax-neutral way of representing RDF expressions. RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent relationships between resources. As such, the RDF data model can therefore resemble an entity-relationship diagram. However, it doesn't provide mechanisms for declaring these properties, and the relationships between these properties and other resources. That is the role of RDF Schema.

RDF has been and will be applied in various areas. It provides better capabilities for search engine in resource discovery. It describes the content and content relationships in formalized way to provide computer-understandable catalogues. It facilitates the knowledge sharing and exchange among various intelligent software agents.

2.3.2 RDF Schema (RDFS)

In the Semantic Web, people are capable to describe the attributes of certain kinds of resources, for instance, to describe the "author", "title", and "subject" for certain bibliographic resources. The declaration of these properties (attributes) and their corresponding semantics are defined in the context of RDF as a RDF schema⁶. A schema defines not only the properties of the resource (e.g., title, author, subject, size,

⁵ <http://www.w3.org/TR/REC-rdf-syntax>; <http://www.w3.org/TR/rdf-mt/>

⁶ <http://www.w3.org/TR/rdf-schema/>

color, etc.) but also the kinds of resources being described (books, Web pages, people, companies, etc.).

The RDFS specifies the mechanisms needed to define the classes of resources, to restrict possible combinations of classes and relationships, and to detect violations of those restrictions.

RDFS is based on non-standard model theory which different layers of modeling primitives are glued together at one layer (Nejdl et al., 2000), which makes RDFS hard to understand. RDFS in particular can be recognized as an ontology language: it talks about classes and properties, range and domain constraints, and subclass and subproperty relations. However, it is a very limited language and more expressive power is clearly demanded to describe data in sufficient detail. Furthermore, such descriptions should be able to support the automated reasoning.

Resources

<http://www.w3.org/RDF/>

<http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>

<http://www.w3.org/TR/rdf-schema/>

<http://www.w3.org/TR/rdf-mt/>

<http://www.w3.org/TR/rdf-syntax-grammar/>

<http://www.w3.org/TR/rdf-testcases/>

<http://www.w3.org/TR/rdf-primer/>

<http://www.w3.org/TR/rdf-concepts/>

2.3.3 Summary

RDF and RDFS provide the way to represent the semantics of information on the Web. They allow the representation of concepts, taxonomies of concepts and binary relations. It is also intended to provide mechanisms to explicitly represent services, processes and business models, allowing non-explicit information to be recognized. They are the first web language capable to define the ontology in a limited way due to the lack of the necessary rich concept forming operators. However, richer languages can be built on top of them (such as OIL, DAML+OIL, and OWL).

Indeed this layered approach does provide benefits. Applications that are aware or understand RDF and RDFS can still process OIL, DAML+OIL or OWL ontologies. Although the application may not be able to extract and use all the information within the ontology, basic information about class hierarchies can be still available to such an application. RDF and RDFS only provided limited reasoning mechanism, which is mainly for constraint checking. While much powerful reasoning service is clearly demanded by the Semantic Web.

However, many design decisions in RDF are rather particular. Properties are defined globally, violating all modeling experiences from object-oriented and frame-based

approaches that define attributes locally for classes. RDF provides powerful reification, i.e. expressions can become terms in meta-level expressions. This makes reasoning service for RDF rather tricky. Most description do not provide such a service. Finally, RDF also does not provide a means for defining constraints and rules, which allow us to specify information intentionally.

2.4 OIL, DAML+OIL and OWL

Due to the pros and cons of RDF and RDFS, efforts have been allocated for developing a more expressive ontology language for the Semantic Web. OIL is one of the trials and has successfully generated the critical mass. In this section, we discuss OIL and its important family member, especially OWL.

2.4.1 OIL

OIL has been developed by the European IST project - Ontoknowledge⁷. As shown in figure 2, OIL is built on top of RDF and RDFS, using as much as possible their constructs in order to maintain backward compatibility. OIL provides modeling primitives used in frame-based and Description Logic oriented ontologies, coming along with a simple and clean semantics. It has a syntax definition using web standards such as RDF(s) and XML(s).

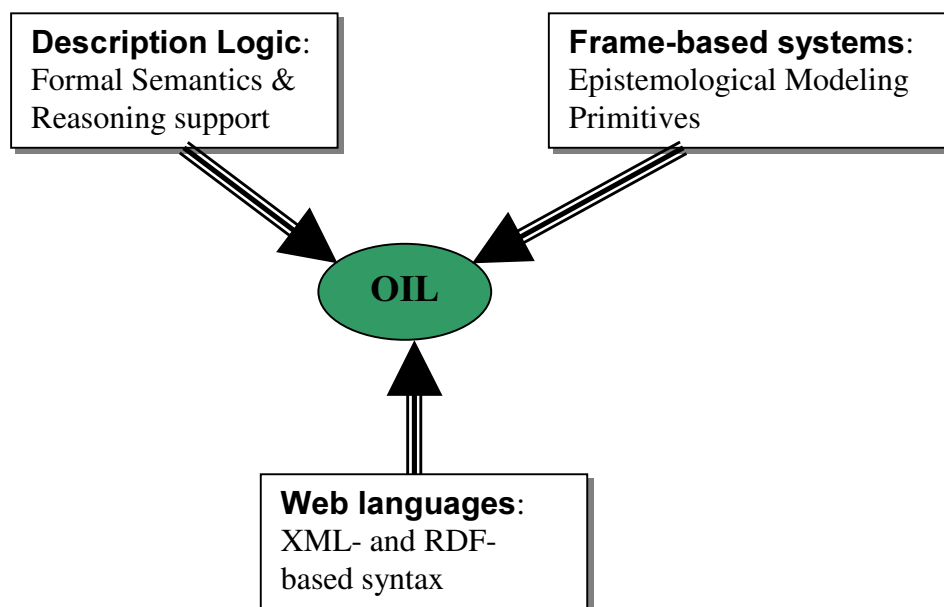


Figure 2. Three pillars of OIL

The three pillars of OIL

⁷ www.ontoknowledge.org/oil

OIL unifies three important aspects provided by different communities (see Figure 2): (1) formal semantics and efficient reasoning support as provided by Description Logic, (2) epistemologically rich modeling primitives as provided by the Frame-based community, and (3) a standard proposal for syntactical exchange notations as provided by the Web community. The details are:

- **Description Logics (DL).** OIL inherits from Description Logic its *formal semantics* and the *efficient reasoning support* (FaCT⁸ reasoner). OIL is capable to describe knowledge in terms of concepts and role restrictions that are used to automatically derive classification taxonomies. The reasoning support contains class and property subsumption, instance classification, query subsumption and query answering over classes and instances, and navigation through ontologies, etc. The FaCT reasoner is used to perform automatic classifications of concepts and constraint checking in taxonomies of concepts.
- **Frame-based systems.** OIL incorporates the essential-modeling primitives of frame-based systems on the notion of a concept, its superclasses and slots. OIL also treats slots as first class objects that can have their own properties (e.g., domain and range) and can be arranged in a hierarchy. With frame-based modeling we make the implicit assumption that only those attributes can be applied to a class that are defined for this class.
- **Web standards: XML and RDF.** First, OIL has a well-defined syntax in XML and XML schema definition. Second, OIL is defined as an extension of the RDF and RDF Schema. OIL extends this approach to a full-blown modeling language.

When describing ontologies in OIL we have to distinguish three different layers:

- The *object level*, where concrete instances of an ontology are described (Klein et al., 2000).
- The *first meta level* (called ontology definition in OIL) where the actual ontological definitions are provided. Here we define the terminology that may be instantiated at the object level. OIL is a means for describing structured vocabulary with well-defined semantics. The main contribution of OIL is in regard to this level.
- The *second meta level* (i.e., the meta-meta level, called ontology container in OIL) is concerned with describing features of such an ontology, like author, name, subject, etc. For representing metadata of ontologies OIL makes use of the Dublin Core Metadata standard.

Reasoning support is another feature of OIL. Reasoning can be used as the design support tool for large ontologies with multiple authors and as the back support for integrating and sharing ontologies. It can provide automatic consistency checking when you edit or generate ontologies especially with many authors synchronously. It can cross-link the inter-ontology relationships and check for the implied relationships.

⁸ <http://www.cs.man.ac.uk/~horrocks/FaCT/>

Layered cake of OIL

It is unlikely that a single ontology language can fulfill all the needs of the large range of users and applications of the Semantic Web. Therefore good ontology language should be layered to provide different levels of services. The core language should only contain consensus primitives. Additional expressive power can be easily added to different layers by language extensions. After two-year development and refinement of OIL, OIL presents a layered approach to a standard ontology language. Each additional layer adds functionality and complexity to the previous layer. This is done in such a way that agents (humans or machines) who can only process a lower layer can still partially understand ontologies that are expressed in any of the higher layers. Figure 3 sketches the relation between the OIL dialects and RDFS:

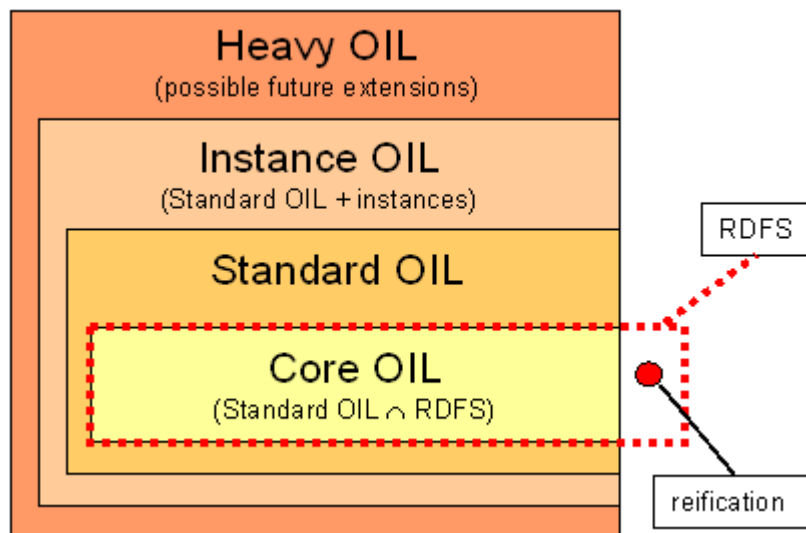


Figure 3. The layered structure of OIL

- **Core OIL** coincides largely with RDF Schema (with the exception of the reification features of RDF Schema). This means that even simple RDF Schema agents are able to process the OIL ontologies, and pick up as much of their meaning as possible with their limited capabilities.
- **Standard OIL** is a language intended to capture the necessary main stream modeling primitives that both provide adequate expressive power and are well understood by allowing the semantics to be precisely specified and complete inference.
- **Instance OIL** includes a thorough individual integration. While the previous layer - Standard OIL - included modeling constructs that allow individual fillers to be specified in term definitions, Instance OIL includes a full-fledged database capability.
- **Heavy OIL** may include additional representational (and reasoning) capabilities.

The layered architecture of OIL has three main advantages: First, an application is not forced to work with a language that offers significant more expressiveness and

complexity than it actually needs. Second, applications that can only process a lower level of complexity are still able to catch some of the aspects of an ontology. Third, an application that is aware of a higher level of complexity can still also understand ontologies expressed in a simpler ontology language.

2.4.2 DAML+OIL

DAML+OIL⁹ is a semantic markup language for Web resources that has been created as a joint effort of the American and European ontology communities for the Semantic Web. The *Joint EU/US ad hoc Agent Markup Language Committee*¹⁰ was installed to further develop DAML+OIL. DAML+OIL is the result of merging DAML-ONT (an early result of the DARPA Agent Markup Language (DAML¹¹) programme) and OIL (the Ontology Inference Layer). This language has a clean and well-defined semantics, and many efforts are being put to provide reasoning mechanisms for DAML+OIL.

DAML+OIL is an ontology language specifically designed for the Semantic Web. It exploits existing Web standards (XML and RDF), adding the ontological primitives of object oriented and frame-based systems, and the formal rigor of expressive description logic. As an ontology language, DAML+OIL is designed to describe the *structure* of a domain. DAML+OIL takes an object-oriented approach, with the structure of the domain being described in terms of *classes* and *properties*, and the set of *axioms* that assert characteristics of these classes and properties.

Datatypes

DAML+OIL supports the full range of datatypes in XML Schema: the so called primitive datatypes such as string, decimal or float, as well as more complex derived datatypes such as integer sub-ranges. This is facilitated by maintaining a clean separation between instances of “object” classes (defined using the ontology language) and instances of datatypes (defined using the XML Schema type system). From a theoretical point of view, this design means that the ontology language can specify constraints on data values, but as data values can never be instances of object classes. This allows the type system to be extended without having any impact on the ontology language, and vice versa. Similarly, the formal properties of hybrid reasoners are determined by those of the two components; in particular, the combined reasoner will be sound and complete if both components are sound and complete.

Axiom

⁹ <http://www.daml.org/language/>; <http://www.daml.org/2001/03/daml+oil-index.html>

¹⁰ <http://www.daml.org/committee/>

¹¹ <http://www.daml.org>

The axioms supported by DAML+OIL includes: `subClassOf`, `sameClassAs`, `subPropertyOf`, `samePropertyAs`, `disjointWith`, `sameIndividualAs`, `differentIndividualFrom`, `inverseOf`, `transitiveProperty`, `uniqueProperty`, `unambiguousProperty`. A crucial feature of DAML+OIL is that `subClassOf` and `sameClassAs` axioms can be applied to arbitrary class expressions. This provides greatly increased expressive power with respect to standard frame-based languages. A consequence of the expressive power of DAML+OIL is that all of the class and individual axioms, as well as the `uniqueProperty` and `unambiguousProperty` axioms, can be reduced to `subClassOf` and `sameClassAs` axioms. DAML+OIL also allows properties of properties to be asserted. It is possible to assert that a property is unique (i.e., functional) and unambiguous (i.e., its inverse is functional). It is also possible to use inverse properties and to assert that a property is transitive.

The meaning of DAML+OIL is defined by a standard model-theoretic semantics¹². The semantics is based on interpretations, where an interpretation consists of a domain of discourse and an interpretation function.

Inference in DAML+OIL

DAML+OIL is equivalent to a very expressive Description Logic, i.e. SHOIQ, with the addition of existentially defined classes (i.e., the one of constructor) and *datatypes*. This equivalence allows DAML+OIL to exploit the considerable existing body of description logic research to define the semantics of the language and to understand its formal properties.

Reasoning can be useful at many stages during the design, maintenance and deployment of ontologies. Reasoning can be used to support ontology design and to improve the quality of the resulting ontology. For example, class consistency and subsumption reasoning can be used to check for logically inconsistent classes and (possibly unexpected) implicit subsumption relationships. Ontology integration can also be supported by reasoning (Calvanese *et al.* 1998).

2.4.3 OWL

OWL is the web ontology language currently under the development of W3C Web Ontology (WebOnt) Working Group. OWL can help power automated tools for the next generation web, offering advanced services such as more accurate Web search, intelligent software agents and knowledge management. OWL is mainly based on OIL and DAML+OIL and therefore the main features of OWL are very similar to those of OIL. Heflin, Volz and Dale (2002) proposed the wish lists for OWL based on various case studies. OWL includes three sub languages called OWL-Lite, which roughly consists of RDFS plus equality and 0/1-cardinality, OWL DL that contains the whole OWL vocabulary, and OWL Full, composed by the complete vocabulary interpreted more broadly than in OWL DL. OWL-Lite suits particularly well to

¹² <http://www.daml.org/2001/03/model-theoretic-semantics>

express the light weight ontologies. But medical science and technical engineering communities among others are complaining about the limited expressive power of OWL-Lite. What they want is much full-fledged semantic web modeling language to fulfill the requirement so as to define much complex and heavy weight ontologies they have in their own fields, the language with at least the expressive power of full OIL/OWL. Therefore the layered design algorithm is very essential for OWL as well. Currently OWL-Lite is functioned as sub language of full OWL.

OWL Abstract Syntax¹³

Besides the DAML+OIL style RDF syntax, the OWL specification also includes an abstract syntax, which provides a higher level and less cumbersome way of writing ontologies. It also has the advantage of allowing a more succinct statement of the semantics. The abstract syntax is defined using an extended BNF notation. A translation from this syntax to the RDF syntax is available. The OWL abstract syntax is very much like OIL in that it provides for compound axioms resembling frames. It is interesting to observe that the OWL abstract syntax has reverted to grouping axioms into frame structures. The basic idea of having a semantic web language to represent ontology is aiming to allowing computer programs to inter-operate each other without pre-existing, outside-of-the-web agreements. If this language also has an effective reasoning mechanism, then computer programs can manipulate this interoperability information themselves.

The OWL language can be used to allow the explicit representation of term vocabularies and the relationships between entities. This language goes beyond XML, RDF and RDF-S in allowing greater machine readable content on the web. The OWL language is a revision of the DAML+OIL web ontology language-incorporating lesson learned from the design and application use of DAML+OIL.

Main features of OWL language include:

- **Ontologies:** An OWL ontology is a sequence of axioms and facts, plus inclusion references to other ontologies, which are considered to be included in the ontology. OWL ontologies are web documents, and can be referenced by means of a URI. Ontologies also have a non-logical component (not yet specified) that can be used to record authorship, and other non-logical information to be associated with a ontology.
- **Axioms:** Axioms are used to associate class and property IDs with either partial or complete specifications of their characteristics, and to give other logical information about classes and properties. It contains Class Axioms, Property axioms, Descriptions and Restrictions.
- **Facts:** Facts state information about particular individuals in the form of a class that the individual belongs to plus properties and values. Individuals can either be given an individualID or be anonymous (blank nodes in RDF terms). The syntax here is set up to mirror the normal RDF/XML syntax.

¹³ **OWL Web Ontology Language Abstract Syntax** This document gives a high level description of the language features in an abstract syntactic form similar to that of OIL (<http://www.w3.org/TR/2002/WD-owl-absyn-20020729/>).

The OWL datatypes is on hold pending the outcome of RDFS. It is assumed that the semantics of OWL will be similar to those of DAML+OIL. The standard semantics of OWL are based on an *open world assumption* (OWA). This means that we cannot assume that all information is known about all the individuals in the domain. Thus, simply being unable to prove that an individual *a* is an instance of *X* does not justify our concluding that *a* is not an instance of *X*. This is in contrast to languages or systems using *negation as failure* or a *closed world assumption* (CWA). The OWA facilitates reasoning about intentional definitions of classes—we do not need to know all the information about the world in order to be able to make deductions about the relationships between classes.

OWL Lite¹⁴

The objective of OWL Lite is to provide an easy-going language for tool builders. Tools play very important role for securing the widespread adoption of OWL. But this has to be balanced by without losing the necessary expressive power of DAML+OIL or OIL. In order to provide a semantic web language that fits to the various needs of a wider audience, the language itself must have the layered structure. Therefore OWL Lite has been developed to capture many of the commonly used features of OWL and DAML+OIL. It attempts to provide more functionality than RDFS, which is important in order to support web applications.

OWL Lite is a subset of the full OWL language constructors and has a few limitations therefore. Unlike the full OWL language (and DAML+OIL), classes can only be defined in terms of named superclasses. Equivalence for classes, and subclass between classes are only allowed on named classes. Property restrictions in OWL-Lite must use named classes as well. The only cardinalities allowed to be explicitly stated in OWL Lite are 0 or 1. See Appendix 1 for more details.

OWL DL

OWL DL includes the complete OWL vocabulary, interpreted under a number of simple constraints. Primary among these is type separation. Class identifiers cannot simultaneously be properties or individuals. Similarly, properties cannot be individuals. OWL DL is so named due to its correspondence with description logics.

OWL Full

¹⁴ **OWL Web Ontology Language Reference** This document gives a detailed specification of the RDF syntax of the language, including usage examples (<http://www.w3.org/TR/2002/WD-owl-ref-20020729/>).

OWL Full includes the complete OWL vocabulary, interpreted more broadly than in OWL DL, with the freedom provided by RDF. In OWL Full a class can be treated simultaneously as a collection of individuals (the class extension) and as an individual in its own right (the class intension). Another significant difference from OWL DL is that a DatatypeProperty can be marked as an InverseFunctionalProperty. These are differences that will be of interest to the advanced user. This document does not describe the use of these features.

OWL DL and OWL-Lite

OWL extends the constructions of OWL Lite with the following:

- **oneOf** (enumerated classes): Classes can be described by enumeration of the individuals that make up the class, e.g., the class of daysOfTheWeek.
- **hasValue** (property values): A property can be required to have a certain individual as a value (also sometimes referred to as property fillers), e.g., the class of austriaCitizens can be characterized as those people that have Austria as a value of their nationality. (Where Austria itself is an instance of the class of all nationalities).
- **disjointWith**: The full OWL language allows the statement that classes are disjoint, for example stating that man and woman are disjoint classes.
- **sameClassAs** and **rdfs:subClassOf**: those are added for class expressions.
- **unionOf**, **complementOf**, and **intersectionOf** (Boolean combinations): OWL allows arbitrary Boolean combinations of classes.
- **minCardinality**, **maxCardinality**, **cardinality** (full cardinality): While in OWL Lite, cardinalities are restricted to either 1 or 0. Full OWL allows cardinality statements for arbitrary non-negative integers.
- **complex classes** : OWL Lite restricts the syntax to single class name. Full OWL extends this to allowing arbitrarily complex class descriptions, consisting of enumerated classes, property restrictions, and Boolean combinations of these. OWL also includes a special "bottom" class with the name **Nothing** that is the empty class.

DAML+OIL vs. OWL (OWL-Lite)

Although OWL and its family are still under development, many new features or changes might happen pretty soon (we will keep on documenting the changes and status of the development for various version of Del 22 during the whole project period). At the moment, OWL differs from DAML+OIL in the following respects:

- The namespace has been changed to <http://www.w3.org/2002/07/owl>.
- Several bugs and omissions in RDF and RDF Schema have now been fixed by the RDF Core Working Group. This allows OWL to use "official" RDF syntax (in contrast to DAML+OIL). Examples include:

- 1) RDF now supports cyclical class and property inclusions (using `rdfs:subClassOf` and `rdfs:subPropertyOf`), so the relevant RDF properties can now be used in OWL.
 - 2) RDF now supports multiple `rdfs:domain` and `rdfs:range` properties, with both being treated as equivalent to the intersection of the individual constraints.
 - 3) RDF now includes a collection parse type (`rdf:parseType="Collection"`) which replaces the old `rdf:parseType="daml:collection"`.
 - 4) RDF now has its own formal semantics (a model theory). But the relationship between this and the OWL semantics is not yet clear.
 - 5) The RDF core working group is examining the issue of datatypes and may produce a recommendation. The issue of OWL datatypes is on hold pending the outcome of their work.
- Some DAML+OIL properties and classes have been renamed. Table 1 shows the relevant DAML+OIL names and their corresponding OWL names:

Table 1. The renamed class and property between DAML+OIL and OWL

DAML+OIL	OWL
<code>daml:hasClass</code>	<code>owl:someValuesFrom</code>
<code>daml:toClass</code>	<code>owl:allValuesFrom</code>
<code>daml:UnambiguousProperty</code>	<code>owl:InverseFunctionalProperty</code>
<code>daml:UniqueProperty</code>	<code>owl:FunctionalProperty</code>

- A new type of property, `owl:SymmetricProperty`, has been added, which can directly state that properties are symmetric.
- Qualified cardinality restrictions are not supported in OWL (. This results in the removal of the DAML+OIL properties `daml:cardinalityQ`, `daml:hasClassQ`, `daml:maxCardinalityQ` and `daml:minCardinalityQ`. This would seem to represent a real reduction in the expressive power of OWL with respect to DAML+OIL.

Resources

<http://www.ontoknowledge.org/oil/>
<http://www.daml.org/2001/03/daml+oil-index.html>
<http://www.w3.org/TR/daml+oil-reference>
<http://www.w3.org/2001/sw/WebOnt/>
<http://www.w3.org/TR/2002/WD-owl-features-20020729/>
<http://www.w3.org/TR/2002/WD-owl-guide-20021104/>
<http://www.w3.org/TR/2002/WD-owl-ref-20021112/>
<http://www.w3.org/TR/2002/WD-owl-semantic-20021108/>
<http://www.w3.org/TR/2002/WD-owl-test-20021024/>
<http://www.mindswap.org/2002/owl.html>
<http://www.daml.org/language/>

Summary

Since OIL has been developed in Europe by the end of 2000, due to the existing bugs and strange modeling theory which RDF and RDFS are based on. The evolution has been spread out the whole Web area. The joint collaboration between EU and USA has come out with DAML+OIL, which has been significantly used throughout various areas for adding semantics to the knowledge to assist computer to understand and process information more intelligently. Still the current semantic web ontology language are far from what the real life required and the Web is such a heterogeneous environment, the language for such environment must be layered. OWL is the layered language for this environment and currently is still under-going. In a nutshell, the current OWL-Lite roughly consists of RDFS + equality + 0/1-cardinality.

2.5 RuleML

Rules for the Web have become to establish their importance since inference rules are identified as a Design Issue of the Semantic Web. Rules are also continued to play an important role in Intelligent Agents and AI shells for knowledge-based systems, which need a Web interchange format. The Rule Markup Initiative has taken initial steps towards defining a shared Rule Markup Language - RuleML¹⁵, permitting both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, and further inferential-transformational tasks.

2.6 Topic Map and its family

Topic Maps provide a formalization of the notion of a back-of-book index. They are been standardized in ISO (ISO/IEC 13250¹⁶). Topic maps provide a mechanism for describing knowledge structures and associating them with resources. Topic Maps contains three notions:

- Topic: A topic represents any “thing” whatsoever – a person, entity, concept – regardless of whether it exists or has any specific characteristics. It allows the specification of classes of topics – thus topics represent both classes and instances.
- Association: Topic associations represent the relationships between topics.
- Occurrence: An occurrence of a topic is an information resource that is deemed to be relevant to the topic in some way.

Topic maps are strongly connected to notions of indexing in Library and Information Science. They might prove to be a useful mechanism as indexing structures within the Semantic Web. However, the lack of concept forming operators or constructs with

¹⁵ <http://www.dfki.uni-kl.de/ruleml>

¹⁶ <http://www.y12.doe.gov/sgml/sc34/document/0129.pdf>

detailed semantics suggests that they may not provide an appropriate formalism for the representation of rich ontologies.

TopicMaps.Org¹⁷ is an independent consortium for developing Topic Maps. Their work includes the development of version 1.0 of an XML grammar for interchanging Web-based Topic Maps, called XML Topic Maps - XTM¹⁸ Version 1.0.

The design idea of Topic Maps are mainly originated from Library and Information Science, where the indexing the structure of knowledge has been existed and applied for many years. In order to index the huge web document, some formalized language is needed. But Topic Maps failed to express much rich ontologies due to the limited functionality for class and property it could provide.

2.7 DAML-S

DAML-S¹⁹ is developed by the DAML (DARPA Agent Markup Language) program. It is a DAML+OIL based Web service ontology, which supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. It enables users and software agents to discover, select, invoke, compose and monitor Web resources based on the well-defined ontology. DAML-S has the following features:

- Automatic Web service discovery: Automatic location of services, which provide a particular or specific service.
- Automatic Web service invocation: The possibility of automatic execution of an identified Web service by a computer program or an agent.
- Automatic Web service composition and interoperation: The selection, composition and interoperation of different services automatically integrated to perform a task
- Automatic Web service monitoring: Trace and monitoring of the services during their execution.

DAML-S is an attempt to bring the benefits of the emergent ontology-based information standards of the Semantic Web to web services. That is to bring the notion of formal, explicit, semantics to the description and integration of web services. To that end, DAML-S employs DAML+OIL, description logic-based, web-enabled, ontological mark-up language to define an upper ontology for web services. DAML-S has a three-level model that corresponds to answering three groups of key questions of web services:

- What does the service require? This information lets an agent who is seeking for services understand what a service is going to need to do what it says it can do. This information is to be found in the *ServiceProfile*.
- How does it work? The information tells an agent how it is going to do what it says it can do. This information is to be found in the *ServiceModel*

¹⁷ www.topicmap.org

¹⁸ <http://www.doctypes.org/xtm/home.html>

¹⁹ <http://www.daml.org/services/>

- How is it used? The information informs an agent of how it should be addressed and communicated with. This information is to be found in the *ServiceGrounding*.

Because DAML-S itself is not a Semantic Web language, rather an application of the Semantic Web language, DAML+OIL or OWL, to define the ontology for the web services, it will not be detailed discussed in this deliverable. If it is needed during the late stage of the project, detailed discussion or comparison of DAML-S will be provided in the forthcoming version of the deliverable.

2.8 Supporting tools

There are some tools available to facilitate the wider-use of the above semantic web languages. Deliverable 13 in Ontoweb gives the detailed roadmap of various existing supporting tools, please visit <http://www.ontoweb.org/deliverable.htm>.

3. COMPARISON

Artificial Intelligence has a strong tradition in developing methods, tools and languages for structuring knowledge and information. While meta-data annotation of Web resources is the example for applying AI techniques on a large and successful scale. In essence, the machine-processable representations of the semantics of these resources are the backbones.

3.1 General comparison of modeling primitives

The modeling primitives here we would like to stress are factual knowledge – data models, terminological knowledge – ontologies, and inference knowledge.

Factual knowledge: Data Models

The data-models underlying the semantic web languages vary:

- Ontobroker's nonstandard onto-attribute is based on F-logic and therefore has the Flogic's rich data model, consisting of classes, attributes with domain and range definitions, is-a hierarchies with set inclusion of subclasses and multiple attribute inheritance.
- XML takes *labelled trees* as its basic data-model, therefore XML and XML Schema actually define the labelled tree data structure – data syntax. There are several ways to represent the same RDF data model in XML.
- RDF 's data model is a syntax-neutral way of representing RDF expressions. It is based on *binary relations*, enhanced with a reification mechanism to enable relations between relations, and statements about the statements. This data model consists of three object types (resources, properties and statements). RDFS uses this basic data model to build a basic object-oriented type schema on top of RDF.
- SHOE's data-model is similar to that of RDFS. It does not include the reification mechanism of RDF. SHOE allows the specification of classes with attributes, with multiple inheritance of attributes between classes.
- OIL, DAML+OIL's data model is based on description logic and Frame-based logic. Therefore, they have the rich class, property, and axiom to model the world.
- OWL is OIL and DAML+OIL based. Provides an abstract syntax as a higher level to write ontologies. Ontologies built as sequences of axioms, facts and reference to other ontologies.

Terminological knowledge: ontologies

Ontologies are shared specification of a conceptualization and the corresponding vocabulary used to describe a domain. Ontologies can define the rich semantics of much complex objects and therefore well-suited for describing heterogeneous, distributed and semi-structured information sources such as things on the Web:

- In SHOE, ontologies can be defined by information-providers themselves inside their own HTML pages. Such an ontology contains a class-lattice and possible relations between instances of these classes. SHOE allows for local definitions of ontologies (or local extensions of central ontologies).
- Ontobroker ontologies are similar in nature (a class-hierarchy, attributes with domain and range definitions, and multiple attribute inheritance). Ontobroker relies on a single centrally defined ontology.
- XML-schema provides the mechanisms for constraining document structure and content; to enable inheritance for element, attribute, and datatype definitions. It also allows creation of user-defined datatypes.
- RDFS can be used directly to describe an ontology with its *Objects*, *Classes*, and *Properties*. On the one hand, RDFS provides rather limited expressive power. A serious weakness of RDF is that it lacks a standard for describing logical axioms. No intentional definitions or complex relationships via axioms can be defined. On the other hand, RDFS provides a rather strong reification mechanism. RDFS can be used to describe what is called an ontology in SHOE and Ontobroker.
- OIL, DAML+OIL and OWL are the languages mainly aiming for defining ontologies. They provide much richer constructors for forming complex class expressions and the axioms can be easily defined by these languages. Another important feature of these language is that they are layered language with different expression power on different layer to fulfil different needs to define simple or complex ontologies.

Inference Knowledge

Here we analyze to which the extent the inferential knowledge can be expressed in the various semantic web languages. We take the class subsumption as a simple example:

- SHOE can describe such inferential knowledge in pure Horn rules inside local Web pages.
- Ontobroker states this inferential knowledge centrally. It allows a larger fragment of first-order logic to be used, which can be translated to stratified normal logic programs via the Lloyd-Topor transformations [Lloyd & Topor, 1984].
- XSLT²⁰ (XSL Transformation language) allows expressing transformations of XML structures, and these can be used to express certain inferential knowledge. RuleML²¹ permits both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, and further inferential-transformational tasks.
- RDF/RDFS' subclass relation can be used to represent the class subsumption.

²⁰ <http://www.w3.org/Style/XSL>

²¹ <http://www.dfki.uni-kl.de/ruleml/>

- OIL, DAML+OIL and OWL can define even much complex rules or axioms.

Table 2 gives the summary of the comparison on modeling primitives, where the range from - to ++ indicates a range of expressive power in the corresponding category.

Table 2: Summary of the modeling primitives

	facts	terminology	inference
SHOE	++	++	+
Ontobroker	++	++	++
XML	+	+/-	-
RDF(S)	+	+/-	-
OIL	++	++	++
DAML+OIL	++	++	++
OWL	++	++	++

3.2 Specific comparison between two languages

Semantic web languages are complementary technological means that will enable ontological support in knowledge management and electronic commerce. XML provides a standard serial syntax for exchanging data. A DTD allows for the definition of structure and elementary tags of an XML document. XSLT allows translating between different XML documents. RDF provides a standard for describing machine-processable semantics of data. Finally, OIL, DAML+OIL and OWL provide much more support to define the complex or heavy weight ontology including the inference.

XMLs vs. DTD

Main improvements of XML schemas compared to DTDs are: 1) XML schemas definitions are themselves XML documents. 2) XML schemas provide a rich set of datatypes that can be used to define the values of elementary tags. 3) XML schemas provide much richer means for defining nested tags (i.e., tags with subtags). 4) XML schemas provide the namespace mechanism to combine XML documents with heterogeneous vocabulary.

RDF vs. XML

RDF is an application of XML to representing meta data. For example, the RDF statement can be represented in linear XML syntax. However, RDF provides a *standard* way on how to represent meta data in XML. Using directly plain XML for representing meta data would result in different syntax. RDFS provides a fixed set of modeling primitives for defining an ontology (classes, resources, properties, is-a and element-of relationship, etc.) and a standard way on how to encode them in XML.

OIL vs. XML

XML can be used as a serial syntax for OIL, which is very useful because it puts OIL in the mainstream of tools that are currently being developed for supporting XML based applications. Validation and rendering techniques developed for XML can directly be used for ontologies specified in OIL. Central for an ontology is the is-a relationship, and XML schemas incorporate the notion of inheritance. [Klein et al., 2000] discussed a more complex translation procedure that leads XML documents to capture the semantics of an ontology in OIL by using type refinement in XML schemas to model the subsumption between concepts in OIL.

OIL vs. XMLs

XML schemas and OIL have one main goal in common, which is to provide vocabulary and structure for exchanging information sources.

- **XML schemas and OIL both have the XML syntax.** The XML syntax of OIL is useful for supporting the exchange of ontologies specified in OIL (see OIL vs. XML section).
- **XML schemas have rich datatypes while OIL not.** XML schemas provide a much richer set of basic datatypes, for example, string, boolean, float, decimal, timeInstant, binary, etc. OIL does not provide these built-in datatypes because reasoning with concrete domains quickly becomes undecidable or at least inefficient. XML schemas do not worry about this aspect because all inheritance needs to be defined explicitly.
- **XML schemas do not support intentional definition of types while OIL does.** OIL provides an explicit language for the intentional definition of types that is completely lacking in XML schemas.
- **XML provides structures: elements, while OIL not.** In XML schemas main modeling primitives are elements. Elements may be simple, composed or mixed, and elements can have attributes. While the modeling primitives of OIL are concepts and slots. Concepts can be roughly identical with elements and child elements are roughly equivalent to slots defined for a concept. However, slots defined globally have no equivalent in XML schemas. The reduction step has to be taken when transforming OIL specifications into XML schema definitions.

- **XML schema provides structures: grammar, while OIL not.** OIL does not provide any grammar for composing concepts from slots, i.e., an instance of a concept is a set of slot values. XML schemas allow defining stronger requirements via a grammar: sequence and choice of attributes applied to an instance can be defined.
- **XML schema provides structures: type-derivation, while OIL not.** XML schemas incorporate the notion of type-derivation. However, this can only partially be compared with what is provided with inheritance in OIL. All inheritance has to be modeled explicitly in XML schemas. While, inheritance can be derived from the definitions of the concepts in OIL.
- **XML schema and OIL provide namespaces.** XML schemas and OIL both provide the same (limited) means for composing specifications.

OIL relates to XML schemas like the Extended Entity Relationship model relates to the relational model. On the one hand, OIL provides much richer modeling primitives. It distinguishes classes and slots and class (or slot) definitions can be used to derive the hierarchy (and their corresponding inheritance). On the other hand, XML schemas provide richer modeling primitives concerning the variety of built-in datatypes and the grammar for structuring the content of elements. Models in OIL can be viewed as a high level description that become further refined when aiming for a document structure model.

RDFS vs. XMLs

RDF Schema enriches the limited modeling primitives defined in RDF by adding, for example, class, subclass relationship, domain and range restrictions for property, and subproperty. With these extensions, RDF Schema comes closer to the existing ontology languages, while XML Schema not. XML Schema and DTDs prescribe the order and combination of tags in an XML document, in another world, the syntax or structure of information. In contrast, RDF Schema only provides information about the interpretation of the statements given in a RDF data model, but it does not constrain the syntactical presentation of a RDF description.

OIL vs. RDF(s)

One of the syntax of OIL is based on RDFS. To ensure maximal compatibility with existing RDF/RDFS-applications and vocabularies, the integration of OIL with the resources defined in RDF-schema is very essential:

- The abstract OIL class `OntologyExpression` is a subclass of `rdfs:Resource` and the abstract OIL class `OntologyConstraint` is a subclass of `rdfs:ConstraintResource`.

- OIL-slots are realized as instances of `rdf:Property` or of subproperties of the original `rdf:Property`. The subplot relationship can be expressed by the `rdfs:subPropertyOf` relationship. `rdf:Property` is enriched by OIL with a number of properties that specify inverse and transitive roles and cardinality constraints, which are not possible in RDF/RDFS.

OIL uses the existing primitives of RDFS as much as possible to retain an unambiguous mapping between the original OIL specification and its RDFS serialization. Therefore, the RDFS contained in the definition of domain ontology in OIL can be easily understood or interpreted by any non-OIL-aware RDFS applications, while OIL-aware applications can be benefited by the added features of formal semantics and reasoning support. In a nutshell, any valid OIL document is also a valid RDF(S) document when all the elements from the OIL-namespace are ignored. In order to allow users to choose the expressive power appropriate to their application, and to allow for future extensions, a layered family of OIL languages has been described. The sub-language OIL Core has been defined to be exactly the part of OIL that coincides with RDF(S).

DAML+OIL vs. RDFS

DAML+OIL is tightly integrated with RDFS by using RDFS to express the syntax of DAML+OIL. Therefore, the existing RDFS infrastructure can be easily reused and ontologies defined by DAML+OIL can be partially or fully compatible with those defined by RDFS. On the other hand, DAML+OIL also inherits the “strange” modeling idea of RDFS, such as restrictions with multiple properties and classes. DAML+OIL’s dependence on RDFS also leads to the consequence of the decidability of the language. Decidability is lost when cardinality constraints can be applied to properties that are transitive, or that have transitive sub-properties. So decidability in DAML+OIL depends on an informal prohibition of cardinality constraints on non-simple properties.

DAML+OIL vs. OIL

From the point of view of language constructs, the differences between OIL and DAML+OIL are rather trivial. There usually exists a one to one mapping of constructors between these two languages. In the cases where the constructors are not completely equivalent, simple translations are possible.

OIL also uses RDFS for its serialisation (although it also provides a separate XML-based syntax). OIL’s RDFS based syntax might cause it to face the same problems as those for DAML+OIL. However, OIL processors would reject ontologies containing unexpected use of the meta-properties. Therefore, not any ontology conforming to the RDFS meta-description will be a valid OIL ontology. OIL and DAML+OIL take rather different positions with regard to the layering of languages on the Semantic Web.

The tight integration of DAML+OIL with RDFS leads to the loose of the frame structure of OIL syntax. A DAML+OIL ontology is more DL-like in that it consists largely of a relatively unstructured collection of subsumptions and equality axioms. This makes difficult to use DAML+OIL with frame-based tools such as Protégé (Grosso *et al.* 1999) or DL based ones as OilEd (Bechhofer *et al.* 2001).

The treatment of individuals in OIL is also very different from that in DAML+OIL. DAML+OIL treats individuals occurring in the ontology (in `oneOf` constructs or `hasValue` restrictions) as true individuals (i.e., interpreted as single elements in the domain of discourse) and not as primitive concepts as in OIL. This weak treatment of the `oneOf` construct is a well known technique for avoiding the reasoning problems that arise with existentially defined classes, and is also used, e.g., in the CLASSIC knowledge representation system (Borgida & Patel-Schneider 1994). Moreover, DAML+OIL makes no unique name assumption: it is possible to explicitly assert that two individuals are the same or different, or to leave their relationship unspecified. DAML+OIL relies wholly on RDF for assertions involving the type (class) of an individual or a relationship between a pair of objects.

DAML+OIL vs. OWL

The difference between DAML+OIL and OWL is rather trivial as well. But OWL abstract syntax has reverted to grouping axioms into frame like structure, which makes it easy to use the frame-based tools such as Protégé or DL based ones OilEd. In this sense, OWL is closer to OIL due to the frame-based feature while DAML+OIL is more DL-like.

Because OWL is still a very young semantic web ontology language (less than half-year old), more development will be continued in this area. Significant works are going on in the Web Ontology Working Group of W3C. We will keep the sharp eye on its evolution and report the changes in our later deliverables.

4. ANALYSIS

In this section, we analyze the existing semantic web ontology languages in more details based on the latest requirement raised in Web Ontology Working Group of W3C and some features we presume are important to report here.

Wish lists for Semantic Web Ontology Languages

Heflin, Volz and Dale (2002) proposed the wish lists for OWL based on various case studies, which can be summarized into eight design goals of a Web Ontology language:

- 1) The first is that it should provide for **ontology sharing**, including the ability of one public ontology to extend another public ontology by, perhaps, specializing some of its terms or properties.
- 2) The second design goal is to support the change or drift over time of ontologies and their constitutive parts (**ontology versioning**). As knowledge domains evolve, ontologies must be able to evolve in order to continue to formally represent their domain.
- 3) Third, it must support **interoperability**, by offering some way to map similarities between disparate ontologies.
- 4) The fourth design goal is to provide the possibility to detect the inconsistencies among different ontologies or data sources (**ontology reasoning** support).
- 5) Fifth, it should strike a balance between **expressiveness and the scalability**.
- 6) The sixth is that it should be **easy to use** in syntax and semantics.
- 7) Seventh, it should be compatible with other Web and industry standards. In particular, this includes XML and related standards (such as XML Schema and RDF) (**Compatibility with other standards**).
- 8) Eighth, it should support the development of multilingual ontologies, and potentially provide different views of ontologies that are appropriate for different cultures (**Internationalization**).

Table 3 summarizes the features provided by these semantic web ontology languages according to these 8 wishlists. The details are:

- **Ontology Sharing:** It is mainly maintained via the reasoning support. XSLT could define the translation rules among different ontologies defined in XML. It is not surprise to see that OIL, DAML+OIL and OWL could partially support this function due to the powerful reasoning service they equipped. Actually this “ontology sharing” is not clearly stated in the wishlist. Is this the partial ontology mapping or two ontologies have the simple parent-child relationship therefore it will be easily extended or shrink by just broadening or narrowing some terms or relations.

- **Ontology versioning:** It is very bad to see that almost none of the existing semantic web languages support such functions, except the early development of SHOE. Clearly the versioning is very much wanted for the next generation of Web. Therefore the later-on developed semantic web language, should be capable to bring such function in. Efforts are already started and going-on (Klein et al., 2002), and as a result of these efforts OWL now contains some versioning primitives.
- **Interoperability:** this overlaps the “Ontology sharing”, or “Ontology sharing” is the subset of “Interoperability”. Same to see here, it is the role of the reasoning. Partial mapping can be supported by defining the mapping rules between two ontologies.
- **Reasoning:** Reasoning is the main concern of the design of the semantic web language. Therefore RuleML is split out to become the independent initiative. But the balance between the power of reasoning and the complexity of the computing should be significantly considered here. Due to the special feature of the web, simple reasoning function is desirable for the reality.
- **Expressive & Scalability:** This is the common problem similar as the above. The more expressiveness it has, the less scalability it could have. How to balance in between also depends on the different user applications. Therefore the layered structure of the language is very important.
- **Easy to use:** The language must be simple and easy access if it is to have a realistic chance of widespread acceptance. Tools need to be provided to assist the ease of use of the language. The clear and formal documentation for the use of the language is important, which can contribute in a major fashion to the reusability and ease of understanding of code. Fail to do this means the dead-end of the language.
- **Compatibility:** The reuse of the knowledge is stressed almost everywhere, especially the language tower visioned by Tim Bernes-Lee, which is established on such compatibility among different layer of the languages. It is nice to see that most of the existing semantic web languages have the compatibility with the existing related languages.

Table 3. Comparison of ontology languages based on the W3C wishlists

Language	SHOE	Ontobroker	XML(s)	RDF(s)	OIL	DAML+OIL	OWL
sharing	No	No	XSLT can define different views	No	Reasoning can be used as the back support for integrating and sharing ontologies.	Reasoning is useful for ontology sharing.	Reasoning is useful for ontology sharing.
versioning	Providing ontology versioning function Maintaining each version of ontology as a separate web page and each instance must state the version to which it adheres.	No	No	No	No	No	Yes
interoperability	No	Horn-Logic can define the mapping rules	XSLT can define translation rules between two ontologies RuleML permits both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, and further inferential-transformational tasks.	Partially: mapping rules can be defined	Partially: mapping rules can be defined	Partially: mapping rules can be defined	Partially: mapping rules can be defined
reasoning	Reasoning supports are provided to handle revision problems	Inference engine is used to derive answers		Provided limited reasoning mechanism mainly for constraint checking.	Automatic consistency checking Cross-linking the inter-ontology relationships and check for the implied relationships.	Supporting design, maintenance and deployment of ontologies, especially the reasoning support from Description Logics	Based on an <i>open world assumption</i> (OWA). Equipped the same reasoning power from DAML+OIL
expressive & scalability	Limited expressive power	Much richer expressiveness	Expressive power in syntax	Limited expressive power in semantics.	Much rich expressive power in	Much rich expressive power in	Much rich expressive power in

				Lack of the necessary rich concept forming operators Not easy – due to the strange modeling theory	semantics. Layered feature for scalability	semantics.	semantics. Layered feature for scalability
Easy to use	Yes	Yes	Yes		Yes	Yes	Yes
compatibility	It is a superset of HTML, which adds the tags to semantic data. There is also an XML-based version of SHOE.	Annotation language is an extension of HTML. Representation language is based on Frame logic Query language is based on Frame logic	Compatible with HTML with the added value that the users can define their own tags.	Syntax is defined in XML	Designed on Description logic, Frame logics, and web standards (RDF(s) and XML(s)). Core OIL coincides with RDF Schema (except the reification features of RDF Schema).	The merge of DAML-ONT and OIL Based on existing Web standards (XML and RDF), Adding object and frame based systems, and description logic. Supporting the full range of XML Schema datatypes	Based on OIL and DAML+OIL OWL-Lite roughly consists of RDFS plus equality and 0/1-cardinality. DAML+OIL style RDF syntax
internationalization	Support different natural languages	Support different natural languages	Support different natural languages	Support different natural languages	Support different natural languages	Support different natural languages	Support different natural languages

Table 4. Detailed language feature comparison

Language	Syntax	Semantics	Layered structure	Redundancy	Compatibility	Special features	Tool support
SHOE	HTML	Extended HTML tags for defining ontology	No	Yes. Mixing ontology and data.	HTML	Versioning support,	SHOE tools
Ontobroker	Not available – due to commeralization	Extended HTML tags for defining ontology.	Not available – due to commeralization	No. Ontology and data are separated	HTML Description Logic Frame-based logic	Reasoning support	Ontobroker tools: webcrawler and inference engine
XML(s)	XML	The structure and semantics of document are interwoven.	No	No. Ontology and data are separated	XML	XML schemas provide a rich set of datatypes	Various XML-related tools: editor, parser and so on.
RDF (s)	XML (Schema) RDF (Schema)	RDFS define the semantics	No	No. Ontology and data are separated	XML XML schema	Too strong reification	Various RDF-related tools: editor, parsers.
OIL	XML (Schema) RDF (Schema) OIL	Rich expressiveness for defining ontologies	Yes OIL-Core OIL-Lite OIL-Standard OIL- instance OIL-heavy	No. Ontology and data are separated	XML (Schema) RDF (Schema) Description Logic Frame-based logic	Description Logic and Frame-based Logic	Various OIL related tools: editors, parsers.
DAML+OIL	XML (Schema) RDF (Schema) DAML+OIL	Rich expressiveness for defining ontologies	No	No. Ontology and data are separated	XML (Schema) RDF (Schema) OIL	DL based.	Various DAML+OIL related tools: editors, parsers.
OWL	XML (Schema) RDF (Schema)	Rich expressiveness for defining ontologies	Yes. OWL-Lite OWL-DL OWL-Full	No. Ontology and data are separated	XML (Schema) RDF (Schema) OIL DAML+OIL	Similar to DAML+OIL but with Frame-based	Validator

Table 5. Strength and weakness of various languages

Language	Strength	Weakness
SHOE	Supporting ontology versioning	. Redundancy is generated
Ontobroker	Real semantic web prototype with annotation, query and inference languages support	. Due to commercialization, many info is not available
XML(s)	Rich datatype XSLT defines the rules for transferring different ontologies	Label tree Define the syntax
RDF (s)	Define semantics	Strange modeling theory: - cyclical definition is not allowed (improved by latest development) - multiple domain and range are not allowed
OIL	Layered languages Rich inference support (description logic) Frame-based XML(s) and RDF(s) compatible	Inheritance some of the strange features of RDFS
DAML+OIL	DL-like XML(s) and RDF(s) compatible	Not frame-based
OWL	Layered languages Rich inference support (description logic) Frame-based XML(s) and RDF(s) compatible	Under development

Strength & Weakness

In this part, we try to put all these languages together to compare the strength or weakness of each based on some criteria, which are foreseen to be important for our Esperanto project. Table 4 summarizes the comparison as the followings:

- **Syntax:** All the latest developed languages are XML or RDF syntax based. This significantly enhances the compatibility among the languages. The existing tools for support XML or RDF can be easily reused for OIL, DAML+OIL and OWL. The layered language tower can be made possible due to the compatible syntax.
- **Semantics:** Adding semantics to the existing information is the main goal for the semantic web languages. But how to add and represent clear, explicit, machine-understandable semantics is not a trivial task. It is a clear go-direction for the design of languages, from SHOE to OWL. Different semantics needs different expressive power, therefore the layered structure is very essential as well.
- **Layered structure:** There will not be one single language, which fulfill all the needs of various web users. In a layered design, a simple core can accommodate simple taxonomies and relationships, while additional layers of expressivity, functionality, and complexity can be added for groups requiring more expressive power. Also the scalability and maintenance burden can be distributed to the different layers of the language, therefore can be alleviated accordingly. The languages with such clear layered structures are OIL and OWL.
- **Compatibility:** Please see the above section.
- **Special features:** SHOE has very basic versioning support, while Ontobroker provides efficient broker functions. XMLs defines very rich datatypes. RDFS has very strange modeling theory which brings the undeciability and too strong reification. OIL is frame-based, while DAML+OIL is not. OWL is much closer to OIL due to the frame-based feature.
- **Tools Support:** Software tools need to be developed for ontology language to really take off. Such tools should include the assist for user to easily create, use and maintain the ontologies. These tools should also provide user-friendly import and export facilities to link with other standard languages. Almost every language has various tool supports, ranging from editors, parsers up to reasoners.

The summary of the strength and weakness of the languages are shown in Table 5.

Other Analytical Dimensions

Web ontology languages have a number of dimensions along which we can evaluate and measure their appropriateness. Here in this deliverable, the comparison or evaluation is situated in the macro-level of the language structure. The much down-to-

earth comparison can be carried out to the detailed functions of the constructors of the languages, such as:

- negation; conjunction; disjunction;
- explicit universal or existential quantification/restriction;
- multi-valued slots;
- role properties such as transitivity, symmetry, reflexivity, functionality or the support of inverses; role hierarchies; number or cardinality restrictions on roles;
- both necessary and sufficient conditions for class membership;
- constraints. If the language supports constraints, how rich is the constraint language? Is the constraint language formally defined?
- rules;
- recursion;
- relations;
- multiple inheritance;
- axioms. Are facilities provided for the definition of axioms? If so, how expressive can those axioms be?
- template/default values;
- method slots (calculated values);
- Data types
- Instances. Does the language provide support for the encoding of instances as well as classes?

Corcho, O. and Gomez-Perez, A. (2002) give the systematic comparison at this level. The full-text of the paper has been attached here in Appendix 2.

Many difficult problems remain to be tackled – trust, time, and identity, to name but a few. However, an ontology language that is powerful enough to capture most desired information, while still being simple enough to allow agents to perform inferences based on the facts at their disposal at any particular moment, will represent a major challenge to the Semantic Web development.

5. CONCLUSION AND FUTURE PLAN

The Web ontology language must be able to describe and organize knowledge in the Web in the machine understandable way. It is a very complicated task, which can be simplified as defining categories by creating concepts, or frames, or classes, rules and so on. However, taken into the extreme, organizing knowledge requires the facilities of the logical formalism, which can deal with temporal, spatial, epistemic, and inferential aspects of knowledge. Implementations of the Web ontology language must provide these inference services, making them much more than just simple data storage and retrieval systems. The Web ontology language must have the clear model theory to secure the precise determination, while well model-theoretic semantics is really nothing more than a generalization of data models, like the relational data model or semi-structured data, that can deal with identifiable objects and uncertain and vague information. Tools are needed for the massive adoption of the languages themselves, therefore the easy-of-user for the programmers to implement all the essential features of the web ontology languages into the supporting tools are very important as well.

After the detailed the comparison and analysis within and between different web ontology languages in previous sections. Here we will give the summary and mention our future plan.

Cimino's dream for web ontology language

Cimino (1998) presents a collection of desiderata for Controlled Medical Vocabularies in the 21st Century. Some of them are quite interesting to state here for the design of Web ontology language:

- **Content.** Formal methodologies are required to support the addition of content.
- **Concept Orientation.** Terms in the ontology should correspond to at least one meaning (nonvagueness) and no more than one meaning (nonambiguity). In addition, meanings should correspond to no more than one term.
- **Concept Permanence.** Ontologies may evolve, but once a concept has been created, its meaning should be inviolate. This is not to say that concepts cannot be refined or redescribed. But unnecessary altering the means of concepts should be prevented. This eases the method for ontology management and versioning
- **Non-semantic Concept Identifier.** Unique identifiers should be provided for concepts.
- **Poly-hierarchy.** Multiple inheritance in subsumption hierarchies should be supported.
- **Formal Definitions.** This is very important for making machine understand the semantics of the data therefore reasoning or inference can be carried out by agents.

- **Multiple Granularities.** Vocabularies should be capable of supporting multiple levels of granularities. The layered structure for web ontology language is rather essential than normal because of the heterogeneous usage of the Web.
- **Multiple Consistent Views.** In tandem with a requirement for multiple granularities is the requirement for multiple views.
- **Evolve Gracefully.** Detailed descriptions of change are required in order to support the maintaining of the ontologies. Formal methodology on ontology versioning should be provided within the web ontology language itself.
- **Recognize Redundancy.** The redundancy will significantly reduce the power of reasoning and increase the complexity of maintenance. Proper redundancy is also necessary for provide difference views. The mechanism to recognize and balance the redundancy should be considered for designing the web ontology language.

Many of the above desiderata can be seen to resonate with the requirements put forward by the WebOnt committee.

Early approaches for the semantic web are based on rule-based approaches. Although there is a clear consensus that the current Web Ontology Language needs an extension in this direction. The *Rule Markup Language Initiative* may help to make a step into this direction.

Giving a real semantics to the semantic web language tower as sketched by Tim Berners-Lee in Figure 1 requires much more work. Currently many layering ideas oriented to syntactical and semantic extensions compete with each other. Working out the proper relationship will be much more challenging than just developing one layer for it.

The easy information access based on the success of the web has made it increasingly difficult to find, present, and maintain the information required by a wide variety of users. In response to this problem, many new research initiatives and commercial enterprises have been set up to enrich available information with machine understandable semantics. This **semantic web** will provide intelligent access to heterogeneous, distributed information, enabling software products to mediate between user needs and the information sources available. **Web Services** tackle with an orthogonal limitation of the current web. It is mainly a collection of information but does not yet provide support in processing this information, i.e., in using the computer as a computational device. Recent efforts around UDDI, WSDL, and SOAP try to lift the web to a new level of service. Software programs can be accessed and executed via the web. However, all these service descriptions are based on semi-formal natural language descriptions. Therefore, the human programmer need be kept in the loop and scalability as well as economy of web services are limited. Bringing them to their full potential requires their combination with semantic web technology. It will provide mechanization in service identification, configuration, comparison, and combination. **Semantic Web enabled Web Services** have the potential to change our life in a much higher degree as the current web already did (Figure 2). We are

currently in the process of extending UPML [Fensel et al, 2003] to a full-fledged **Web Service Modeling Language (WSML)**.

In the continuing version of the deliverables in the Workpackage, we will keep the sharp eye on the newly development and evolving in the semantic web ontology language area and document and discuss the pros and cons to provide the significant support for the Esperanto tool development.

REFERENCES

- Bechhofer, S.; Goble, C.; and Horrocks, I. (2001). DAML+OIL is not enough. In *Proc. of the First Semantic Web Working Symposium (SWWS'01)*, 151–159. CEUR Electronic Workshop Proceedings, <http://ceur-ws.org/>.
- Berners-Lee, T. (1999). *Weaving the Web*. San Francisco: Harper.
- Borgida, A., and Patel-Schneider, P. F. (1994). A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research* 1:277–308.
- Calvanese, D.; De Giacomo, G.; Lenzerini, M.; Nardi, D.; and Rosati, R. (1998). Information integration: Conceptual modeling and reasoning support. In *Proc. of CoopIS'98*, 280–291.
- Cimino, J. J. (1998). Desiderata for Controlled Medical Vocabularies in the Twenty-First Century. *Methods Inform Med* 1998; 37: 394-403.
- Gómez-Pérez, A. and Corcho, O. (2002). Ontology Languages for the Semantic Web. IEEE Intelligent Systems, January/February
- Fensel, D.; Angele, J.; Decker, S.; Erdmann, M.; Schnurr, H.-P.; Staab, S.; Studer, R. and Witt, A. (1999): On2broker: Semantic-Based Access to Information Sources at the WWW. In *Proceedings of the World Conference on the WWW and Internet (WebNet 99)*, Honolulu, Hawaii, USA, October 25-30, 1999.
- Fensel, D.; van Harmelen, F.; Horrocks, I.; McGuinness, D. L. and Patel-Schneider, P. F. (2001). OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems* 16(2): 38–45.
- Fensel, D.; Motta, E.; van Harmelen, F.; Benjamins, V. R.; Crubezy, M.; Decker, S.; Gaspari, M.; Groenboom, R.; Grosso, W.; Musen, M.; Plaza, E.; Schreiber, G.; Studer, R. and Wielinga, B (2003): The Unified Problem-solving Method Development Language UPML, *Knowledge and Information Systems (KAIS): An International Journal*, 5(1), 2003.
- Grosso, W. E.; Eriksson, H.; Ferguson, R. W.; Gennari, J. H.; Tu, S. W.; and Musen, M. A. (1999). Knowledge modelling at the millenium (the design and evolution of protégé-2000). In *Proc. Of Knowledge acquisition workshop (KAW-99)*.
- Heflin, J. and Hendler, J. (2000). Searching the Web with SHOE. In AAI-2000 Workshop on AI for Web Search.
- Heflin, J. and Hendler, J. (2001). A Portrait of the Semantic Web in Action. *IEEE Intelligent Systems*, 16(2).

Heflin, J. and Hendler, J. and Luke, S. (1999). SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078 (UMIACS TR-99-71).

Heflin, J.; Volz, R. and Dale, J. (2002). Requirements for a Web Ontology Language. W3C Working Draft 08 July 2002.

Horrocks, I. 1998. Using an expressive description logic: FaCT or fiction? In *Proc. of KR-98*, 636–647.

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

M. Klein, D. Fensel, F. van Harmelen, and I. Horrocks: The relation between ontologies and schema-languages: Translating OIL-specifications in XML-Schema. In *Proceedings of the Workshop on Applications of Ontologies and Problem-solving Methods, 14th European Conference on Artificial Intelligence ECAI'00*, Berlin, Germany August 20-25, 2000.

Klein, M.; Kiryakov, A.; Ognyanov, D and Fensel, D. (2002). Ontology Versioning and Change Detection on the Web. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, Sigüenza, Spain, October 1-4, 2002.

Lloyd, J. W. and Topor, R. W (1984). Making Prolog more Expressive, *Journal of Logic Programming*, 3:225-240.

Luke, S.; Spector, L.; Rager, D. and Hendler, J. (1997). Ontology-based Web Agents. In *Proceedings of First International Conference on Autonomous Agents 1997*.

Nejdl, W.; Wolpers, M. and Capella, C. (2000). The RDF Schema Revisited. In Ebert J. et al. (eds.), *Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik, Modellierung 2000*, St. Goar, April 5-7, 2000.

QL (1998). *Proceedings of W3C Query Language Workshop (QL'98) - The Query Languages Workshop*, Boston, Massachusetts, December 3-4, 1998. <http://www.w3.org/TandS/QL/QL98/>.

Appendix 1

Since the OWL specification is constantly evolving a link to the OWL Lite expanded summary is provided: <http://www.w3.org/TR/owl-features/>