OWL Web Ontology Language Use Cases and Requirements

W3C Proposed Recommendation 15 December 2003

This version:
http://www.w3.org/TR/2003/PR-webont-req-20031215/
Latest version:
http://www.w3.org/TR/webont-req/
Previous version:
http://www.w3.org/TR/2003/CR-webont-req-20030818/
Editor:
Jeff Heflin (Lehigh University) heflin@cse.lehigh.edu

Abstract

This document specifies usage scenarios, goals and requirements for a web ontology language. An ontology formally defines a common set of terms that are used to describe and represent a domain. Ontologies can be used by automated tools to power advanced services such as more accurate Web search, intelligent software agents and knowledge management.

Status of this document

This section describes the status of this document at the time of its publication. Other documents may supersede this document. A list of current W3C publications and the latest revision of this technical report can be found in the W3C technical reports index at http://www.w3.org/TR/.

Publication as a Proposed Recommendation does not imply endorsement by the W3C Membership. This is a draft document and may be updated, replaced or obsoleted by other documents at any time. It is inappropriate to cite this document as other than work in progress.

This draft is one of six parts of the Proposed Recommendation (PR) for OWL, the Web Ontology Language. It has been developed by the Web Ontology

Working Group as part of the W3C Semantic Web Activity (Activity Statement, Group Charter) for publication on 15 December 2003.

The design of OWL expressed in earlier versions of these documents has been widely reviewed and satisfies the Working Group's technical requirements. The Working Group has addressed all comments received, making changes as necessary. During Candidate Recommendation, many implementations were reported, covering among them all features of the language. Changes to this document since the Candidate Recommendation version are detailed in the change log.

W3C Advisory Committee Representatives are now invited to submit their formal review via Web form, as described in the Call for Review. Additional comments may be sent to a Team-only list, w3t-semweb-review@w3.org. The public is invited to send comments to public-webont-comments@w3.org (archive) and to participate in general discussion of related technology at www-rdf-logic@w3.org (archive). The review period extends until **19 January 2004**.

The W3C maintains a list of any patent disclosures related to this work.

Table of contents

1 Introduction

The Semantic Web is a vision for the future of the Web in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web. The Semantic Web will build on XML's ability to define customized tagging schemes *[XML]* and RDF's flexible approach to representing data *[RDF Concepts]*. The next element required for the Semantic Web is a Web ontology language which can formally describe the semantics of classes and properties used in web documents. In order for machines to perform useful reasoning tasks on these documents, the language must go beyond the basic semantics of RDF Schema *[RDF Schema]*.

This document is one part of the specification of OWL, the Web Ontology Language. The Document Roadmap section of the OWL Overview document describes each of the other documents. This document enumerates the requirements of a web ontology language as perceived by the working group. However, it is expected that future languages will extend OWL, adding, among other things, greater logical capabilities and the ability to establish trust on the Semantic Web.

We motivate the need for a Web ontology language by describing six use cases. Some of these use cases are based on efforts currently underway in industry and academia, others demonstrate more long-term possibilities. The use cases are followed by design goals that describe high-level objectives and guidelines for the development of the language. These design goals will be considered when evaluating proposed features. The section on Requirements presents a set of features that should be in the language and gives motivations for those features. The Objectives section describes a list of features that might be useful for many use cases but may not necessarily be addressed by the working group.

The Web Ontology Working Group charter tasks the group to produce this more expressive semantics and to specify mechanisms by which the language can provide "more complex relationships between entities including: means to limit the properties of classes with respect to number and type, means to infer that items with various properties are members of a particular class, a well-defined model of property inheritance, and similar semantic extensions to the base languages." The detailed specification of the Web Ontology language will take into consideration:

- the design goals and requirements that are contained in this document
- review comments on this document from public feedback, invited experts and working group members
- specifications of or proposals for languages that meet many of these requirements

1.1 What is an ontology?

An ontology defines the terms used to describe and represent an area of knowledge. Ontologies are used by people, databases, and applications that

need to share domain information (a domain is just a specific subject area or area of knowledge, like medicine, tool manufacturing, real estate, automobile repair, financial management, etc.). Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them (note that here and throughout this document, definition is not used in the technical sense understood by logicians). They encode knowledge in a domain and also knowledge that spans domains. In this way, they make that knowledge reusable.

The word ontology has been used to describe artifacts with different degrees of structure. These range from simple taxonomies (such as the Yahoo hierarchy), to metadata schemes (such as the Dublin Core), to logical theories. The Semantic Web needs ontologies with a significant degree of structure. These need to specify descriptions for the following kinds of concepts:

- Classes (general things) in the many domains of interest
- The relationships that can exist among things
- The properties (or attributes) those things may have

Ontologies are usually expressed in a logic-based language, so that detailed, accurate, consistent, sound, and meaningful distinctions can be made among the classes, properties, and relations. Some ontology tools can perform automated reasoning using the ontologies, and thus provide advanced services to intelligent applications such as: conceptual/semantic search and retrieval, software agents, decision support, speech and natural language understanding, knowledge management, intelligent databases, and electronic commerce.

Ontologies figure prominently in the emerging Semantic Web as a way of representing the semantics of documents and enabling the semantics to be used by web applications and intelligent agents. Ontologies can prove very useful for a community as a way of structuring and defining the meaning of the metadata terms that are currently being collected and standardized. Using ontologies, tomorrow's applications can be "intelligent," in the sense that they can more accurately work at the human conceptual level.

Ontologies are critical for applications that want to search across or merge information from diverse communities. Although XML DTDs and XML Schemas are sufficient for exchanging data between parties who have agreed to definitions beforehand, their lack of semantics prevent machines from reliably performing this task given new XML vocabularies. The same term may be used with (sometimes subtle) different meaning in different contexts, and different terms may be used for items that have the same meaning. RDF and RDF Schema begin to approach this problem by allowing simple semantics to be associated with identifiers. With RDF Schema, one can define classes that may have multiple subclasses and super classes, and can define properties, which may have sub properties, domains, and ranges. In this sense, RDF Schema is a simple ontology language. However, in order to achieve interoperation between numerous, autonomously developed and managed schemas, richer semantics are needed. For example, RDF Schema cannot specify that the

Person and Car classes are disjoint, or that a string quartet has exactly four musicians as members.

One of the goals of this document is to specify what is needed in a Web Ontology language. These requirements will be motivated by potential use cases and general design objectives that take into account the difficulties in applying the standard notion of ontologies to the unique environment of the Web.

2 Use cases

Ontologies can be used to improve existing Web-based applications and may enable new uses of the Web. In this section we describe six representative use cases of Web ontologies. Note that this is not an exhaustive list, but instead a cross-section of interesting use cases. These use cases served as a guideline in choosing requirements for the OWL language (see Section 4). The requirements were chosen based on the aspects of the use cases that the working group considered most important, while considering the scope of the OWL charter and other design constraints. As such, one should not assume that OWL will directly support every aspect of the use cases.

2.1 Web portals

A Web portal is a web site that provides information content on a common topic, for example a specific city or domain of interest. A web portal allows individuals that are interested in the topic to receive news, find and talk to one another, build a community, and find links to other web resources of common interest.

In order for a portal to be successful, it must be a starting place for locating interesting content. Typically this content is submitted by members of the community, who often index it under some subtopic. Another means of collecting content relies on the content providers tagging the content with information that can be used in syndicating it. Typically, this takes the form of simple metatags that identify the topic of the content, etc.

However, a simple index of subject areas may not provide the community with sufficient ability to search for the content that its members require. In order to allow more intelligent syndication, web portals can define an ontology for the community. This ontology can provide a terminology for describing content and axioms that define terms using other terms from the ontology. For example, an ontology might include terminology such as "journal paper," "publication," "person," and "author." This ontology could include definitions that state things such as "all journal papers are publications" or "the authors of all publications are people." When combined with facts, these definitions allow other facts that are necessarily true to be inferred. These inferences can, in turn, allow users to obtain search results from the portal that are impossible to obtain from conventional retrieval systems. Such a technique relies on content providers using the Web ontology language to capture high-quality ontology relationships, and an objective of OWL is to enable sufficiently rich and useful

metadata content to motivate the necessary effort. It is a separate challenge to minimize this effort and an ontology language will likely have a greater impact if it can facilitate metadata capture as an nonintrusive part of any information creation process.

One example of an ontology based portal is OntoWeb. This portal serves the academic and industry community that is interested in ontology research. Another example of a portal that uses Semantic Web technologies and could benefit from an ontology language is The Open Directory Project; a large, comprehensive human-edited directory of the Web. It is constructed and maintained by a vast, global community of volunteer editors. RDF dumps of the Open Directory database are available for download.

2.2 Multimedia collections

Ontologies can be used to provide semantic annotations for collections of images, audio, or other non-textual objects. It is even more difficult for machines to extract meaningful semantics from multimedia than it is to extract semantics from natural language text. Thus, these types of resources are typically indexed by captions or metatags. However, since different people can describe these non-textual objects in different ways, it is important that the search facilities go beyond simple keyword matching. Ideally, the ontologies would capture additional knowledge about the domain that can be used to improve retrieval of images.

Multimedia ontologies can be of two types: media-specific and content-specific. Media specific ontologies could have taxonomies of different media types and describe properties of different media. For example, video may include properties to identify length of the clip and scene breaks. Content-specific ontologies could describe the subject of the resource, such as the setting or participants. Since such ontologies are not specific to the media, they could be reused by other documents that deal with the same domain. Such reuse would enhance search that was simply looking for information on a particular subject, regardless of the format of the resource. Searches where media type was important could combine the media-specific and content-specific ontologies.

As an example of a multimedia collection, consider an archive of images of antique furniture. An ontology of antique furniture would be of great use in searching such an archive. A taxonomy can be used to classify the different types of furniture. It would also be useful if the ontology could express definitional knowledge. For example, if an indexer selects the value "Late Georgian" for the style/period of (say) an antique chest of drawers, it should be possible to infer that the data element "date.created" should have a value between 1760 and 1811 A.D. and that the "culture" is British. Availability of this type of background knowledge significantly increases the support that can be given for indexing as well as for search. Another feature that could be useful is support for the representation of default knowledge. An example of such knowledge would be that a "Late Georgian chest of drawers," in the absence of other information, would be assumed to be made of mahogany. This knowledge is crucial for real semantic queries, e.g. a user query for "antique

mahogany storage furniture" could match with images of Late Georgian chests of drawers, even if nothing is said about wood type in the image annotation.

2.3 Corporate web site management

Large corporations typically have numerous web pages concerning things like press releases, product offerings and case studies, corporate procedures, internal product briefings and comparisons, white papers, and process descriptions. Ontologies can be used to index these documents and provide better means of retrieval. Although many large organizations have a taxonomy for organizing their information, this is often insufficient. A single ontology is often limiting because the constituent categories are likely constrained to those representing one view and one granularity of a domain; the ability to simultaneously work with multiple ontologies would increase the richness of description. Furthermore, the ability to search on values for different parameters is often more useful than a keyword search with taxonomies.

An ontology-enabled web site may be used by:

- A salesperson looking for sales collateral relevant to a sales pursuit.
- A technical person looking for pockets of specific technical expertise and detailed past experience.
- A project leader looking for past experience and templates to support a complex, multi-phase project, both during the proposal phase and during execution.

A typical problem for each of these types of users is that they may not share terminology with the authors of the desired content. The salesperson may not know the technical name for a desired feature or technical people in different fields might use different terms for the same concept. For such problems, it would be useful for each class of user to have different ontologies of terms, but have each ontology interrelated so translations can be performed automatically.

Another problem is framing queries at the right level of abstraction. A project leader looking for someone with expertise in operating systems should be able to locate an employee who is an expert with both Unix and Windows.

One aspect of a large service organization is that it may have a very broad set of capabilities. But when pursuing large contracts these capabilities sometimes need to be assembled in new ways. There will often be no previous single matching project. A challenge is to reason about how past templates and documents can be reassembled in new configurations, while satisfying a diverse set of preconditions.

2.4 Design documentation

This use case is for a large body of engineering documentation, such as that used by the aerospace industry. This documentation can be of several different types, including design documentation, manufacturing documentation, and

testing documentation. These document sets each have a hierarchical structure, but the structures differ between the sets. There is also a set of implied axes which cross-link the documentation sets: for example, in aerospace design documents, an item such as a wing spar might appear in each.

Ontologies can be used to build an information model which allows the exploration of the information space in terms of the items which are represented, the associations between the items, the properties of the items, and the links to documentation which describes and defines them (i.e., the external justification for the existence of the item in the model). That is to say that the ontology and taxonomy are not independent of the physical items they represent, but may be developed/explored in tandem.

A concrete example of this use case is design documentation for the aerospace domain, where typical users include:

- Maintenance engineer looking for all information relating to a particular part (e.g., "wing-spar").
- Design engineer looking at constraints on re-use of a particular sub-assembly.

To support this kind of usage, it is important that constraints can be defined. These constraints may be used to enhance search or check consistency. An example of a constraint might be:

```
biplane(X) => CardinalityOf(wing(X)) = 2
wingspar(X) AND wing(Y) AND isComponentOf(X,Y) => length(X) < length(
```

Another common use of this kind of ontology is to support the visualization and editing of charts which show snapshots of the information space centered on a particular concept (e.g., a class or instance). These are typically activity/rule diagrams or entity-relationship diagrams.

2.5 Agents and services

The Semantic Web can provide agents with the capability to understand and integrate diverse information resources. A specific example is that of a social activities planner, which can take the preferences of a user (such as what kinds of films they like, what kind of food they like to eat, etc.) and use this information to plan the user's activities for an evening. The task of planning these activities will depend upon the richness of the service environment being offered and the needs of the user. During the service determination / matching process, ratings and review services may also be consulted to find closer matches to user preferences (for example, consulting reviews and rating of films and restaurants to find the "best").

This type of agent requires domain ontologies that represent the terms for restaurants, hotels, etc. and service ontologies to represent the terms used in the actual services. These ontologies will enable the capture of information

necessary for applications to discriminate and balance among user preferences. Such information may be provided by a number of sources, such as portals, service-specific sites, reservation sites and the general Web.

Agentcities is an example of an initiative that is exploring the use of agents in a distributed service environment across the Internet. This will involve building a network of agent platforms that represent real or virtual cities, such as San Francisco or the Bay Area, and populating them with the services of those cities. Initially, these services will be oriented towards business to consumer services, such as hotels, restaurants, entertainment, etc., but eventually, they will be expanded to include business to business services, such as payroll, and business to enterprise services.

This will require a number of different domain and service ontologies: Key issues include:

- Use and integration of multiple separate ontologies across different domains and services
- Distributed location of ontologies across the Internet
- Potentially different ontologies for each domain or service (ontology translation/cross-referencing)
- Simple ontology representation to make the task of defining and using ontologies easier

2.6 Ubiquitous computing

Ubiquitous computing is an emerging paradigm of personal computing, characterized by the shift from dedicated computing machinery to pervasive computing capabilities embedded in our everyday environments. Characteristic to ubiquitous computing are small, handheld, wireless computing devices. The pervasiveness and the wireless nature of devices require network architectures to support automatic, ad hoc configuration. An additional reason for development of automatic configuration is that this technology is aimed at ordinary consumers.

A key technology of true *ad hoc* networks is service discovery, functionality by which "services" (i.e., functions offered by various devices such as cell phones, printers, sensors, etc.) can be described, advertised, and discovered by others. All of the current service discovery and capability description mechanisms (e.g., Sun's JINI, Microsoft's UPnP) are based on *ad hoc* representation schemes and rely heavily on standardization (i.e., on a priori identification of all those things one would want to communicate or discuss).

The key issue (and goal) of ubiquitous computing is "serendipitous interoperability," interoperability under "unchoreographed" conditions, i.e., devices which weren't necessarily designed to work together (such as ones built for different purposes, by different manufacturers, at a different time, etc.) should be able to discover each others' functionality and be able to take advantage of it. Being able to "understand" other devices, and reason about their services/functionality is necessary, since full-blown ubiquitous computing

scenarios will involve dozens if not hundreds of devices, and a priori standardizing the usage scenarios is an unmanageable task.

The interoperation scenarios are dynamic in nature (i.e., devices appear and disappear at any moment as their owners carry them from one room or building to another) and do not involve humans in the loop as far as (re-)configuration is concerned. The tasks involved in the utilization of services involve discovery, contracting, and composition. The contracting of services may involve representing information about security, privacy and trust, as well as about compensation-related details (the provider of a service may have to be compensated for services rendered). In particular, it is a goal that corporate or organizational security policies be expressed in application-neutral form, thus enabling constraint representation across the diversity of enforcement mechanisms (e.g., firewalls, filters/scanners, traffic monitors, application-level routers and load-balancers).

Thus, an ontology language will be used to describe the characteristics of devices, the means of access to such devices, the policy established by the owner for use of a device, and other technical constraints and requirements that affect incorporating a device into a ubiquitous computing network. The needs established for DAML-S (particularly the issues surrounding the expressiveness of the language) and the RDF-based schemes for representing information about device characteristics (namely, W3C's Composite Capability/Preference Profile (CC/PP) and WAP Forum's User Agent Profile or UAProf) directly relate to this use case and the resource infrastructure which will support applications which will negotiate and dynamically configure ad hoc networks.

3 Design goals

Design goals describe general motivations for the language that do not necessarily result from any single use case. Along with the Use Cases, these motivate the Requirements and Objectives in Sections 4 and 5. In this section, we describe eight design goals for the Web ontology language, in particular those dealing with:

- using established ontologies
- changing established ontologies
- integrating established ontologies
- detecting inconsistencies across ontologies and instances of use
- providing a balance between expressivity and scalability when creating ontologies
- avoiding unnecessary complexity which may discourage widespread adoption
- maintaining compatibility with other standards
- supporting internationalization

For each goal, we describe the tasks it supports and explain the rationale for the goal. We also describe the degree to which RDF and RDF Schema supports the goal.

3.1 Shared ontologies

Ontologies should be publicly available and different data sources should be able to commit to the same ontology for shared meaning. Also, ontologies should be able to extend other ontologies in order to provide additional definitions.

Supported Tasks:
Any use case in which distributed data sources use shared terminology.

Justification:
Interoperability requires agreements on the definitions of identifiers. Ontologies can provide standard sets of identifiers and formal descriptions of those identifiers. Data sources that commit to the same ontology explicitly agree to use the same identifiers with the same meanings.

Often, shared ontologies are not sufficient. An organization may find that an existing ontology provides 90% of what it needs, but the remaining 10% is critical. In such cases, the organization should not have to create a new ontology from scratch, but instead be able to create an ontology which extends an existing ontology and adds any desired identifiers and definitions.

RDF(S) Support:
In RDF, each schema has its own namespace identified by a URI. Each resource in the schema has an ID, and a globally unique identifier can be created by combining the ID with the URI of the namespace. Any resource that uses this URI references the term as defined in that schema. However, RDF is unclear on the definition of a term that has partial definitions in multiple schemas. The specification appears to assume that the definition is the union of all descriptions that use the same identifier, regardless of source. However, this may lead to problems in a distributed environment, where some schemas may contain incorrect or false definitions. There is no way in RDF for a resource to indicate which set of definitions it agrees to.

3.2 Ontology evolution

An ontology may change during its lifetime. A data source should specify the version of an ontology to which it commits.

An important issue is whether or not documents that commit to one version of an ontology are compatible with those that commit to another. Both compatible and incompatible revisions should be allowed, but it should be possible to distinguish between the two. Note that since formal descriptions only provide approximations for the meanings of most identifiers, it is possible for a revision to change the intended meaning of an identifier without changing its formal description. Thus determining semantic backwards-compatibility requires more than a simple comparison of term descriptions. As such, the ontology author needs to be able to indicate such changes explicitly.

Supported Tasks:

Any use case in which the ontology could potentially change, and in particular those in which the owner of the ontology is different from the data providers.

Justification:
Since the web is constantly growing and changing, we must expect ontologies to change as well. Ontologies may need to change because there were errors in prior versions, because a new way of modeling the domain is preferred, or because new terminology has been created (e.g., as the result of the invention of new technology). A web ontology language must be able to accommodate ontology revision. Note that ontology evolution is different from ontology extension, which does not change the original ontology.

RDF(S) Support:
The RDF Schema Specification recommends that each version of a schema should be a separate resource with its own URI. The rdfs:subClassOf and rdfs:subPropertyOf properties can be used to relate new versions of classes and properties to older versions. However, this has the drawback that incorrect definitions cannot be retracted. For example, assume that in schema v1, v1:Dolphin is a rdfs:subClassOf v1:Fish. When this mistake is noticed, the new version of the schema, v2, says that v2:Dolphin is a rdfs:subClassOf v2:Mammal. However, if we make v2:Dolphin a rdfs:subClassOf v1:Dolphin, then we also say that v2:Dolphin is an rdfs:subClassOf v1:Fish which perpetuates the error.

3.3 Ontology interoperability

Different ontologies may model the same concepts in different ways. The language should provide primitives for relating different representations, thus allowing data to be converted to different ontologies and enabling a "web of ontologies."

Supported Tasks:
Any use case in which data from different providers with different terminologies must be integrated.

Justification:
Although shared ontologies and ontology extension allow a certain degree of interoperability between different organizations and domains, there are often cases where there are multiple ways to model the same information. This may be due to differences in the perspectives of different organizations, different professions, different nationalities, etc. In order for machines to be able to integrate information that commits to heterogeneous ontologies, there need to be primitives that allow ontologies to map concepts to their equivalents in other ontologies.

RDF(S) Support:
RDF provides minimal support for interoperability by means of the rdfs:subClassOf and rdfs:subPropertyOf properties.

3.4 Inconsistency detection

Different ontologies or data sources may be contradictory. It should be possible to detect these inconsistencies.

Supported Tasks:
Any use cases in which decentralization of data and lack of controlling authority can lead to conflicts in the data. Any ontology extension task that may result in incoherent descriptions (possibly by extending an ontology in a way that generated an over constrained concept).

Justification:
The Web is decentralized, allowing anyone to say anything. As a result, different viewpoints may be contradictory, or even false information may be provided. In order to prevent agents from combining incompatible data or from taking consistent data and evolving it into an inconsistent state, it is important that inconsistencies can be detected automatically.

RDF(S) Support:
RDF and RDFS do not allow inconsistencies to be expressed.

3.5 Balance of expressivity and scalability

The language should be able to express a wide variety of knowledge, but should also provide for efficient means to reason with it. Since these two requirements are typically at odds, the goal of the web ontology language is to find a balance that supports the ability to express the most important kinds of knowledge.

Supported Tasks:
Any use case that uses large ontologies or large data sets and requires the representation of diverse knowledge.

Justification:
There are over one billion pages on the Web, and the potential application of the Semantic Web to embedded devices and agents poses even larger amounts of information that must be handled. The web ontology language should support reasoning systems that scale well. However, the language should also be as expressive as possible, so that users can state the kinds of knowledge that is important to their applications.

Expressivity determines what can be said in the language, and thus determines its inferential power and what reasoning capabilities should be expected in systems that fully implement it. An expressive language contains a rich set of primitives that allow a wide variety of knowledge to be formalized. A language with too little expressivity will provide too few reasoning opportunities to be of much use and may not provide any contribution over existing languages.

RDF(S) Support:
RDF is very scalable (with the exception of the XML syntax being extremely verbose) but is not very expressive.

3.6 Ease of use

The language should provide a low learning barrier and have clear concepts and meaning. The concepts should be independent from syntax.

Supported Tasks:
Markup and querying of semantic web documents by users, either directly or indirectly.

Justification:
Although ideally most users will be isolated from the language by front end tools, the basic philosophy of the language must be natural and easy to learn. Furthermore, early adopters, tool developers, and power users may work directly with the syntax, meaning human readable (and writable) syntax is desirable.

RDF(S) Support:
RDF is fairly easy to use, but RDF Schema is more complex. The syntax appears to be a major barrier for many.

3.7 Compatibility with other standards

The language should be compatible with other commonly used Web and industry standards. In particular, this includes XML and related standards (such as XML Schema and RDF), and possibly other modeling standards such as UML.

Supported Tasks:
Exchange of ontologies and data in a standard format.

Justification:
Compatibility with other standards eases tool development and deployment of the language.

RDF(S) Support:
RDF has an XML serialization syntax *[RDF/XML Syntax]*.

3.8 Internationalization

The language should support the development of multilingual ontologies, and potentially provide different views of ontologies that are appropriate for different cultures.

Supported Tasks:
Tasks where the same ontology is used by multiple countries or cultures.

Justification:
The Web is an international tool. The Semantic Web should aid in the exchange of ideas and information between different cultures, and should thus

make it easy for members of different nations to use the same ontologies.

RDF(S) Support:
To the extent that XML supports internationalization, so does RDF.

4 Requirements

The use cases and design goals motivate a number of requirements for a Web Ontology language. The Working Group currently feels that the requirements described below are essential to the language. Each requirement includes a short description and is motivated by one or more design goals from the previous section.

R1. Ontologies as distinct resources

> Ontologies must be resources that have their own unique identifiers, such as a URI reference.
>
> **Motivation:** Shared ontologies

R2. Unambiguous concept referencing with URIs

> Two concepts in different ontologies must have distinct absolute identifiers (although they may have identical relative identifiers). It must be possible to uniquely identify a concept in an ontology using a URI reference.
>
> **Motivation:** Shared ontologies, Ontology interoperability

R3. Explicit ontology extension

> Ontologies must be able to explicitly extend other ontologies in order to reuse concepts while adding new classes and properties. Ontology extension must be a transitive relation; if ontology A extends ontology B, and ontology B extends ontology C, then ontology A implicitly extends ontology C as well.
>
> **Motivation:** Shared ontologies

R4. Commitment to ontologies

> Resources must be able to explicitly commit to specific ontologies, indicating precisely which set of definitions and assumptions are made.
>
> **Motivation:** Shared ontologies

R5. Ontology metadata

> It must be possible to provide meta-data for each ontology, such as

author, publish-date, etc. These properties may or may not be borrowed from the Dublin Core element set.

Motivation: Shared ontologies

R6. Versioning information

The language must provide features for comparing and relating different versions of the same ontology. This should include features for relating revisions to prior versions, explicit statements of backwards-compatibility, and the ability to deprecate identifiers (i.e., to state they are available for backwards-compatibility only, and should not be used in new applications/documents.)

Motivation: Ontology evolution

R7. Class definition primitives

The language must be able to express complex definitions of classes. This includes, but is not limited to, sub classing and Boolean combinations of class expressions (i.e., intersection, union, and complement).

Motivation: Balance of expressivity and scalability, Ontology interoperability, Inconsistency detection

R8. Property definition primitives

The language must be able to express the definitions of properties. This includes, but is not limited to, sub properties, domain and range constraints, transitivity, and inverse properties.

Motivation: Balance of expressivity and scalability, Ontology interoperability, Inconsistency detection

R9. Data types

The language must provide a set of standard data types. These data types may be based on XML Schema data types *[XML-SCHEMA2]*.

Motivation: Compatibility with other standards, Ease of use

R10. Class and property equivalence

The language must include features for stating that two classes or properties are equivalent.

Motivation: Ontology interoperability

R11. Individual equivalence

The language must include features for stating that pairs of identifiers represent the same individual (note that consistent with the terminology used in other OWL documents, an individual here is any instance of an OWL class, and does not necessarily mean a person). Due to the distributed nature of the Web, it is likely that different identifiers will be assigned to the same individual. The use of a standard URL does not solve this problem, because some individuals may have multiple URLs, such as a person who has home and work web pages or e-mail addresses.

Motivation: Ontology interoperability

R12. Attaching information to statements

The language must provide a way to allow statements to be "tagged" with additional information such as source, timestamp, confidence level, etc. The language need not provide a standard set of properties that can be used in this way, but should instead provide a general mechanism for users to attach such information. RDF reification may be one possible way to accommodate the requirement, although reification is a somewhat controversial feature.

Motivation: Shared ontologies, Ontology interoperability

R13. Classes as instances

The language must support the ability to treat classes as instances. This is because the same concept can often be seen as a class or an individual, depending on the perspective of the user. For example, in a biological ontology, the class Orangutan may have individual animals as its instances. However, the class Orangutan may itself be an instance of the class Species. Note, that Orangutan is not a subclass of Species, because then that would say that each instance of Orangutan (an animal) is an instance of Species.

Motivation: Ontology interoperability

R14. Cardinality constraints

The language must support the specification of cardinality restrictions on properties. These restrictions set minimum and maximum numbers of individuals that any single individual can be related to via the specified property.

Motivation: Balance of expressivity and scalability goal,, Inconsistency detection

R15. XML syntax

The language should have an XML serialization syntax. XML has become

widely accepted by industry and numerous tools for processing XML have been developed. If the web ontology language has an XML syntax, then these tools can be extended and reused.

Motivation: Compatibility with other standards

R16. User-displayable labels

The language should support the specification of multiple alternative user-displayable labels for the resources specified by an ontology. This can be used, for example, to view the ontology in different natural languages.

Motivation: Internationalization, Ease of use

R17. Supporting a character model

The language should support the use of multilingual character sets.

Motivation: Internationalization, Compatibility with other standards

R18. Supporting a uniqueness of Unicode strings

In some character encodings, e.g. Unicode based encodings, there are some cases where two different character sequences look the same and are expected, by most users, to compare equal. An example is one using a pre-composed form (just one c-cedilla character) and another using a decomposed form (a 'c' character followed by a cedilla accent character). Given that the W3C I18N WG has decided that early uniform normalization (to Unicode Normal Form C) as the usual approach to solving this problem, any other solution needs to be justified.

Motivation: Internationalization, Compatibility with other standards

5 Objectives

In addition to the set of features that should be in the language (as defined in the previous section), there are other features that would be useful for many use cases. These features will be addressed by the working group if possible, but the group may decide that there are good reasons for excluding them from the language or for leaving them to be implemented by a later working group. Some of these objectives are not fully defined, and as such need further clarification if they are to be addressed by the language. Note that the order of the objectives below does not imply relative priority or degree of consensus.

O1. Layering of language features

The language may support different levels of complexity for defining ontologies. Applications can conform to a particular layer without

supporting the entire language. A guideline for identifying layers may be based on functionality found in different types of database and knowledge base systems.

Motivation: Balance of expressivity and scalability

O2. Default property values

The language should support the specification of default values for properties. Such values could be used to make inferences about typical members of classes. However, true default values (such as those used in defeasible inheritance reasoning) are nonmonotonic, which can be problematic on the Web where new information is constantly being discovered or added. Furthermore, there is no commonly accepted method for dealing with defaults. An alternative is for the language specification to recommend to users how they can create their own default mechanisms.

Motivation: Balance of expressivity and scalability

O3. Ability to state closed worlds

Due to the size and rate of change on the Web, the *closed-world assumption* (which states that anything that cannot be inferred is assumed to be false) is inappropriate. However, there are many situations where closed-world information would be useful. Therefore, the language must be able to state that a given ontology can be regarded as complete. This would then sanction additional inferences to be drawn from that ontology. The precise semantics of such a statement (and the corresponding set of inferences) remains to be defined, but examples might include assuming complete property information about individuals, assuming completeness of class-membership, and assuming exhaustiveness of subclasses.

Motivation: Shared ontologies, Inconsistency detection

O4. Range constraints on data types

The language should support the ability to specify ranges of values for properties. This mechanism may borrow from XML Schema data types *[XML-SCHEMA2]*.

Motivation: Inconsistency detection

O5. Chained properties

The language may support the composition of properties in statements about classes and properties. An example of the use of property composition would be the assertion that a property called *uncleOf* is the same as the composition of the *fatherOf* and *brotherOf* properties.

Motivation: Balance of expressivity and scalability

O6. Effective decision procedure

The language should be decidable.

Motivation: Balance of expressivity and scalability

O7. Commitment to portions of ontologies

The language should support the ability to commit to portions of an ontology, as well as committing to an entire ontology. However, it is unclear what granularity should be used here. Possible choices are to choose a subset of concepts with their entire definitions, or to choose individual pieces of definitions. Note that borrowing partial definitions of concepts must address the potential interoperability problems that can arise since different applications will be using the same identifier to mean different things.

Motivation: Shared ontologies

O8. View mechanism

The language should support the ability to create ontology views, in which subsets of an ontology can be specified or concepts can be assigned alternate names. This is particularly useful in developing multicultural versions of an ontology. Note that this requirement may be satisfied by having multiple ontologies and using an ontology mapping mechanism.

Motivation: Internationalization, Ontology interoperability

O9. Integration of digital signatures

The W3C XML Digital Signature specification is an important building block for communication among untrusted properties, which is important for many ontology applications. The web ontology language should be designed in a way that makes it straightforward to use XML Signatures.

Motivation: Compatibility with other standards

O10. Arithmetic primitives

The language should support the use of arithmetic functions. These can be used in translating between different units of measure.

Motivation: Ontology interoperability

O11. String manipulation

The language should support string concatenation and simple pattern

matching. These features can be used to establish interoperability between ontologies that treat complex information as a formatted string and those that have separate properties for each component. For example, one ontology may represent an e-mail address as a single string, while another may divide it into a string for user name and a string for server name. To integrate the two ontologies, one would need to specify that the concatenation of the user name, the '@' character, and the server name is equivalent to the single value used in the first ontology.

Motivation: Ontology interoperability

O12. Aggregation and grouping

The language should support the ability to aggregate information in a way similar to SQL's GROUP BY clause. It should allow counts, sums, and other operations to be computed for each group. This would allow interoperability between ontologies that represented information at different levels of granularity, and could relate things such as budget category totals to budget line item amounts, or the number of personnel to individual data on employees.

Motivation: Ontology interoperability

O13. Procedural attachment

The language should support the use of executable code to evaluate complex criteria. Procedural attachments greatly extend the expressivity of the language, but are not well-suited to formal semantics. A procedural attachment mechanism for web ontologies should specify how to locate and execute the procedure. One potential candidate language would be Java, which is already well-suited to intra-platform sharing on the Web.

Motivation: Ontology interoperability

O14. Local unique names assumptions

In general, the language will not make a *unique names assumption*. That is, distinct identifiers are not assumed to refer to different individuals (see Requirement R11). However, there are many applications where the unique names assumption would be useful. Users should have the option of specifying that all of the names in a particular namespace or document refer to distinct individuals.

Motivation: Inconsistency detection

O15. Complex data types

The language should support the definition and use of complex / structured data types. These may be used to specify dates, coordinate

pairs, addresses, etc.

Motivation: Compatibility with other standards, Ease of use

References

[DWM]
Aseem Das, Wei Wu, and Deborah L. McGuinness. Industrial Strength Ontology Management. in Isabel Cruz, Stefan Decker, Jerome Euzenat, and Deborah L. McGuinness, eds. *The Emerging Semantic Web.* IOS Press, 2002. http://www.ksl.stanford.edu/people/dlm/papers/ontologyBuilderVerticalNet-abstract.html

[Hef]
Heflin, J. *Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment*. Ph.D. Thesis, University of Maryland, College Park. 2001. http://www.cse.lehigh.edu/~heflin/pubs/#heflin-thesis

[RDF Concepts]
Resource Description Framework (RDF): Concepts and Abstract Syntax. Graham Klyne and Jeremy J. Carroll, eds. W3C Proposed Recommendation 15 December 2003. Latest version is available at `http://www.w3.org/TR/rdf-concepts/`.

[RDF Schema]
RDF Vocabulary Description Language 1.0: RDF Schema. Dan Brickley and R.V. Guha, eds. W3C Proposed Recommendation 15 December 2003. Latest version is available at `http://www.w3.org/TR/rdf-schema/`.

[RDF/XML Syntax]
RDF/XML Syntax Specification (Revised). Dave Beckett, ed. W3C Proposed Recommendation 15 December 2003. Latest version is available at `http://www.w3.org/TR/rdf-syntax-grammar/`.

[XML]
Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, 6 October 2000, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, eds. http://www.w3.org/TR/2000/REC-xml-20001006

[XML-SCHEMA2]
XML Schema Part 2: Datatypes - W3C Recommendation, World Wide Web Consortium, 2 May 2001.

Appendix A: Change Log

Changes are listed in reverse chronological order.

Changes Since Candidate Recommendation

- Removed link to WAP Forum's User Agent Profile, which now requires a

password
- Updated references.

Changes Between Last Call and Candidate Recommendation

- Added text to Section 2 explaining that not all requirements implied by the use cases were selected as requirements by the working group.
- Revised text in Section 2.1 about capturing high-quality ontology relationships.
- Changed wording in Section 2.2 so as to not give the probabilistic interpretation some may read into the word "typically."
- Changed wording in Section 2.3 regarding the limitations of a single taxonomy/ontology.
- Changed wording in Section 2.5 regarding the types of ontologies needed.
- Reordered the the last part of Section 2.6.
- Added text to first paragraph of Section 3.
- Added mention of defeasible inheritance reasoning to Objective O2.
- Added reference to OWL Overview document in the introduction.
- Changed Objective O11 to make the example more culturally neutral.
- Added a references section.
- Added this change log.

Acknowledgments

Raphael Volz and Jonathan Dale made significant authorial contributions to this document, and along with the current editor, were co-editors of the initial working drafts. The initial effort to identify design goals and requirements was co-led by Deborah McGuinness and the editor. Some of the requirements were directly contributed by Dr. McGuinness [DWM] based upon over a decade of work in building ontology-based systems. Other requirements were identified as part of the editor's Ph.D. thesis work in building a prototype Semantic Web system [Hef]. A draft version of the Corporate Web Site Management section was written by Michael Smith.

The content of this document is the result of extensive discussions within the Web Ontology Working Group as a whole. The members of this working group included: Yasser alSafadi, Jean-François Baget, James Barnette, Sean Bechhofer, Jonathan Borden, Stephen Buswell, Jeremy Carroll, Dan Connolly, Peter Crowther, Jonathan Dale, Jos De Roo, David De Roure, Mike Dean, Larry Eshelman, Jérôme Euzenat, Tim Finin, Nicholas Gibbins, Sandro Hawke, Patrick Hayes, Jeff Heflin, Ziv Hellman, James Hendler, Bernard Horan, Masahiro Hori, Ian Horrocks, Jane Hunter, Ruediger Klein, Natasha Kravtsova, Ora Lassila, Deborah McGuinness, Enrico Motta, Leo Obrst, Mehrdad Omidvari, Martin Pike, Marwan Sabbouh, Guus Schreiber, Noboru Shimizu, Michael K. Smith, John Stanton, Lynn Andrea Stein, Herman ter Horst, David Trastour, Frank van Harmelen, Bernard Vatant, Raphael Volz, Evan Wallace, Christopher Welty, Charles White, Frederik Brysse, Francesco Iannuzzelli, Massimo Marchiori, Michael Sintek and John Yanosy.


W3C
XHTML
1.0