Requirements of Ontology Languages

Authors: Sean Bechhofer, Carole Goble, Ian Horrocks
Editor: Sean Bechhofer
Affiliations: University of Manchester, UK
OntoWeb Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2000-29243.

Vrije Universiteit Amsterdam (VU)-Coordinator
Faculty of Sciences,
Division of Mathematics and Computer Science
De Boelelaan 1081a
1081 HV Amsterdam, the Netherlands
Fax and Answering machine: +31-(0)84-872 27 22
Mobil phone: +31-(0)6-51850619
Contactperson: Dieter Fensel
E-mail: dieter@cs.vu.nl
# Revision Information

<table>
<thead>
<tr>
<th>Revision date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-03-2002</td>
<td>1.0</td>
<td>Draft</td>
</tr>
<tr>
<td>15-03-2002</td>
<td>1.1</td>
<td>Minor Changes and additions</td>
</tr>
<tr>
<td>22-03-2002</td>
<td>1.2</td>
<td>Final</td>
</tr>
</tbody>
</table>
Introduction

This deliverable provides an overview and summary of work in requirements for Ontology Languages. This is taken from a variety of third party sources, primarily the W3C’s WebOnt working group along with requirements produced by communities such as the bioinformatics and medical communities. Note that as with Deliverable 4.0, this document is not concerned specifically with content, rather with the languages that may be used to deliver content. Particular domains will, by their nature, exact particular requirements from a representation language, but our interest here is in identifying a global collection of requirements (or at least the identification of such requirements).

This document is a collection of information from other sources – it does not seek to present or recommend requirements.

Ontologies

An ontology may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of term. [Usc98].

Guarino [Gua98] defines an ontology as “an intensional semantic structure which encodes the implicit rules constraining the structure of a piece of reality”. A formal ontology has some underlying logical structure which allows us to reason about the concepts in the ontology.

Gruber [Gru93] defines an ontology as ‘the specification of conceptualisations, used to help programs and humans share knowledge’. The conceptualisation is the couching of knowledge about the world in terms of entities (things, the relationships they hold and the constraints between them). The specification is the representation of this conceptualisation in a concrete form. One step in this specification is the encoding of the conceptualisation in a knowledge representation language.

The goal is to create an agreed-upon vocabulary and semantic structure for exchanging information about that domain.

The main components of an ontology are concepts, relations, instances and axioms. A concept represents a set or class of entities or ‘things’ within a domain. Concepts fall into two kinds:

- primitive concepts are those which only have necessary conditions (in terms of their properties) for membership of the class.
- defined concepts are those whose description is both necessary and sufficient for a thing to be a member of the class.

Relations describe the interactions between concepts or a concept’s properties. Relations also fall into two broad kinds:

- Taxonomies that organise concepts into sub- super-concept tree structures. The most common forms of these are
OntoWeb: Ontology-based Information Exchange for Knowledge Management and Electronic Commerce

- Specialisation relationships commonly known as the `is a kind of' relationship.
- Partitive relationships describe concepts that are part of other concepts.

- **Associative** relationships that relate concepts across tree structures. Commonly found examples include the following:
  - Nominative relationships describe the names of concepts.
  - Locative relationships describe the location of one concept with respect to another.
  - Associative relationships that represent, for example, the functions, processes a concept has or is involved in, and other properties of the concept.
  - Many other types of relationships exist, such as `causative' relationships.

The relations, like concepts, can be organised into taxonomies. Relations also have properties that capture further knowledge about the relationships between concepts. These include, but are not restricted to:

- whether it is universally necessary that a relationship must hold on a concept.
- whether a relationship can optionally hold on a concept
- whether the concept a relationship links to is restricted to certain kinds of concepts.
- the cardinality of the relationship.
- other properties such as transitivity, symmetry or reflexivity.

Finally, *axioms* can be used to constrain values for classes or instances. In this sense the properties of relations are kinds of axioms. Axioms also, however, include more general rules.

**Ontology Uses**

There are a number of tasks that ontologies may be used for. This deliverable does not intend to enumerate or describe these tasks in detail, but an overview of these tasks can help in identifying general requirements of languages. The following four categories provide a loose classification of possible tasks. These tasks can then be further explored through Use Cases, such as those adopted by WebOnt.

a) **Exchanging** ontologies – a completely syntactic common formatting and conversion exercise. This guarantees that people will use the same words but guarantees nothing about their interpretation. There are no semantics at this level.

b) **Sharing** ontologies – a common semantic interpretation of the constructs and constraints in the language to support interoperation

c) **Interacting** with ontologies – a set of services supported by the ontology, such as querying or updating reasoning services. For example a Description Logic and an API.

d) **Building** ontologies – the systematic construction of ontologies supported by the interaction services and perhaps supported by classification/subsumption reasoning services.

**Exchanging ontologies**

The exchange of ontologies requires:

i) a delivery language or transport layer e.g. XML
ii) a specification in that language of the ontological constructs and constraints expected by all parties e.g. a DTD in XML

iii) an at worst *informal* description of what the constructs mean, so that each party that participates in the exchange can map those constructs consistently to their own language.

**Sharing/interoperating ontologies**

In order to share, a common semantics for the ontological constructs and reasoning capabilities is needed so that the ontology can be unambiguously interpreted.

**Interacting with ontologies**

Interacting with ontologies requires some specification of the reasoning services that are supplied; e.g. querying, constraints, consistency checking etc. This could be provided as a dynamic service with an API through some sort of ontology server, e.g. OKBC and Ontolingua. It could be bespoke code in an application.

E.g. In a frame based system the hasparent? question could be answered through a traversal through an isakindof slot to the parent frame; in a description logic, the hasparent? question would be answered through a subsumption test. Of course both of these questions rely on a well-defined semantics.

**Building ontologies**

Involves all the above and more. There are further issues that arise when we consider building ontologies, in particular version and change management. Although these are activities that are likely to be supported through tools and methodologies, rather than directly within the language itself, it may be useful to bear them in mind.

**Knowledge Representation Languages**

For ontologies to be used within an application, the ontology must be specified, that is, delivered using some concrete representation. There are a variety of languages which can be used for representation of conceptual models, with varying characteristics in terms of their expressiveness, ease of use and computational complexity. The field of knowledge representation (KR) has, of course, long been a focal point of research in the Artificial Intelligence community.

Major considerations in the choice of representation are the expressivity of the encoding language, the rigour of an encoding and the semantics of a language:

- **The expressivity** of an encoding language is a measure of the range of constructs that can be use to formally, flexibly, explicitly and accurately describe the components of an ontology. For example, first order logic is very expressive. However, there is a trade-off between expressivity (what one can say) and complexity (whether the language is computable in real time).

- **The rigour** of an encoding is a measure of the satisfiability and consistency of the representation within the ontology. A model is satisfiable if none of the statements within contradict each other. Consistency within an ontology is a matter of encoding the conceptualisation of the knowledge in the same manner throughout the ontology. The rigour of an ontology's representational scheme should be maintained by the systematic enforcement of mechanisms using the ontology, which ensures the uniform and universal interpretation of the ontology. Rigour can be maintained computationally via
logic based systems or by the skill of the human encoder. Obviously, in the latter case, mistakes are more easily made and confidence in re-use of the ontology by other developers would be reduced.

- The semantics of a language refers to the fact that it is unambiguously what the language means. For example, the language construct `A subconcept-of B` -- does this mean that all the instances of A are also instances of B, or parts of B, or special kinds of B? Just because two languages use the same syntax does not mean they intend the same meaning. Clearly defined and well-understood semantics are essential if the ontology is to be used within a community for exchange of information.

Languages used for specifying ontologies fall broadly into three kinds: vocabularies defined using natural language; object-based knowledge representation languages such as frames and UML, and languages based on predicates expressed in logic such as Description Logics.

**Vocabularies** support the creation of purely hand-crafted ontologies with simple tree-like inheritance structures. The Gene Ontology [GO], for example, has a hierarchical structure which is asserted -- the position of each concept and its relation with others in the ontology is completely determined by the modeller or ontologist. Each entry or concept in the GO has a name, an identifier and other optional pieces of information such as synonyms, references to external databases and so on.

Although this provides great flexibility, the lack of any structure in the representation can lead to difficulties with maintenance or preserving consistency, and there are usually no formally defined semantics. The single inheritance provided by a tree structure (each concept has only one parent in the is-a hierarchy) can also prove limiting. Maintaining multiple inheritance hierarchies, however, is an arduous task -- the hand-crafting of single inheritance hierarchies is a difficult enough exercise.

A **frame-based system** provides greater structure. Frame-based systems are based around the notion of frames or classes which represent collections of instances (the concepts of the ontology). Each frame has an associated collection of slots or attributes which can be filled by values or other frames. In particular, frames can have a kind-of slot which allows the assertion of a frame taxonomy. This hierarchy can then be used for inheritance of slots, allowing a sparse representation. As well as frames representing concepts, a frame-based representation may also contain instance frames, which represent particular instances.

Frame-based systems have been used extensively in the KR world, particularly for applications in natural language processing. The most well known frame system is Ontolingua. Frames are popular because frame-based modelling is similar to object-based modelling and is intuitive for many users.

The OKBC standard defines a semantics of frame systems, although this is a little unclear in places. For example, it is not always clear how to interpret an assertion that a slot is filled with a particular value. Does this mean that all instances of the frame must have this particular attribute taking this value? Or does the value represent possible fillers for the slot for each instance?

An alternative to frames is logic, notably **Description Logics** (DLs). DLs describe knowledge in terms of concepts and relations that are used to automatically derive classification taxonomies. A major characteristic of a DL is that concepts are defined in terms of descriptions using other roles and concepts. The model is built up from small pieces in a descriptive way, rather than through the assertion of hierarchies. The DL
supplies a number of reasoning services which allow the construction of classification hierarchies and the checking of consistency of these descriptions. These reasoning services can then be made available to applications that wish to make use of the knowledge represented in the ontology. The March 2001 DAML+OIL specification [DAML+OIL] is effectively a Description Logic with an RDF(S) based delivery syntax.

Frames generally provide quite a rich set of language constructs but impose very restrictive constraints on how they can be combined or used to define a class. They only support the definition of primitive concepts, and the kind of taxonomy must be hand-crafted. Description Logics have a more limited set of language constructs, but allow primitives to be combined to create defined concepts. The taxonomy for these defined concepts is automatically established by the logic reasoning system of the Description Logic.

The drawback, however, is that as languages become more and more expressive, the computational complexity of reasoning increases. Recent results, however, show that efficient and practical implementations of expressive languages are feasible, despite their theoretical complexity [Hor99].

A central difference between frame-based approaches and approaches based on Description Logics are that the former rely solely on explicit statements of class-subsumption, whereas the latter are able to efficiently compute the subsumption relationship between classes on the basis of the intensional definition of these classes. Other relations between classes such as disjointness, consistency etc., can all be expressed in terms of the same subsumption relationship. The ability to automatically compute these relations is important for verification of ontologies. This may be less important with small local ontologies that are probably designed by one expert person. However, if the intention is to exchange, share, reuse and merge ontologies, then reasoning support can be a valuable tool. This has been demonstrated even for database schema integration, which should be much easier than integrating ontologies.

**Efforts**

A number of groups and communities have investigated requirements for ontology languages.

The W3C’s WebOnt [WebOnt] working group is currently the highest profile group involved in ontology language specification. Their ultimate findings and recommendations will have significant impact on the field.

Of course, languages for the representation of conceptual or terminological knowledge are not new, and attempts have been made in the past to determine the requirements of such languages. In particular, the medical community has for many years used terminological models (or coding schemes) for the representation of content. These can be seen as simple ontologies (*simple* in terms of their structure – such schemes usually consist of a single taxonomic hierarchy which is often conflated with other relationships such as partonomy. There are often no mechanisms for composition or “formal definition” [Cim98]). Recognition that richer structures were required was made some time ago. Cimino’s desiderata for controlled vocabularies in the 21st Century [Cim98] provide interesting reading here.

The bioinformatics community surveyed requirements in work towards the definition of XOL (an early XML based ontology language proposal). FIPA’s Ontology Service [FIPA
OS] specification provides a number of use cases which also point the way towards requirements for Ontology Languages. Other meetings such as the Semantic Web Working Symposium [SWWS] have also turned their attention to these issues.

WebOnt

The W3C’s WebOnt [WebOnt] working group is a body charged with:

“[defining a] Web ontology language, that builds on current Web languages that allow the specification of classes and subclasses, properties and subproperties (such as RDFS), but which extends these constructs to allow 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.”

To that end, the WebOnt group have been collecting example uses cases for an ontology language and using these use cases to inform the production of requirements for a Semantic Web ontology language. Although WebOnt is still at the Working Draft stage, the findings of the committee will be crucial with respect to OntoWeb, and we provide here a brief overview of the use cases and list of current core requirements arising from the WebOnt activity.

Use Cases

The following six use cases have been identified:

Web portals. Web portals are web sites that collect information on a common topic. Web portals may be based on a specific city or interest area. Each 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 to allow more intelligent syndication, web portals can define an ontology for the community. This ontology can provide an expressive terminology for describing content, and inferences sanctioned by the ontology can be used to improve the quality of search on the portal.

Multimedia collections. Ontologies can be used to provide semantic annotations for collections of images, audio, or other non-textual objects. These annotations can support both indexing and search. 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.

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.

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 these 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.

**Intelligent agents.** 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. When building the actual services, the information may come from a number of sources, such as portals (yahoo.com, citysearch.com, etc.), service-specific sites (marriott.com, hyatt.com, etc.), reservation sites (reservation.com, etc.) and the general Web.

**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.

In addition to the above use cases, other areas identified as being pertinent to the use of ontologies include Bioinformatics/Healthcare and Astrophysics/Space Science. Both of these fields require the modelling of complex data and the integration of multimedia structured (e.g. XML) information.
Design Goals

Analysis of the Use Cases above has produced the following design goals.

**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.

**Ontology evolution.** Ontologies can be changed over time and data sources should specify which version of the ontology they commit to.

**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."

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

**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.

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

**XML syntax.** The language should have an XML serialization.

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

Requirements

The design goals and use cases together suggest the following list of requirements for ontology languages.

**Ontologies as distinct objects**

Ontologies must be objects that have their own unique identifiers, such as a URI reference.

**Unambiguous term referencing with URIs**

Two terms in different ontologies must have distinct absolute identifiers (although they may have identical relative identifiers). It must be possible to uniquely identify a term in an ontology using a URI reference.

**Explicit ontology extension**

Ontologies must be able to explicitly extend other ontologies in order to reuse terms 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.

**Commitment to ontologies**

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

**Ontology metadata**
It must be possible to provide meta-data for each ontology, such as author, publish-date, etc. The language should provide a standard set of common metadata properties. These properties may or may not be borrowed from the Dublin Core element set.

**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 terms (i.e., to state they are available for backwards-compatibility only, and should not be used in new applications/documents.)

**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.

**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.

**Data types**

The language must provide a set of standard data types. These data types may be based on XML Schema data types.

**Class and property equivalence**

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

**Individual equivalence**

The language must include features for stating that pairs of identifiers represent the same individual. 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.

**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 objects (see the previous requirement). 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 objects.

**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.

**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 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.

**Complex data types**
The language must support the definition and use of complex/structured data types. These may be used to specify dates, coordinate pairs, addresses, etc.

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

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

**Supporting a character model**
The language should support the use of multilingual character sets.

**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.

**Cimino’s Desiderata**
In [Cim98], Cimino presents a collection of desiderata for Controlled Medical Vocabularies in the 21st Century. As Cimino says, some of these may seem obvious, but it can never harm to present such criteria as they may be overlooked.

**Content.** Here, the prime need is for the ability to add content and extend existing vocabularies. As usage and domain knowledge expands, so will the need for coverage in a vocabulary or ontology. In particular, formal methodologies are required to support the addition of content. Again, this is not per se a requirement on an ontology language, but care should be taken that representational choices do not impact on the ability to support such methodologies.

**Concept Orientation.** Terms in the ontology should correspond to at least one meaning (nonvagueness) and no more than one meaning (nonambiguity). In addition, meanings should correspond to no more than one term.

**Concept Permanence.** Ontologies may evolve, but once a concept has been created, its meaning should be inviolate. This is important if concepts are to be used or recorded in a persistent fashion. This is not to say that concepts cannot be refined or redescribed. Cimino gives the example of pacemaker being renamed as implantable pacemaker without changing its meaning. Care must be taken, however, that concepts such as non-A-non-B-Hepatitis is not changed to Hepatitis-C as the two definitions are not synonymous. Supporting this requires mechanisms and policies for change management and versioning.

**Nonsemantic Concept Identifier.** Unique identifiers for concepts.
Polyhierarchy. Multiple inheritance in subsumption hierarchies

Formal Definitions. The inclusion of formal definitions (in terms of relationships to other terms in the ontology) is a central requirement of any proposed Semantic Web language. As Cimino says, the key aspect here is that the definitions are in a format which is readily accessible to manipulation by machine – without this there will be little reasoning or inference carried out by agents. Of course the addition of such definitions brings with it additional demands in terms of the modelling.

Reject “Not Elsewhere Classified”. This is a feature particular to medical terminologies, where catch-all terms are used to encode information not explicitly represented by existing terms. Cimino states that the problems with such terms is that they have no formal definition other than on of exclusion. If the language is rich enough to support the notion of negation, however, this becomes less of a problem as it may then be possible to formalize such concepts within the language.

Multiple Granularities. Vocabularies should be capable of supporting multiple levels of granularities. “Insistence on a single level of detail within vocabularies may explain why they often are not reusable”. Different users will require different (but consistent) levels of abstraction.

Multiple Consistent Views. In tandem with a requirement for multiple granularities is the requirement for multiple views. Cimino gives examples such as collapsing fine-grained detailed descriptions into an abstraction or presenting a multiple inheritance hierarchy as a singly inherited tree (which may require duplication of nodes).

Evolve Gracefully. Clear, detailed descriptions of change are required in order to support those maintaining ontologies, for example recording what has changed and why.

Recognize Redundancy. It is sometimes the case that information can be presented in multiple ways. For example, synonyms are a useful device allowing different users to use familiar terms for the same concept. An example given here is of the situation where a user uses a composed term to describe a situation. At a later date the vocabulary is extended to include a term that covers the composition. There are now two ways of coding or representing the concept and these must be reconciled.

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

SWWS Ontologies and Ontology Maintenance Track

The SWWS conference [SWWS] included a track on Ontologies and Ontology Maintenance [SWWSOnto]. One of the outcomes of that track was a list of tools wanted and described. The following list of “tools wanted for (ontology) …” was formulated during the Track Summary session:

- Maintenance
- Versioning
- Collaboration
- Reasoning
- Merging
- Creation
• Validation
• Classification
• Serving
• Management of change
• Tool library management

Again, although this is primarily a tools issue, we can see overlap with the requirements for languages, and it is clear that activities such as versioning, maintenance and classification place requirements on the language. For example, in order to support classification, it is necessary to have a well-defined semantics so that the classification process can itself be well-defined.

FIPA OS

FIPA’s Ontology Service Specification [FIPA OS] describes the services that should be supplied by an Ontology Agent. The specification adopts the OKBC [OKBC] knowledge model for the Ontology Service specification.

The OS specification document describes sample “scenarios” for ontology use. These are not particularly informative in terms of requirements of languages, however, but are more concerned with the ontology level services that may be provided. For example, in Scenario 1, Definition of Terms Querying, an agent queries an ontology service to find sub-species of *citrus*. This requires some notion of hierarchy to be represented within the ontology, but says little else about the representation used within the ontology. Similarly Scenarios 2,3, and 4 are about ontology level interactions (for example Scenario 4 discusses the situation where an agent wants to find a list of ontologies relating to a given term. Scenario 5 discusses term translation through the use of an ontology integrating data source ontologies but is again silent on the requirements of the representation.

Dimensions

Ontology Representation languages have a number of dimensions along which we can evaluate and measure their appropriateness. Any standard proposal must be evaluated against criteria corresponding to these dimensions. The dimensions are not completely independent though, for example the expressiveness of the language will have a direct effect on the provision of reasoning or inference over the language. Criteria to consider include the following.

Language Support and Standardization

General issues concerning the depth of support for the language, including technical support and relationship with existing standards efforts. Also support in terms of environments and tools.

Expressiveness/capabilities

The richness of the expressive capabilities of the language. What assumptions does the language make about the ontology to be represented? If the ontology representation language is to be used across a variety of domains and applications (as will be the case for
a general Semantic Web representation language), such assumptions should be kept to a minimum.

The following list contains potential concept forming operators and capabilities that may be present in a language.

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

### WebOnt and Expressiveness

The WebOnt recommendations do not go into detail regarding the expressivity required, other than recognizing that expressivity and scalability must be balanced. There is a requirement that the language must be able to express complex definitions of classes and a brief discussion of the requirement for expressing relationship cardinalities, but no detailed enumeration of expressivity or operators is given – e.g. stating whether or not negation is available in the language.
Ease of Use

Again related to expressiveness and capabilities, but a language that is not easy to use will not be used. Similarly, ontologies represented using a language must be understandable. An ontology that cannot be understood will not be reused. The expectation is that tools will be provided which will assist in the construction of ontologies, but it is important, for example, to ensure that the underlying language does not preclude ease of use.

The ability to produce clear and formal documentation is important. This is perhaps more of an issue concerned with tool development and support for a language, but the ability to document while constructing an ontology is useful. As a (non-ontological) example, the Javadoc system [JavaDoc] encourages the documentation of Java code during the programming process and, when well-used, can contribute in a major fashion to the reusability and ease of understanding of code.

Layering

Closely related to the capabilities of language. A layered language standard is one in which a variety of choices concerning expressivity can be made. For example, the original OIL language description [OIL] proposed a number of languages ranging from Core OIL through Standard Oil and Instance OIL to Heavy OIL. At each stage, further expressivity was introduced (e.g. Instance OIL added instances to Standard OIL). In this way, it is made obvious to users of the language what is on offer, and what effect (for example in terms of performance or tractability) the use of the features may provide.

The W3C offers a layered approach with its Semantic Web architecture (as shown in Figure 1). Here a declarative logical layer adds a formal semantics and reasoning support to the schema layer provided by RDFS. Further layers on top may add further expressivity that compromises the ability to reason over the models – however this compromise is made clear and explicit.

Thus there is a question of whether to use a layered language, and if so, how to determine those features that should be present in each layer. This in turn will require the identification of those features which are:

- Essential;
- Desirable;
- Optional.
Performance

Some notion of what might be expected in terms of performance if one were to use a given language. Are there any limits (or the limits of available translators/parsers) in the size of the ontology, the length of names/values, etc. (theoretical or practical). What is the overhead (bytes) for a language parser? interpreter?

For resources which depend on an information service for support (such as Ontolingua), does the service have the capacity to support all of the users of the technology?

References and Web Resources


[DAML+OIL] DAML+OIL Language Specification
http://www.daml.org/2001/03/daml+oil


[OIL] Oil Home Page http://www.ontoknowledge.org/oil
