

dOWL: A DYNAMIC ONTOLOGY LANGUAGE

John Avery, John Yearwood
School of Information Technology and Mathematical Sciences
University of Ballarat

ABSTRACT

Ontologies in a web setting, particularly those used in a group context (such as a virtual community), need to be flexible and open to changes that reflect the evolution of knowledge. OWL the ontology language of the semantic web provides very little for facilitating the description of evolutionary changes in an ontology. We propose a dynamic web ontology language (dOWL), an extension to OWL, which consists of a set of elements that can be used to model these evolutionary changes in an ontology.

KEYWORDS

Ontology Evolution, Dynamic Ontologies, OWL, Semantic web, Knowledge Evolution

1. INTRODUCTION

Knowledge in many areas, from molecular biology [7], to legislation [8], is constantly evolving. Knowledge in many areas of the virtual world also evolves[1], both in reflection of real world knowledge evolution and through actions unique to on-line groups like the web log community. If the full potential of the semantic web is to be realised then it must be able to cope with this evolving knowledge.

Ontologies are a corner stone in the semantic web vision, they provide schemas which facilitate knowledge sharing and knowledge re-use. If our knowledge is evolving so too must our ontologies evolve [2,4]. The web ontology language (OWL) [9,6] is the ontology language of the semantic web and although it provides a powerful means of describing and relating ontologies we believe that it is deficient in it's ability to model evolutionary changes in ontology. The problem lies not in OWL's ability to express a new (evolved) version of an ontology, this is no different to expressing any other ontology, but in the inability of OWL to express the semantics of the differences between an old and new version of an ontology. Work in evolving ontologies [2,5,4] has tended to focus on creating, understanding, and recognising the elements of ontology evolution, but has failed to express these differences in a formal language like that presented here.

One of the main advantages of being able to express these differences is one of practicality, if we can express exactly how one ontology differs from another then we can automatically convert documents marked-up with the old ontology into documents that are marked-up with the new ontology. This can potentially add value to existing documents and facilitate the process of document retrieval.

Another advantage of being able to express these differences is that we can use our new understanding to gain insight into old data. This is particular apparent in cases where we change the new version of an ontology in a way which invalidates a document or documents that were marked up with a previous version of the ontology. This kind of identification could, for example, help determine why an argument is void, or why a proposed solution failed.

Finally being able to express these differences provides a model of, and thus means for analysing, knowledge evolution. This kind of analysis tool could help the realisation of more useful virtual communities such as Engelbert's vision of a networked improvement community where community members should find new ways to improve both the way they understand the domain and the way they

perform their tasks [3].

In the next section of this paper we discuss some of the reasons why OWL's versioning mechanism is insufficient for modeling evolutionary changes in ontology and list the ways in which an ontology can evolve. In section 3 we describe our dynamic extension to owl (dOWL) and in section 4 we present our conclusions, focusing on the many avenues of further research provoked by this proposal.

2. WHY OWL ISN'T ENOUGH

OWL provides version support through the elements listed in table 1.

Table 1. The version elements of OWL

<i>Element</i>	<i>Description</i>
<code>owl:imports</code>	Imports the definitions of another OWL ontology into this ontology (this element is not a <i>version</i> element but is important to the discussion).
<code>owl:versionInfo</code>	This element generally contains a string which gives information about the version of this ontology.
<code>owl:priorVersion</code>	This identifies an ontology as a prior version of this ontology.
<code>owl:backwardCompatibleWith</code>	This identifies an ontology as a prior version of this ontology and indicates that this version is backwards compatible with the prior version.
<code>owl:incompatibleWith</code>	This identifies an ontology as a prior version of this ontology and indicates that this version is not backwards compatible with the prior version.
<code>owl:DeprecatedClass</code>	This indicates that a Class has been preserved to maintain backwards compatibility.
<code>owl:DeprecatedProperty</code>	This indicates that a Property has been preserved to maintain backwards compatibility.

Although these elements provide a simple versioning system, if we assume that we want to be able to modify our existing ontology these versioning elements provide us with very little. Through OWL elements we can add a class, property, domain, range, or restriction to an ontology but there are many ways in which an ontology can evolve for which the versioning (or other) elements of OWL do not suffice. Owl provides no means for:

1. Renaming a class or a property;
2. Removing a class or a property;
3. Redefining the restriction of a class;
4. Redefining the domain or range of a property;
5. Redefining a property as a transitive, symmetric, functional, or inverse functional, property;
6. Coalescing many classes or properties into one;
7. Dividing one class or property into many.

Deprecation does not give us the ability to rename a class or property. There is no way to indicate that two classes are the same but that one of them is deprecated. If we indicate that a new class is the same class (using the `rdfs:sameClassAs` element) as the old class, and then we deprecate the old class, we also deprecate the new class. Deprecation does not give us the ability to remove a class or property because the OWL language reference specifies that deprecation has no meaning in terms of model theoretic semantics beyond those defined by the RDF(S) model theory. In other words a class(or property) that is deprecated can still be used exactly as if it were not deprecated.

The other mechanisms listed are beyond the scope of the versioning elements of OWL, but Klein's work on evolving ontology [5] which presents an ontology change management system can identify and map changes to ontology of types 1 through 4.

3. dOWL

The dOWL language provides an enhanced versioning mechanism for OWL which better caters for the evolution of ontologies. A dOWL ontology consists of a set of one or more OWL ontologies followed by a set of zero or more versions that provide a mapping or translation between the old version(s) and the new version. Formally the dOWL language is defined as an OWL ontology at the URL <http://www.ballarat.edu.au/~javery/2003/4/dowl#>. This namespace is informally referred to as dowl for the rest of this document.

A version in a dOWL ontology has the same structure as an OWL ontology except that within the ontology header there are one or more dowl:importsAll or dowl:importsNone elements. The dowl:importsAll element imports all the elements of an ontology just as an owl:imports element except the elements explicitly mentioned within the dowl:importsAll tag. The dowl:importsNone element imports none of the elements of an ontology except the elements explicitly mentioned within the dowl:importsNone tag. The import elements contain zero or more of the elements listed in table 2.

Table 2. Elements inside a dOWL import

<i>Element</i>	<i>Description</i>
dowl:removeClass dowl:removeClassRestrictions	These elements contain an rdf:about which points to a class that should be removed, or have its restrictions removed, from the ontology (if the restrictions are removed they can be redefined as part of the normal OWL ontology).
dowl:removeProperty dowl:removeRange dowl:removeDomain	These elements contain an rdf:about which points to a property that should be removed, or have its domain or range removed, from the ontology (if the domain or range is removed it can be redefined as part of the normal OWL ontology).
dowl:renameClass	This element contains an rdf:ID which defines the new class name and an rdf:about element which points to the class to be renamed.
dowl:renameProperty	This element contains an rdf:ID which defines the new property name and an rdf:about element which points to the property to be renamed.
dowl:defineAsTransitive dowl:defineAsSymmetric dowl:defineAsFunctional dowl:defineAsInverseFunctional dowl:defineAsNormalProperty	These elements contain an rdf:about element which points to the property whose type is to be redefined.
dowl:coalesceClass*	This element contains an rdf:ID element which defines the name of the new coalesced class, and an owl:unionOf of the classes to be coalesced into the new class.
dowl:coalesceProperty*	This element contains an rdf:id element which defines the name of the new coalesced property, and an owl:unionOf of the properties to be coalesced into the new property.
dowl:divideClass*	This element contains an rdf:about element which points to the class to be divided, and an owl:unionOf or enumeration of the classes the old class is to be divided into.
dowl:divideProperty*	This element contains an rdf:about element which points to the property to be divided, and an owl:unionOf of the properties the old property is to be divided into.

* Changes to an ontology of these types preclude the automatic mark-up of document into the new version and further discussion of them is presented in section 4.

4. CONCLUSIONS AND FUTURE WORK

The dOWL language proposed here enables the modelling of evolutionary changes in ontology which facilitates the automated mark-up of old documents into new versions of an ontology, provides a method for analysing ontology evolution, and may prove useful for analysing old documents and for document retrieval.

Future work can progress in many areas, including:

- **The further elaboration of the language, particularly the coalesce and divide elements.** The coalesce properties could be broken down into several sub-properties which define how the classes or properties are to be coalesced (concatenated, combined via a formula, not automatable, etc). The divide properties could also be broken down into several sub-properties which explain how the classes or properties are to be divided (will all old classes be duplicated into several new classes, will they be divided among new classes according to some kind of rule, etc). Other language elements may also be required.
- **The implementation and testing of automatic, or semi-automatic, new version mark-up tools.** Before we can begin to see the benefits of techniques like this we need tools which will give us the ability to make use of the features provided. One of the most important of these is a tool which can automatically mark-up documents, and take note of interesting occurrences (like documents that violate newly imposed restrictions).
- **The exploration of dynamic ontologies in group contexts such as virtual communities and organisations.** Ontologies in group contexts are likely to evolve as the groups understanding of their problem or domain evolves. What tools facilitate this evolution process? Can we use them to make the process more efficient?

It is clear that the semantic web needs some means of not only coping with evolutionary changes in ontology, but modelling them. An extension of the versioning mechanism of the ontology language of the semantic web that provides a mapping between old elements and new elements seems a logical choice and the work presented here demonstrates that this approach is both effective and achievable.

REFERENCES

- [1] Michael Bieber, Starr Roxanne Hiltz, Edward Stohr, Douglas C. Engelbart, John Noll, Murray Turoff, Richard Furuta, Jennifer Preece, and Bartel Van De Wallei. *Virtual community knowledge evolution*. In Proceedings of the Thirty-Fourth Hawaii International Conference on System Sciences, January 2001.
- [2] John Davies, Alistair Duke, and Audrius Stonkus. *Ontoshare: Using ontologies for knowledge sharing*. In Proceedings of the International Workshop on the Semantic Web at the Eleventh International World Wide Web Conference, May 2002.
- [3] Doug Engelbert. *Joint keynote address*. In Hypermedia'98 and Digital Libraries'98, June 1998.
- [4] Jonghyun Kahng and Dennis McLeod. *Dynamic classification ontologies*. In Michael A. Arbib and Jeffery Grethe, editors, *Computing the brain: A guide to Neuroinformatics*. Academic Press, 2000.
- [5] Michel Klein. *Supporting evolving ontologies on the web*. In Proceedings of the EDBT 2002 PhD Workshop, March 2002.
- [6] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. *Owl web ontology language semantic and abstract syntax - w3c working draft*. In <http://www.w3.org/TR/2003/WD-owl-semantics-200303331/>, March 2003.
- [7] Srinivasn Ramachandran, Debasis Dash, and Sami K. Brahmachari. *The multitude of 'omics' and omes': Evolution of scientific terms in molecular biology in the new millenium*. In *Current Science* 77(7), 1999.
- [8] Edwina L. Rissland and M. Timur Friedman. *Detecting change in legal concepts*. In Proceedings of the Fifth International Conference on AI and Law, pages 127-136, 1995.
- [9] Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. *Owl web ontology language reference - w3c working draft*. In <http://www.w3.org/TR/2003/WD-owl-ref200303331/>, March 2003.