Web Ontology Language Design and Related Tools: A Survey

Farheen Siddiqui
(Hamdard University, India
fsiddiqui@jamiahamdard.ac.in)

M. Afshar Alam
(Hamdard University, India
aalam@jamiahamdard.ac.in)

Abstract— **The foundation of the semantic web lies firmly on a strong foundation of Web Ontology Language(OWL). OWL was developed by World Wide Web Consortium (W3C), Web Ontology Working Group and has features from several families of representation languages. OWL also shares many characteristics with RDF, the W3C base of the Semantic Web. This paper will discuss how the "thought process" behind OWL can be connected to its older formalisms, with modifications due to the functionalities that OWL had to give. A smooth move from RDF to OWL is made possible by handling every possible situation of RDF in OWL. Also various species of OWL are discussed with their features and restrictions. The recent development in ontology engineering and the immense tool support for OWL is a conformance of its acceptability.**

Index Terms— **Ontologies, Semantic Web, Description Logics, Frames, RDF**

I. Introduction

Ontologies have become a very popular topic, not only in AI but also in other disciplines of computing. There are also efforts focused on developing ontologies in many other branches of science and technology. Hence ontologies are growing fast into a distinct scientific field with its own theories, formalisms, and approaches. The major purpose of ontologies is not to serve as vocabularies and taxonomies; it is knowledge sharing and knowledge reuse by applications. Every ontology provides a description of the concepts and relationships that can exist in a domain and that can be shared and reused among intelligent agents and applications. Moreover, working agents and applications should be able to communicate such ontological knowledge. Shared ontologies let us build specific knowledge bases that describe specific situations but clearly rely on the same underlying knowledge structure and organization.

OWL 2 [1] is ontology language for the Semantic Web, developed by the World Wide Web Consortium (W3C) Web Ontology Working Group. The OWL Web Ontology Language is intended to provide a language that can be used to describe the classes and relations between them that are inherent in Web documents and applications. In simpler words OWL represents information about categories of objects pertaining to a specific domain and their interdependencies—the "domain mode" that is often called an ontology. OWL can also represent information about the objects themselves—the sort of information that is often thought of as data.

OWL is an effort in W3C's Semantic Web activity, and is compatible with XML and RDF. OWL is primarily used to formalize a domain by defining classes and properties of those classes, defining individuals and assert properties about them, and reason about these classes and individuals to the degree permitted by the formal semantics of the OWL language. Before the development of OWL there was already several ontology languages designed for use in the Web thus OWL had to maintain a upward compatibility with earlier web languages to ensure its smooth adaptability.

The multiple influences on OWL resulted in some difficult trade-offs. OWL is devised in such a way that it could be shown to have various desirable features, while still retaining sufficient compatibility with its roots. This paper describes some of the trade-offs and design decisions that had to be made by the Web Ontology Working Group during the design of OWL and its tool support After a brief introduction and quick survey of OWL in Section II , in Section III we present influences on design of OWL and section IV discusses different ontology language that preceded OWL. The issues related to OWL design are discussed in Section V and section VI gives details of how these issues were addressed .OWL versions are discussed in VII and section VIII discusses current trends in ontology engineering and tool support.

doi:10.4304/jetwi.3.1.47-59

II. OWL Overview

Ontology is a term borrowed from philosophy that refers to the science of describing the kinds of entities in the world and how they are related. Informally, the ontology of a certain domain is about its terminology (domain vocabulary), all essential concepts in the domain, their classification, their taxonomy, their relations (including all important hierarchies and constraints), and domain axioms. According to Dragan Gaˇsevi´c · etal [9] I"f someone who wants to discuss topics in a domain D using a language L, an ontology provides a catalog of the types of things assumed to exist in D;the types in the ontology are represented in terms of the concepts, relations, and predicates of L".

Both formally and informally, the ontology is an extremely important part of the knowledge about any domain. Moreover, the ontology is the fundamental part of the knowledge, and all other knowledge should rely on it and refer to it. In the context of the Semantic Web, ontologies are expected to play an important role in helping automated processes (so called "intelligent agents") to access information. Generally ontologies provide a number of useful features for intelligent systems, as well as for knowledge representation in general and for the knowledge engineering process. In particular, ontologies are expected to be used to provide structured vocabularies that explicate the relationships between different terms, allowing intelligent agents (and humans) to interpret their meaning flexibly yet unambiguously. Ontology specifies terms with unambiguous meanings, with semantics independent of reader and context. Translating the terms in ontology from one language to another does not change the ontology conceptually. Thus ontology provides a vocabulary and a machine-processable common understanding of the topics that the terms denote. The meanings of the terms in ontology can be communicated between users and applications.

The OWL Web Ontology Language is a language for defining and instantiating Web ontologies. OWL ontology may include descriptions of classes, properties and their instances. Given such ontology, the OWL formal semantics specifies how to derive its logical consequences, i.e. facts not literally present in the ontology, but entailed by the semantics. These entailments may be based on a single document or multiple distributed documents that have been combined using defined OWL mechanism

Terms whose meaning is defined in ontologies can be used in semantic markup that describes the content and functionality of web accessible resources [2]. Ontologies and ontology-based semantic markup could be used in

– E-commerce where they can provide a common domain model of products on form of product ontology for both sellers and buyers

– search engines, where they can help in finding pages that contain semantically similar but syntactically different words and phrases

– Web services, where they can provide rich service descriptions that can help in locating suitable services.

In order to support these and other usage scenarios, OWL takes the basic fact-stating ability of RDF [8] and the class- and property-structuring capabilities of RDF Schema [6] and extends them in important ways. OWL can declare classes, and organize these classes in a "subclass" hierarchy, as can RDF Schema. OWL classes can be specified as logical combinations (intersections, unions, or complements) of other classes, or as enumerations of specified objects, going beyond the capabilities of RDFS. OWL can also declare properties, organize these properties into a "subproperty" hierarchy, and provide domains and ranges for these properties, again as in RDFS. The domains of OWL properties are OWL classes, and ranges can be either OWL classes or externally-defined datatypes such as string or integer. OWL can state that a property is transitive, symmetric, functional, or is the inverse of another property, here again extending RDFS.

OWL can express which objects (also called "individuals") belong to which classes, and what the property values are of specific individuals. Equivalence statements can be made on classes and on properties, disjoint statements can be made on classes, and equality and inequality can be asserted between individuals. However, the major extension over RDFS is the ability in OWL to provide restrictions on how properties behave that are local to a class. OWL can define classes where a particular property is restricted so that all the values for the property in instances of the class must belong to a certain class (or datatype); at least one value must come from a certain class (or datatype); there must be at least certain specific values; and there must be at least or at most a certain number of distinct values. For example, using RDFS we can

– declare classes like StudyProgaram, Courses, and student;

– state that RegularStudent is a subclass of Student;

With OWL we can additionally

– RegularStudent and ParttimeStudent are disjoint classes;
– RegularStudent is equivalent to Student with EnrollIn property value from RegularCourse
– Instructs and HasInstructor are inverse properties
– A StudyProgram should have minimum of 5 Courses
– Enrollment is a functional property.

The core education ontology will define concept such as study Programs, Courses, students and Instructors.(to link a study program with its courses and courses with its instructors)There could be a subtype of study Program such as Online program and Regular Program Using OWL logical connectors it is further possible to define classes by their logical characteristics such as regular Student could be defined as student who is enrolled in some Regular Study Program Also OWL allows classes to be defined disjoint or equivalent by logical connectors equivalent and disjoint. For example the class Regular Student is equivalent to class Student will Enroll In

property value belonging to some regular course. Also the class Regular Student can be defined Disjoint to the class Online Student. These defined classes allow reasoners to automatically classify existing domain objects into matching types (classes).

The above shows that OWL is quite a sophisticated language. OWL has both RDF/XML exchange syntax and an abstract frame-like syntax, and it has three named sublanguages. This multiplicity is the direct result of trying to satisfy a large number of sometimes conflicting influences and requirements, as will be discussed subsequently in this paper.

III. INFLUENCES ON OWL

As mentioned above, the design of OWL has been subject to a variety of influences. These included influences from established formalisms and knowledge representation paradigms, influences from existing ontology languages, and influences from existing Semantic Web languages. Some of the most important influences on the design of OWL came, via its predecessor DAML+OIL, from Description Logics, from the frames paradigm, and from RDF. In particular, the formal specification of the language was influenced by Description Logics, the surface structure of the language (as seen in the abstract syntax) was influenced by the frames paradigm, and the RDF/XML exchange syntax was influenced by a requirement for upwards compatibility with RDF.

The major influence on the design of OWL was the requirement to maintain the maximum upwards compatibility with existing web languages and in particular with RDF [3]. On the face of it this requirement made good sense as RDF (and in particular RDF Schema) already included several of the basic features of a class and property based ontology language, e.g., it allows subclass and subproperty relationships to be asserted. Moreover, the development of RDF preceded that of OWL, and it seemed reasonable to try to appeal to any user community already established by RDF. In order to provide maximum upwards compatibility, however, it was also thought necessary to ensure that the semantics of OWL ontologies was also consistent with the semantics of RDF. This proved to be difficult given the greatly increased expressive power provided by OWL. Each of these influences will be examined in more detail in the following sections.

A. Description Logics

Description Logics are a family of class-based (concept-based) knowledge representation formalisms [9]. They are characterized by the use of various constructors to build complex classes from simpler ones, an emphasis on the decidability of key reasoning problems, and by the provision of sound, complete and (empirically) tractable reasoning services. Thus an entire group of languages for knowledge representation, based on description logics, is of particular interest in the context of ontology representation and development. The semantics of the underlying concepts of these languages identifies them as decidable fragments of first-order logic. Developing a knowledge base using a description logic language means setting up a terminology (the vocabulary of the application domain) in a part of the knowledge base called the TBox, and assertions about named individuals (using the vocabulary from the TBox) in a part of the knowledge base called the ABox. The vocabulary consists of concepts and roles. Concepts denote sets of individuals. Roles are binary relationships between individuals. There are atomic concepts and roles (names of concepts and roles) and complex concepts and roles (terms for concepts and roles). The complex concepts are built using descriptions expressed in the corresponding description logic language and are assigned names in the TBox. For example, if StudyProgram and Course are atomic concepts and hasCourse d is an atomic role (a relation), part of a TBox defining complex concepts about relationships may look like this :

$$\text{RegularStudent} \equiv \text{Student} \cap \text{EmrollIn.RegularCourse}$$

In fact, the TBox defines (in a general way) semantic relationships between individuals introduced in the ABox and their properties. In other words, the ABox describes a specific state of affairs in the world in terms of the concepts and roles defined in the TBox. An ABox corresponding to the above TBox might be

RegularStudent(JAMES)

An intelligent system with a knowledge base structured as a TBox–ABox pair can reason about its terminology and assertions. It can determine whether a description in the TBox is satisfiable (i.e., noncontradictory), and whether there is a subsumption relationship between two descriptions (i.e., one is more general than the other). For example, in the above TBox Person subsumes Woman, and both Parent and Woman subsume Mother. Two important reasoning tasks about assertions in the ABox are to determine whether the assertions imply that a particular individual is an instance of a given concept description, and whether a set of assertions is consistent (whether it has a model). From a pragmatic point of view, consistency checks of sets of assertions and satisfiability checks of concept descriptions help to determine whether a knowledge base is meaningful at all. Subsumption tests help to organize the terminology into a meaningful hierarchy of concepts according to their generality. Each test can be also interpreted as a query about objects of interest; it retrieves the set of individuals that satisfies the test.

Description Logics [4], and insights from Description Logic research, had a strong influence on the design of OWL, particularly on the formalization of the semantics, the choice of language constructors, and the integration of data types and data values. In fact OWL DL and OWL Lite (two of the three species of OWL) can be viewed as expressive Description Logics, with an ontology being equivalent to a Description Logic knowledge base. Like OIL and DAML+OIL, OWL uses a Description Logic style model theory to formalize the meaning of the

language. This was recognized as an essential feature in all three languages, as it allows ontologies, and information using vocabulary defined by ontologies, to be shared and exchanged without disputes as to precise meaning. The need for this kind of formality was reinforced by experience with early versions of the RDF and RDFS specification, where a lack of formality soon led to arguments as to the meaning of language constructs such as domain and range constraints [10]. In order to avoid such problems, the meaning of RDF is now also defined in terms of a model theory [11].

Another advantage of formalizing the meaning of the language in this way is that automated reasoning techniques can be used to check the consistency of classes and ontologies, and to check entailment relationships. This is crucial if the full power of ontologies is to be exploited by intelligent agents, and the ability to provide such reasoning support was a key design goal for OWL.

B. Frames Paradigm

In frame based languages, each class is described by a frame. The frame includes the name of the class, identifies the more general class (or classes) that it specializes, and lists a set of "slots". A slot may consist of a property-value pair or a constraint on the values that can act as slot "fillers" (in this context, value means either an individual or a data value). This structure was used in the OIL (Ontology Interface Language), with some enrichment of the syntax for specifying classes and slot constraints so as to enable the full power of a Description Logic style language to be captured. In addition, property frames were used to describe properties, e.g., specifying more general properties, range and domain constraints, transitivity and inverse property relationships. A class frame is semantically equivalent to a Description Logic axiom asserting that the class being described by the frame is a subclass of each of the classes that it specializes and of each of the property restrictions corresponding to the slots. As well as a richer slot syntax, OIL also offered the possibility of asserting that the class being described by the frame was exactly equivalent to the relevant intersection class, (i.e., that they were mutually subsuming). A property frame is equivalent to a set of axioms asserting the relevant sub property relationships, range and domain constraints etc.].

The traditional frame-based languages lacked precise characterization of their semantics. Moreover, discrimination between different types of knowledge embedded in a frame system was not clear. As a result, every frame-based system behaved differently from the others (which were developed using different frame-based languages), in many cases despite virtually identical-looking components and even identical relationship names [Baader et al., 2003]. As a consequence, different systems could not interoperate and share knowledge. Later frame languages introduced more formal semantics, retaining the hierarchical concept structures and ease of representation and simultaneously improving the efficiency of reasoning and the representational rigor. Another disadvantage of frame-based languages is inadequate way in which they deal with procedural knowledge. The way they do this is usually to use some limited arithmetic, plus calling procedures and functions written in a procedural language and attached to slot facets. The procedural knowledge encoded in this other language is not represented in a frame-based way – it is hard-coded in the corresponding function/procedure. As a consequence, the resulting systems can only reason with that knowledge, but not about it.

In the Semantic Web context, where users with a wide range of expertise might be expected to create or modify ontologies, readability and general ease of use are important considerations for an ontology language. In the design of OIL, one of the languages on which OWL is based; these requirements were addressed by providing a surface syntax based on the frames paradigm. Frames group together information about each class, making ontologies easier to read and understand, particularly for users not familiar with (Description) Logics. The frames paradigm has been used in a number of well known knowledge representation systems including the Prot´eg´e ontology design tool [13] and the OKBC knowledge model [8]. The design of OIL was influenced by XOL [14]—a proposal for an XML syntax for OKBC Lite (a cut down version of the OKBC knowledge model).

The formal specification and semantics of OWL are given by an abstract syntax [13] that have been heavily influenced by frames in general and by the design of OIL in particular. In the abstract syntax, axioms are compound constructions that are very like an OIL-style frame. For classes, they consist of the name of the class being described, a modality of "partial" or "complete" (indicating that the axiom is asserting a subclass or equivalence relationship respectively), and a sequence of property restrictions and names of more general classes. Similarly, a property axiom specifies the name of the property and its various features. The frame style of the abstract syntax makes it much easier to read (compared to the RDF/XML syntax), and also easier to understand and to use. Moreover, abstract syntax axioms have a direct correspondence with Description Logic axioms, and they can also be mapped to a set of RDF triples.

C. RDF Syntax

The third major influence on the design of OWL was the requirement to maintain the maximum upwards compatibility with existing web languages and in particular with RDF [3]. On the face of it this requirement made good sense as RDF (and in particular RDF Schema) already included several of the basic features of a class and property based ontology language, e.g., it allows subclass and sub property relationships to be asserted. Moreover, the development of RDF preceded that of OWL, and it seemed reasonable to try to appeal to any user community already established by RDF. It may seem easy to meet this requirement simply by giving OWL an RDF based syntax. In order to provide maximum upwards compatibility, however, it was also thought necessary to ensure that the semantics of OWL ontologies was also consistent with the semantics of

RDF. This proved to be difficult given the greatly increased expressive power provided by OWL. This will be discussed in more detail in Section 5.

IV. OWL PREDECESSORS

OWL came into existence after development of RDFS, SHOE, OIL, DAMLONT and DAML+OIL and was soon adapted by W3C as standard for ontology description. DAML+OIL in particular was a major influence on OWL, and members of Web Ontology working group emphasized that the design of OWL should be based on DAML+OIL. DAML+OIL in turn were heavily influenced by the OIL language, with additional influence from work on DAML-ONT and RDFS.

A. Simple HTML Ontology Extensions

One of the first attempts at defining an ontology language for deployment on the Web was SHOE [16]. SHOE is a frame-based language with an XML syntax that could be safely embedded in existing HTML documents. SHOE used URI references for names, an important innovation (see Section 7) that was subsequently adopted by DAML-ONT and DAML+OIL. Giving the authors the ability to embed knowledge directly into HTML pages, making it also simple for user agents and robots to retrieve and store knowledge, was the goal of the so-called Simple HTML Ontology Extension (SHOE). This approach allows authors to add semantic content to web pages, relating the context to common ontologies that provide contextual information about the domain [16]. Most web pages with SHOE annotations tend to have tags that categorize concepts; therefore there is no need for complex inference rules to perform automatic classification [20]. This approach extends HTML with a set of object-oriented tags to provide structure for knowledge acquisition. It associates meaning with content by committing web pages to existing ontologies. These ontologies permit the discovery of implicit knowledge through the use of taxonomies and inference rules, allowing information providers to encode only the necessary information into their web pages. An ontology tag delimits the machine-readable portion of the ontology.

SHOE focuses on the problem of maintaining consistency as the ontologies evolve. In [21] the use of SHOE in a real world internet application is described. Tools for annotating pages, information gathering tasks,and querying are provided. SHOE also placed emphasis on the fact that ontologies would be tightly interlinked and subject to change. Consequently, SHOE included a number of directives which allowed importing of other ontologies, local renaming of imported constants, and stating versioning and compatibility information between ontologies. This line of thinking has influenced the extra-logical vocabulary of OWL that is designed to partially deal with such issues. SHOE was of lesser influence on the syntactic and semantic design of OWL since it was not based on RDF, and did not come with a formal semantics

B. DAML-ONT

In 1999 the DARPA Agent Markup Language (DAML) program3 was initiated with the aim of providing the foundations of a next generation "semantic" Web [19]. As a first step, it was decided that the adoption of a common ontology language would facilitate semantic interoperability across the various projects making up the program. RDFS (which had already been proposed as a W3C standard) was seen as a good starting point, but was not sufficiently expressive to meet DAML's requirements. A new language called DAML-ONT was therefore developed that extended RDF with language constructors from object-oriented and frame-based knowledge representation languages. DAML-ONT was tightly integrated with RDFS, and while this was useful from a compatibility viewpoint, it led to some serious problems in the design of the language. Like RDFS, DAML-ONT suffered from an inadequate semantic specification, and it was soon realized that this could lead to disagreements, both amongst humans and machines, as to the precise meaning of terms in DAMLONT ontology. Moreover, DAML-ONT property restrictions had, like those of RDFS, global rather than local scope, and while this was reasonable for the domain and range constraints provided by RDFS, global cardinality constrains, for example, are difficult to understand and of doubtful utility—in fact it seems likely that this would have been recognized as a design flaw if the semantics of the language had been adequately formalized.

C. DAML-ONT

At around the same time that DAML-ONT was being developed, a group of (largely European) researchers with aims similar to those of the DAML researchers had designed another Web oriented ontology language called OIL (the Ontology Inference Layer) [12]. OIL was the first ontology language to combine elements from Description Logics, frame languages and web standards such as XML and RDF. OIL placed a strong emphasis on formal rigor, and the language was explicitly designed so that its semantics could be specified via a mapping to the SHIQ description logic [23]. The structure of the language was, however, frame-based, using compound class "definitions" in the style described in Section 3.2. OIL had both XML and RDF syntaxes, but although the RDF syntax was designed to maintain compatibility with RDFS, it did not concern itself with the precise details of RDF semantics, which had not at that time been formally defined.

D. DAML +OIL

It became obvious to both the DAML-ONT and OIL groups that their objectives could best be served by combining their efforts, the result being the merging of DAML-ONT and OIL to produce DAML+OIL. The development of DAML+OIL was undertaken by a committee largely made up of members of the two language design teams, and rather grandly titled the Joint US/EU ad hoc Agent Markup Language Committee.5 The merged language has a formal semantics given by its

own DL style model theory instead of via a translation into a suitable DL. The DL derived language constructors of OIL were retained in DAML+OIL, but the frame structure was largely discarded in favour of DL style axioms, which were more easily integrated with RDF syntax. Influenced by DAML-ONT, DAML+OIL is more tightly integrated with RDF. DAML+OIL, however, only provided a meaning for those parts of RDF which were consistent with its own syntax and DL style model theory. This did not seem to be too much of a problem given that RDF did not at that time have a formally specified meaning of its own, but was the cause of serious difficulties when DAML+OIL was used as the basis for OWL.

V CHALLANGES FOR OWL

The multiple influences on OWL have actually created number of chanllages for its developers. For example it a tradeoff between RDF/XML as the official OWL syntax and having an redable and more user friendly syntax. Some of these problems arise from a need to maintain a upward compatibility to OWL predecessors.

A. Syntactic challanges

For a number of reasons, including maintaining connections to frames and Description Logics, OWL should have an easy-to-read syntax that can be easily understood and easily created. However, it was a requirement of OWL that it use XML as its normative syntax, and, moreover, use XML in the same way as it is used in RDF [8]. This requirement had already been addressed by OIL and, later, by DAML+OIL: OIL has both an RDF/XML and XML syntax [15], while DAML+OIL has only an RDF/XML syntax [9]. Taken just as a syntax for OWL, RDF in the form of RDF/XML has a number of problems. These problems can be overcome, but they do make OWL more complex that it might otherwise be.

One problem is that RDF/XML is extremely verbose. Compare for example, information about a class as it would be given in a Description Logic syntax

Student = Person ∩>1 enrolledIn

(a Student is a Person who is enrolledIn at least 1 thing), with how it would most naturally be written using the OWL RDF/XML syntax

```
<owl:Class rdf:ID="Student">
<owl:intersectionOf rdf:parsetype="Collection">
<owl:Class rdfs:about="Person" />
<owl:Restriction>
<owl:onProperty rdf:resource="enrolledIn" />
<owl:minCardinality rdfs:datatype="&xsd;Integer">
1
</owl:minCardinality>
</owl:Restriction>
<owl:intersectionOf>
</owl:Class>
```

Another problem is that RDF breaks everything down into RDF triples [17] This means that many OWL constructs, such as property restrictions, have to be encoded as several triples. OWL generally uses an encoding similar to that used by DAML+OIL. For example, an OWL value restriction that would be written in Description Logic syntax as 9child.person (the class whose instances have some child that is a person) is encoded as two RDF triples something like

:x owl:onProperty ex:child .

:x owl:someValuesFrom ex:Person .

where _:x is a syntactic placeholder for the restriction as a whole.

A third problem is that all RDF triples are independent. This means, for example, that as far as RDF is concerned there is no requirement that the two above triples must always occur together. Similarly, there is no requirement that there not be extra triples, so adding

:x owl:onProperty ex:friend .

:x owl:allValuesFrom ex:Doctor .

to the above two triples cannot be ruled out in RDF.

A fourth problem is that RDF triples are all accessible. This means that circular and other unusual structures cannot be ruled out. For example, there is no problem in RDF with collections of triples like

:x owl:onProperty ex:child .

:x owl:allValuesFrom :x .

These issues are not addressed in OIL, which provides no guidance as to what should happen for collections of triples that don't match the syntax productions of the language. DAML+OIL take a different approach, allowing unusual constructions but declining to give them a DAML+OIL meaning. OWL has roughly followed the DAML+OIL solution, but with several modifications.

B. Semantic Challanges

Once issues of syntax have been addressed, issues related to meaning still remain. RDF provides a meaning for every triple, so if OWL is to be considered to be an extension of RDF, the meaning that OWL provides for triples needs to be an extension of this RDF meaning. This was not as much of an issue when OIL and DAML+OIL were designed, as the meaning of RDF was not very well specified. OIL in particular does not bother to relate the RDF meaning of its RDF/XML syntax to the OIL meaning of this syntax—the RDF/XML syntax for some OIL constructs does more-or-less line up with the RDF meaning of these constructs but this is by no means the case for all such constructs. For example, OIL has a special property (oil:hasSlotConstraint) used to relate a class to its slots, but the RDF meaning [11] of this property, namely the standard meaning assigned to any RDF triple is ignored by the OIL semantics.

DAML+OIL do a better job of abiding by the RDF meaning of its syntax. The DAML+OIL model theory [18] includes a semantic condition for triples that is close to the RDF meaning (as defined at that time) for triples. Further, DAML+OIL uses the built-in RDF and RDFS vocabulary to a greater extent than does OIL, and uses it in a way generally compatible with the RDF or RDFS

meaning (as defined at that time) for this vocabulary. For example DAML+OIL uses rdfs:subClassOf to relate classes to superclasses, including DAML+OIL restrictions, whereas OIL uses oil:hasSlotConstraint in some of these situations. Even when DAML+OIL was being developed, however, there were some aspects of the meaning of RDFS that could not be reconciled with the appropriate meaning in DAML+OIL. In particular, RDFS [5] then had an unusual meaning for domains and ranges of properties. Only a single range was permitted for properties and multiple domains were treated disjunctively. For example,

ex:foo rdfs:domain ex:Person .
ex:foo rdfs:domain ex:Rock .

would allow both people and rocks to participate in the foo property. This disjunctive reading of domains caused problems for the DAML+OIL semantics so a choice was made to change this to a conjunctive reading and allow multiple domains and ranges, both with a conjunctive reading. As part of its clean-up of the RDF and RDFS semantics, the RDF Core working group has decided to make this change, eliminating a problem for OWL. While cleaning up problems with RDF and RDFS, the RDF Core working group also decided to put RDF on a firmer semantic ground. It did this by providing a model theory for RDF and RDFS, along with a standard treatment of inference for RDF and RDFS. This has meant that there is now more meaning provided for RDF and RDFS that OWL has to be compatible with. In particular, all the triples that are used to encode the OWL syntax now have RDF meaning, and this RDF meaning has to be taken into account by OWL if the semantics of OWL are to be fully compatible with those of RDF and RDFS. Neither OIL nor DAML+OIL provided a standard theory of inference. This was common in the formalisms that influenced OIL and DAML+OIL.

Frames generally provided an interface to the internal data structures in lieu of any other inferences or even queries. Description Logics do provide a formal theory of querying, but this is somewhat different from a standard theory of inference. The difference is that Description Logic querying could have been defined for DAML+OIL in a way that would have helped to hide the RDF meaning of triples. For example, asking whether an individual belonged to a class could add the syntax used to specify the class to the premises of the query. However, a standard theory of inferencing cannot do this. The effects of this change can be seen in a simple example. Given the following information

ex:John rdf:type ex:Student .
ex:John rdf:type ex:Employee .

it would have been fairly easy to arrange it so that asking whether John belonged to the intersection of student and employee first ensured that this intersection existed and then asked whether John belonged to it. However, turning this into an entailment requires the above information to entail

_:c owl:intersectionOf _:l1 .
_:l1 rdf:first ex:Student .
:l1 rdf:rest :l2 .
_:l2 rdf:first ex:Employee .
_:l2 rdf:rest rdf:nil .
ex:John rdf:type _:c .

which, because of the RDF meaning ascribed to all triples, requires the existence of the triples that encode the syntax. OWL thus has had to develop a method that augments the new RDF semantics just enough to support the above inferences without being too strong.

C. Expresive Power

Because many things were expected of, there were many demands for expressive power going beyond that generally provided by Description Logics. For example, many users wanted to be able to associate information with classes and properties and to make classes belong to other classes, as is possible in RDF. Similarly, there were many demands for expressive power going beyond RDF and RDFS. For example, many users wanted to be able to provide local typing for property values, as is possible in Description Logics. OWL had to be designed to somehow allow these sorts of expressivity while still retaining connections to its roots. When DAML+OIL was developed, the only datatype supported by RDF was literals: roughly undifferentiated values given as strings. DAML+OIL thus had to provide its own solution for datatypes, and did so by allowing the use of XML Schema datatypes [19]. However, any reasonable solution to datatyping that uses only RDF syntax needs help from RDF (i.e., an extension to RDF syntax), and thus the DAML+OIL solution remained incomplete. Recently RDF has added its own version of datatyping, similar to, but different from, the DAML+OIL solution. OWL has thus needed to move from DAML+OIL data typing to RDF data typing.

D. Computational challanges

One aspect of OWL that distinguishes it from RDF and RDFS is that it supports a rich set of inferences. Some of these inferences are quite obvious, such as the example given above about students and employees, and thus appear to be easy to compute. Other inferences supported by OWL, however, are quite complex, requiring, e.g., reasoning by cases and following chains of properties. Taking all the representational desires for OWL together would have created formalism where key inference problems were undecidable. For example, allowing relationships to be asserted between property chains (such as saying that an uncle is precisely a parent's brother) would make OWL entailment undecidable. 9 In addition, some aspects of RDF, such as the use of classes as instances, interact with the rest of OWL to create computational difficulties, taking OWL beyond the range of practical algorithms and implemented reasoning systems. OWL has thus had to provide a solution for these computational issues while still retaining upwards compatibility with RDF and RDFS.

VI SOLUTION

The Web Ontology working devoted a lot of efforts in overcoming the basic tensions underlying the above problems. The difficulty lay not in each problem in isolation, but in the combination of all the above problems and the constraints placed on the design of OWL. It would have been much easier, for example, to meet all the above requirements if only OWL could have used an extension of the RDF syntax. If this had been allowed, OWL could have added new, natural syntax for its constructs whose semantics would not have been required to carry along an RDF triple meaning. Nevertheless a viable solution has been found that satisfies all the above requirements. Or, actually, it is more accurate to say that three solutions have been found, each of which satisfies almost all of the above requirements.

OWL DL: If friendly syntax or decidable inference is considered of primary importance, then OWL DL, a version of OWL with decidable inference that can be written in a frame or Description Logic manner, is appropriate.

OWL Lite: If an even-simpler syntax and more tractable inference is considered of primary importance, then OWL Lite, a syntactic subset of OWL DL, is appropriate.

OWL Full: If compatibility with RDF and RDFS is considered of primary importance, then OWL Full, a syntactic and semantic extension of RDFS, is appropriate.

The next section provides a more-detailed description of these versions (species) of OWL, and explains how the problems described above have been overcome.

Each of these sublanguages is an extension of its simpler predecessor, both in what can be legally expressed and in what can be validly concluded. The following set of relations hold. Their inverses do not.

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

A. Readability

As shown by the examples above, OWL is not very readable when written as RDF/XML or even as RDF triples. Part of this problem is that RDF/XML is extremely verbose, but the major part of the readability problem is the encoding of OWL constructs into RDF/XML or RDF triples. In part to address this problem, an abstract syntax was created for OWL, along with a mapping from abstract syntax to RDF graphs. This abstract syntax is closer to the syntax of OIL than of DAML+OIL, but without OIL's extreme emphasis on readability. In this abstract syntax the Student example above would be written

```
Class(Student complete
Person
restriction(enrolledIn minCardinality(1))).
```

OWL DL was then defined as the syntactic subset of OWL induced by the translation from the abstract syntax to RDF graphs. That is, an RDF graph is an OWL DL ontology just when it is the translation of some ontology in the abstract syntax. Users and tools that are more interested in readability than in RDF/XML can use this abstract syntax internally, or even externally for presentation to users, reserving the RDF/XML syntax for purposes of exchange between applications.

B. Handling Malformed Graphes

Because OWL Full allows arbitrary RDF graphs, it must be able to handle malformed OWL syntax. (OWL DL does not suffer from this problem as it is defined in terms of the necessarily well-formed RDF graphs that can be generated from the abstract syntax.) OWL uses an extension of the DAML+OIL solution: only triples that together form well-formed OWL constructs are given an extra meaning, so

```
:x owl:onProperty ex:child .
```

by itself does not have any special OWL meaning. To handle the cases of too many triples, OWL again uses the DAML+OIL solution of picking out all the well-formed subsets and giving them OWL meaning. This has unusual consequences—for example

```
_:x owl:onProperty ex:child .
_:x owl:someValuesFrom ex:Person .
_:x owl:onProperty ex:friend .
_:x owl:allValuesFrom ex:Doctor .
```

ends up equating the extension of four different OWL restrictions (all possible combinations of the two properties with the two classes), which is almost certainly not what was intended by the user. This solution, however, maintains monotonicity, and the (possibly) non-intuitive meaning is a minor problem given that such malformed constructions can easily be avoided.

VII. OWL

This section describes how the solutions outlined above have been incorporated in the final design of the OWL language. For various reasons, described in the preceding sections, there are two styles of using OWL. In the first style, embodied in OWL DL and OWL Lite, only certain constructions are allowed, and these constructions can only be combined in certain ways. The benefits of staying within these limitations include decidability of inferences and the possibility of thinking of OWL in a more-standard fashion, essentially as an expressive Description Logic. In the second style, embodied in OWL Full, all RDF graphs are allowed. The benefits of this expansive style include total upward compatibility with RDF and a greater expressive power. Even the more-limited versions of OWL have some differences from

standard Description Logics. These differences move these versions of OWL from the formal Description Logic world to the Semantic Web world.

- OWL uses URI references as names, and constructs these URI references in OWL to use qualified names as shorthands for URI references, using, for example, the qualified name owl:Thing for the URI reference http://www.w3.org/2002/07/owl#Thing.
- OWL gathers information into ontologies, which are generally stored as Web documents written in RDF/XML. Ontologies can import other ontologies, adding the information from the imported ontology to the current ontology.
- Even the DL/Lite style of using OWL allows RDF annotation properties to be used to attach information to classes, properties, and ontologies, such as owl:DeprecatedClass. These annotations are RDF triples, and are therefore required to carry a full semantic weight. They cannot be treated as informal comments without a formal meaning. This partly breaks down the firm Description Logic distinction between individuals, on the one hand, and classes and properties, on the other.
- OWL uses the facilities of RDF datatypes and XML Schema datatypes to provide datatypes and data values.

– The DL and Lite versions of OWL have a frame-like abstract syntax, whereas RDF/XML is the official exchange syntax for all of OWL.

A. OWL as Description Logic

OWL DL—the Description Logic style of using OWL—is very close to the SHOIN(D) Description Logic which is itself an extension of the the influential SHOQ(D) Description Logic [23] (extended with inverse roles and restricted to unqualified number restrictions). OWL DL can form descriptions of classes, datatypes, individuals and data values. OWL DL uses these description-forming constructs in axioms that provide information about classes, properties, and individuals.

B. Semantics for OWL DL

A formal semantics, very similar to the semantics provided for Description Logics, is provided for this style of using OWL. Full details on this model theory can be found in the OWL Semantics and Abstract Syntax [6].

Because OWL includes datatypes, the semantics for OWL is very similar to that of Description Logics that also incorporate datatypes, in particular SHOQ(D). However, the particular datatypes used in OWL are taken from RDF and XML Schema Datatypes [19]. Data values such as "44" ^^ xsd:integer thus mean what they would mean as XML Schema data values.

The semantics for OWL DL does include some unusual (for Description Logics) aspects. Annotations are given a simple separate meaning, not shown here, that can be used to associate information with classes, properties, and individuals in a manner compatible with the RDF semantics. Ontologies also live within the semantics and can be given annotation information. Finally, owl:imports is given a meaning that involves finding the referenced ontology (if possible) and adding its meaning to the meaning of the current ontology. What makes OWL DL a Semantic Web language, therefore, is not its semantics, which are quite standard for a Description Logic, bu t instead the use

of URI references for names, the use of XML Schema datatypes for data values, and the ability to connect to documents in the World Wide Web.

C. OWL Lite

OWL DL is related to SHOIN(D), a very expressive Description Logic. This Description Logic is somewhat difficult to present to naive users, as it is possible to build complex boolean descriptions using, for example, union and complement. SHOIN(D) is also difficult to reason with, as key inference problems have NExpTime complexity, and somewhat difficult to build even non-reasoning tools for, because of the complex descriptions. For these reasons, a subset of OWL DL has been identified that should be easier on all the above metrics; this subset is called OWL Lite. OWL Lite prohibits unions and complements, restricts intersections to the implicit intersections in the frame-like class axioms, limits all embedded descriptions to concept names, does not allow individuals to show up in descriptions or class axioms, and limits cardinalities to 0 or 1. These restrictions make OWL Lite similar to the Description Logic SHIF(D). Like SHIF(D), key inferences in OWL Lite can be computed in worst case exponential time (ExpTime), and there are already several optimized reasoners for logics equivalent to OWL Lite. This improvement in tractability comes with relatively little loss in expressive power—although OWL Lite syntax is more restricted than that of OWL DL it is still possible to express complex descriptions by introducing new class names and exploiting the implicit negations introduced by disjointness axioms. Using these techniques, all OWL DL descriptions can be captured in OWL Lite except those containing either individual names or cardinalities greater than 1.

D. OWL Full as RDF Extension

OWL DL and OWL Lite are extensions of a restricted use of RDF and RDFS, because, unlike RDF and RDFS, they do not allow classes to be used as individuals, and he language constructors cannot be applied to the language itself. For users who need these capabilities, a version of OWL that is upward compatible with RDF and RDFS has been provided; this version is called OWL Full. In OWL Full, all RDF and RDFS combinations are allowed. For example, in OWL Full, it is possible to impose a cardinality constraint on rdfs:subClassOf, if so desired. OWL Full contains OWL DL, but goes well outside the standard Description Logic framework. The penalty to be paid here is two-fold. First, reasoning in OWL Full is undecidable (because restrictions required in order to maintain the decidability of OWL DL do not apply to OWL full). Second, the abstract syntax for OWL DL is inadequate for OWL Full, and the official OWL exchange syntax, RDF/XML, must be used.

E. OWL 2 and OWL 1

Like OWL 1, OWL 2 is designed to facilitate ontology development and sharing via the Web, with the ultimate goal of making Web content more accessible to machines. The RDF-Based Semantics assigns meaning directly to RDF graphs and so indirectly to ontology structures via the Mapping to RDF graphs. The RDF-Based Semantics is fully compatible with the RDF Semantics, and extends the semantic conditions defined for RDF. The RDF-Based Semantics can be applied to any OWL 2 Ontology, without restrictions, as any OWL 2 Ontology can be mapped to RDF. "OWL 2 Full" is used informally to refer to RDF graphs considered as OWL 2 ontologies and interpreted using the RDF-Based Semantics. OWL 2 has a very similar overall structure to OWL 1.Almost all the building blocks of OWL 2 are present in OWL 1, albeit possibly under different names.

The central roles of RDF/XML, the role of other syntaxes, and the relationships between the Direct and RDF-Based semantics (i.e., the correspondence theorem) have not changed. More importantly, backwards compatibility with OWL 1 is maintained and all OWL 1 Ontologies remain valid OWL 2 Ontologies, with identical inferences in all practical cases OWL 2 adds new functionality with respect to OWL 1. Some of the new features are syntactic uses (e.g., disjoint union of classes) while others offer new expressivity, including:

- keys;
- property chains;
- richer datatypes, data ranges;
- qualified cardinality restrictions;
- asymmetric, reflexive, and disjoint properties; and
- enhanced annotation capabilities

OWL 2 also defines three new profiles and a new syntax In addition, some of the restrictions applicable to OWL DL have been relaxed; as a result, the set of RDF Graphs that can be handled by Description Logics reasoners is slightly larger in OWL 2.

An OWL 2 profile (commonly called a fragment or a sublanguage in computational logic) is a trimmed down version of OWL 2 that trades some expressive power for the efficiency of reasoning. This document describes three profiles of OWL 2, each of which achieves efficiency in a different way and is useful in different application scenarios. The profiles are independent of each otherand the choice of which profile to use in practice will depend on the structure of the ontologies and the reasoning tasks at hand

- OWL 2 EL is particularly useful in applications employing ontologies that contain very large numbers of properties and/or classes. This profile captures the expressive power used by much such ontology and is a subset of OWL 2 for which the basic reasoning problems can be performed in time that is polynomial with respect to the size of the ontology. Dedicated reasoning algorithms for this profile are available and have been demonstrated to be implementable in a highly scalable way. The EL acronym reflects the profile's basis in the EL family of description logics , logics that provide only Existential quantification.

- OWL 2 QL is aimed at applications that use very large volumes of instance data, and where query answering is the most important reasoning task. In OWL 2 QL, conjunctive query answering can be implemented using conventional relational database systems. Using a suitable reasoning technique, sound and complete conjunctive query answering can be performed in LOGSPACE with respect to the size of the data (assertions). As in OWL 2 EL, polynomial time algorithms can be used to implement the ontology consistency and class expression subsumption reasoning problems. The expressive power of the profile is necessarily quite limited, although it does include most of the main features of conceptual models such as UML class diagrams and ER diagrams. The QL acronym reflects the fact that query answering in this profile can be implemented by rewriting queries into a standard relational Query Language.

- OWL 2 RL is aimed at applications that require scalable reasoning without sacrificing too much expressive power. It is designed to accommodate OWL 2 applications that can trade the full expressivity of the language for efficiency, as well as RDF(S) applications that need some added expressivity. OWL 2 RL reasoning systems can be implemented using rule-based reasoning engines. The ontology consistency, class expression satisfiability, class expression subsumption, instance checking, and conjunctive query answering problems can be solved in time that is polynomial with respect to the size of the ontology. The RL acronym reflects the fact that reasoning in this profile can be implemented using a standard Rule Language.

OWL 2 profiles are defined by placing restrictions on the structure of OWL 2 ontologies. Syntactic restrictions can be specified by modifying the grammar of the functional-style syntax and possibly giving additional global restrictions. An ontology in any profile can be written into an ontology document by using any of the syntaxes of OWL 2.

VIII. Ontology engineering and Related tools

Ontology engineering is by now an established engineering discipline, providing the full range of methodologies, methods, techniques, and software tools that allow for real-world projects to be feasibly undertaken. Ontology engineering methodologies provide guidelines for developing, managing and maintaining ontologies; recent surveys on ontology engineering methodologies are available in [39]. Classical ontology engineering is moving towards collaborative approaches based on wikis [40] tagging [41] or casual games [42)]. Methodologies for ontology reuse [Gangemi et al. (1998); Paslaru-Bontas (2007); Pinto et al. (2000)] or complement the overall picture, guiding the ontology support activities of the ontology life cycle.

The target client for ontologies build using OWL are semantic web based applications.. For the semantic web to become a reality, a number of frameworks have to be built to support the ontology creation activities involved in the process. These activities, as we envision this process, are as follows:

Gathering: Before the extraction phase, we have to collect documents carrying knowledge from the domain we are interested in, process them, and end with a suitable form to carry out the next operations. It usually involves dealing with unstructured data in natural language from digital archives [24, 25]. Some useful software tools to carry out gathering tasks are: Spade8 and OntoExtract9.

Extraction:The process of extraction is based on ontology learing .A methodology for ontology learing is discussed in [43].A number of ontology learning tools are available. The purpose of this kind of software is to help the ontology engineer to explore specific domains and extract ontology components. This requires background knowledge for creating taxonomies of the domain in a semi-automatic way. Learning techniques may be applied by the knowledge engineer for this task [26–30]. Some useful extraction software tools are: Grubber10 and Onto- Builder11.

Organization: Once the ontology components have been extracted from the domain, it is time to generate formal representations of the knowledge acquired. Ontology software tools may be useful at this stage. Later, this knowledge may be embedded into digital archives, e.g., web pages, to be used by software agents or humans [31-33]. Some useful ontology software tools are: OntoEdit12, SMORE13, and Protégé14.

Merging: Defining mapping rules to facilitate interlingua exchange relating information from one context to another. This activity is as important as extraction. It can be referred to as finding commonalities between two knowledge bases and deriving a new knowledge base [34-35]. Some helpful software tools for merging ontologies are: PROMPT15, and quimaera16.

Refinement: Improving the structure and content of the knowledge by eliciting knowledge from the domain experts. It amends the knowledge at a finer granularity level. It is also of particular importance after merging operations ([35-37]), for instance, when two e-commerce agents are trying to negotiate. A number of software tools for organizing ontology components include refinement capabilities as well.

Retrieval: This is the ultimate semantic web goal and it is going to take a while yet before we see smart software applications, but when the semantic web is populated, then those applications, e.g., semantic robots, agents, will traverse the web looking for data for us in a knowledge-based fashion. In the mean while, we still have to wait for those frameworks to mature. Racer17, and KAON218 are some promising early tools to carry out these tasks.

Practical guidelines and recommendations for developing ontology-based applications in specific sectors are available, for instance, in [44]; another study reported on social and technical bottlenecks which hinder the wide uptake of ontologies and one of its main finding was the need for advanced technology to cope with ontology development and maintenance especially in rapidly changing domains [45].

IX. Summary

Because of the ambitious design goals, multiple influences, and also because of the structural requirements constraining OWL, the development of OWL has not been without problems. Through hard work and compromise, these problems have largely been overcome, resulting in a ontology language that is truly part of the Semantic Web. It was not possible to simultaneously satisfy all of the constraints on OWL, so two styles of using OWL have been developed, each suitable under different circumstances.

If an expressive ontology language with decidable inference is the main concern, then the OWL DL style is indicated. This style of using OWL loses some compatibility with RDF, mostly having to do with using classes and properties as individuals, but retains an expressive and useful ontology language. OWL DL also has a frame-like alternative syntax that can be used to make working with OWL easier.. If a simpler ontology language is the main concern, then the OWL Lite subset of OWL DL can be used.

If, on the other hand, upward compatibility with RDF is the main concern, then the OWL Full style is indicated. This style extends RDF and RDFS to a full ontology language with a well-specified entailment relationship that extends entailment in RDF and RDFS, while avoiding any paradoxes that might arise. However, entailment in OWL Full is non decidable, which can be a significant issue in some circumstances. Also, the user-friendly alternative syntax is not adequate for OWL Full, so RDF/XML must be used for OWL Full. These styles of using OWL provide an ontology layer for the Semantic Web, significantly extending the capabilities of RDF and RDFS, and expanding the usefulness of the Semantic Web.

OWL is primarily designed to be used for semantic web applications and thus a large number of tools were developed that facilitate the ontology development process and subsequently use of ontology in semantic web applications.

REFERENCES

[1] Jie Bao, Elisa F. Kendall, Deborah L. McGuinness, Peter F. Patel-Schneider, eds. "OWL 2 Web Ontology Language: Quick Reference Guide W3C Working Draf", 11 June 2009, http://www.w3.org/TR/2009/WD-owl2-quick-reference-20090611/. Latest version available at http://www.w3.org/TR/owl2-quick-reference/ .

[2] Tim Berners-Lee. "Weaving the Web". Harper, San Francisco, 1999.

[3] Frank Manola and Eric Miller. "RDF primer." W3C Working Draft, 23 January 2003.

[4] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. "Practical reasoning for expressive description logics", Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99), number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer, 1999.

[5] Michael K. Smith, Chris Welty, and Deborah McGuinness. "OWL web ontology language guide". W3C Working Draft, 31 March 2003.

[6] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. "OWL web ontology language semantics and abstract syntax" W3C Working Draft, 31 March 2003.

[7] Jeff Heflin. "Web ontology language (owl) use cases and requirements", W3C Working Draft, 31 March 2003.

[8] Dave Beckett. "RDF/XML syntax specification (revised)", W3C Working Draft, 2002.

[9] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. "The Description Logic Handbook". Cambridge University Press, 2007.

[10] J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. "Enabling knowledge representation on the web by extending RDF schema"., Proceedings of the tenth World Wide Web conference WWW'10, pages 467–478, May 2001.

[11] Patrick Hayes. "RDF semantics". W3C Working Draft, 2003. Available at http://www.w3.org/TR/2003/WD-rdf-mt-20030123.

[12] Li Ma et al "Towards a Complete OWL Ontology Benchmark", The Semantic Web: Research and Applications pages 125-139, June 2006.

[13] Natalya Fridman Noy, Ray W. Fergerson and Mark A. Musen " The Knowledge Model of Protégé-2000: Combining Interoperability and Flexibility", Knowledge Engineering and Knowledge Management Methods, Models, and Tools ,pages 69-82 ,June 2000

[14] Peter D. Karp, Vinay K. Chaudhri, and Jerome Thomere. XOL: "An XML-based ontology exchange language" Technical Report SRI AI Technical Note 559, SRI International, Menlo Park (CA, USA), 1999.

[15] I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. "OIL: The Ontology Inference Layer" Technical Report IR-479, Vrije Universiteit Amsterdam, Faculty of Sciences, September 2000.

[16] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. "DAML+OIL (March 2001) reference description". W3C Note, 18 December 2001. Available at http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218.

[17] Graham Klyne and Jeremy J. Carroll. "Resource Description Framework (RDF): Concepts and abstract syntax". W3C Working Draft, 2003. Available at http://www.w3.org/TR/2003/WD-rdf-concepts-20030123.

[18] Frank van Harmelen, Ian Horrocks, and Peter F. Patel-Schneider. "A model theoretic semantics for DAML+OIL "(March 2001). W3C Note, 18 December 2001. Available at http://www.w3.org/TR/2001/NOTE-daml+oil-model-20011218.

[19] Paul V. Biron and Ashok Malhotra. "XML Schema Part 2: Datatypes. W3C Recommendation, 2001". Available at http://www.w3.org/TR/2002/WD-xmlschema-

[20] J. Heffin, J. Hendler, "Dynamic ontologies on the web", American Association For Artificial Intelligence Conference, AAAI Press, California, 2000, pp. 251–254.

[21] J. Heffin, et al., "Applying ontology to the web: a case study", Engineering Applications of Bio-Inspired Artificial Neural Networks

[22] I. Horrocks, "DAML+ OIL: a reasonable web ontology language", Lecture Notes in Computer Science (LNCS), Vol. 2287, Springer-Verlag, Berlin, 2002, pp. 2–13.

[23] Ian Horrocks and Ulrike Sattler," Ontology reasoning in the SHOQ(D) description logic".Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001), pages 199–204, 2001.

[24] D. Elliman, JRG Pulido, "Automatic derivation of on-line document ontologies", presented at the Int. Workshop on Mechanisms for Enterprise Integration: From Objects to Ontologies (MERIT 2001), the 15th European Conference on Object Oriented Programming, Budapest, Hungary June, 2001.

[25] JRG. Pulido, et al., "Identifying ontology components from digital archives on the semantic web", IASTED Advances in Computer Science and Technology (ACST), 2006.

[26] T. Quan, et al., "Automatic generation of ontology of scholarly semantic web, in: S. McIlraith (Ed.), The Semantic Web ISWC 2004: Third International Semantic Web Conference, Lecture Notes in Computer Science (LNCS), Vol. 3298, Springer, 2004, pp. 726–740.

[27] S. Legrand, JRG Pulido, "A hybrid approach to word sense disambiguation: Neural clustering with class labeling", Workshop on Knowledge Discovery and Ontologies, 15th European Conference on Machine Learning (ECML) and 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), Pisa, Italy, 2004, pp. 127.

[28] M. Ehrig, S. Staab, QOM—quick ontology mapping, in: S. McIlraith (Ed.), The Semantic Web—ISWC 2004: Third International Semantic Web Conference, Lecture Notes in Computer Science (LNCS), Vol. 3298, Springer, 2004, pp. 683–697.

[29] A. Maedche, V. Zacharias, "Clustering ontology-based metadata in the semantic web", T. Elomaa (Ed.), Principles of Data Mining and Knowledge Discovery: Sixth European Conference, PKDD 2002, Lecture Notes in Computer Science (LNCS), Vol. 2431, Springer, 2002, pp. 348–360.

[30] A. Maedche, S. Staab, "Ontology learning for the semantic web", IEEE Intell. Syst. 16 (2) (2001) 72–79.

[31] M. Vargas, et al., "MnM: Ontolgy driven semi-automatic and automatic support for semantic markup", A. Gómez, V. Benjamins (Eds.), Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web: 13th International Conference, EKAW 2002, Lecture Notes in Computer Science (LNCS), Vol. 2473, Springer, 2002, pp. 379–391.

[32] J. Goldbeck, et al.,"New tools for the semantic web", iA. Gómez, V. Benjamins (Eds.), Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web: 13th International Conference, EKAW 2002, Lecture Notes in Computer Science (LNCS), Vol. 2473, Springer, 2002, pp. 392–400.
[33] J. Fernandez, R. Martinez, "A cooperative tool for facilitating knowledge management", Expert Syst. Appl. 18 (2000) 315–330.
[34] J. Euzenat, "An API for ontology alignment", S. McIlraith (Ed.), The Semantic Web ISWC 2004: Third International Semantic Web Conference, Lecture Notes in Computer Science (LNCS), Vol. 3298, Springer, 2004, pp. 698–712.
[35]] R. Eijk, et al., "On dynamically generated ontology translators in agent communication", Int. J. Intell. Syst. 16 (2001) 587–607.
[36] N. Sugiura, et al.,"Towards on-the-Xy ontology-constructionfocusing on ontology quality improvement", C. Bussler, et al. (Eds.), The Semantic Web: Research and Applications: First European Semantic Web Symposium, ESWS 2004, Lecture Notes in Computer Science (LNCS), Vol. 3053, Springer, 2004.
[37] A. Maedche, S. Staab, "Measuring similarity between ontologies", A. Gomez, R. Benjamins (Eds.), Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web: 13th International Conference, EKAW 2002, Lecture Notes in Computer Science (LNCS), Vol. 247, pp. 251–263, Springer, 2002.
[38] M. Hatala, G. Richards, Value-added metatagging: Ontology and rule based methods for smarter metadata, in: M. Schroeder, G. Wagner (Eds.), Rules and Rule Markup Languages for the Semantic Web: Second International Workshop, RuleML 2003, Lecture Notes in Computer Science (LNCS), Vol. 2876, pp. 65–80, Springer, 2003.
[39] Sure, Y., Tempich, C., Vrandecic, D.. "Ontology Engineering Methodologies". Semantic Web technologies: Trends and Research in Ontology-based Systems, 171-87, Wiley 2006.
[40] Tempich, C., Simperl, E., Pinto, S., Luczak, M., Studer, R. "Argumentation-based Ontology Engineering", IEEE Intelligent Systems, 22(6):52-59, 2007.
[41] Braun, S., Schmidt, A., Walter, A., Nagyp, G., Zacharias, V. "Ontology Maturing: a Collaborative Web 2.0 Approach to Ontology Engineering,", Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge (CKC 2007) at the 16th International World Wide Web Conference (WWW2007),2007.
[42] Siorpaes, K., Hepp, M. , "Games with a purpose for the semantic web". IEEE Intelligent Systems, 23(3):50-60, 2008.
[43] Simperl, E., Tempich, C. " A Methodology for Ontology Learning", P. Buitelaar and P. Cimiano (editors) Bridging the Gap between Text and Knowledge – Selected Contributions to Ontology Learning and Population from Text, IOS Press 2007.
[44] Mochol, M., Simperl, E " Practical guidelines for building semantic eRecruitment applications" Proceedings of International Conference on Knowledge Management (iKnow'06), Special Track: Advanced Semantic Technologies (AST2006).
[45] Hepp M. " Possible ontologies: How reality constrains the development of relevant ontologies", IEEE Internet Computing, 11(1):90-96. 2007.

Call for Papers and Special Issues

Aims and Scope

Journal of Emerging Technologies in Web Intelligence (JETWI, ISSN 1798-0461) is a peer reviewed and indexed international journal, aims at gathering the latest advances of various topics in web intelligence and reporting how organizations can gain competitive advantages by applying the different emergent techniques in the real-world scenarios. Papers and studies which couple the intelligence techniques and theories with specific web technology problems are mainly targeted. Survey and tutorial articles that emphasize the research and application of web intelligence in a particular domain are also welcomed. These areas include, but are not limited to, the following:

- Web 3.0
- Enterprise Mashup
- Ambient Intelligence (AmI)
- Situational Applications
- Emerging Web-based Systems
- Ambient Awareness
- Ambient and Ubiquitous Learning
- Ambient Assisted Living
- Telepresence
- Lifelong Integrated Learning
- Smart Environments
- Web 2.0 and Social intelligence
- Context Aware Ubiquitous Computing
- Intelligent Brokers and Mediators
- Web Mining and Farming
- Wisdom Web
- Web Security
- Web Information Filtering and Access Control Models
- Web Services and Semantic Web
- Human-Web Interaction
- Web Technologies and Protocols
- Web Agents and Agent-based Systems
- Agent Self-organization, Learning, and Adaptation
- Agent-based Knowledge Discovery
- Agent-mediated Markets
- Knowledge Grid and Grid intelligence
- Knowledge Management, Networks, and Communities
- Agent Infrastructure and Architecture
- Agent-mediated Markets
- Cooperative Problem Solving
- Distributed Intelligence and Emergent Behavior
- Information Ecology
- Mediators and Middlewares
- Granular Computing for the Web
- Ontology Engineering
- Personalization Techniques
- Semantic Web
- Web based Support Systems
- Web based Information Retrieval Support Systems
- Web Services, Services Discovery & Composition
- Ubiquitous Imaging and Multimedia
- Wearable, Wireless and Mobile e-interfacing
- E-Applications
- Cloud Computing
- Web-Oriented Architectrues

Special Issue Guidelines

Special issues feature specifically aimed and targeted topics of interest contributed by authors responding to a particular Call for Papers or by invitation, edited by guest editor(s). We encourage you to submit proposals for creating special issues in areas that are of interest to the Journal. Preference will be given to proposals that cover some unique aspect of the technology and ones that include subjects that are timely and useful to the readers of the Journal. A Special Issue is typically made of 10 to 15 papers, with each paper 8 to 12 pages of length.

The following information should be included as part of the proposal:

- Proposed title for the Special Issue
- Description of the topic area to be focused upon and justification
- Review process for the selection and rejection of papers.
- Name, contact, position, affiliation, and biography of the Guest Editor(s)
- List of potential reviewers
- Potential authors to the issue
- Tentative time-table for the call for papers and reviews

If a proposal is accepted, the guest editor will be responsible for:

- Preparing the “Call for Papers” to be included on the Journal’s Web site.
- Distribution of the Call for Papers broadly to various mailing lists and sites.
- Getting submissions, arranging review process, making decisions, and carrying out all correspondence with the authors. Authors should be informed the Instructions for Authors.
- Providing us the completed and approved final versions of the papers formatted in the Journal’s style, together with all authors’ contact information.
- Writing a one- or two-page introductory editorial to be published in the Special Issue.

Special Issue for a Conference/Workshop

A special issue for a Conference/Workshop is usually released in association with the committee members of the Conference/Workshop like general chairs and/or program chairs who are appointed as the Guest Editors of the Special Issue. Special Issue for a Conference/Workshop is typically made of 10 to 15 papers, with each paper 8 to 12 pages of length.

Guest Editors are involved in the following steps in guest-editing a Special Issue based on a Conference/Workshop:

- Selecting a Title for the Special Issue, e.g. “Special Issue: Selected Best Papers of XYZ Conference”.
- Sending us a formal “Letter of Intent” for the Special Issue.
- Creating a “Call for Papers” for the Special Issue, posting it on the conference web site, and publicizing it to the conference attendees. Information about the Journal and Academy Publisher can be included in the Call for Papers.
- Establishing criteria for paper selection/rejections. The papers can be nominated based on multiple criteria, e.g. rank in review process plus the evaluation from the Session Chairs and the feedback from the Conference attendees.
- Selecting and inviting submissions, arranging review process, making decisions, and carrying out all correspondence with the authors. Authors should be informed the Author Instructions. Usually, the Proceedings manuscripts should be expanded and enhanced.
- Providing us the completed and approved final versions of the papers formatted in the Journal’s style, together with all authors’ contact information.
- Writing a one- or two-page introductory editorial to be published in the Special Issue.

More information is available on the web site at http://www.academypublisher.com/jetwi/.