OntoView: Comparing and Versioning Ontologies

Michel Klein and **Dieter Fensel**
Vrije Universiteit Amsterdam
mcaklein|dieter@cs.vu.nl

Atanas Kiryakov and **Damyan Ognyanoff**
OntoText Sofia
naso|damyan@sirma.bg

1 Ontology versioning system

In the vision of a Semantic Web, ontologies have a role in defining and relating concepts that are used to describe data on the web. The distributed and dynamic character of the web will cause that many versions and variants of ontologies will arise. Ontologies are often developed by several persons and continue to evolve over time. This will likely cause incompatibilities in the applications and ontologies that refer to them, and will give wrong interpretations to data or make data inaccessible.

To handle ontology changes, it is necessary to maintain the links between the versions and variants, which specify how they are related. These links can be used to re-interpret data and knowledge under different versions. We present a web-based system that supports the user in *finding*, *specifying* and *storing* the relations between ontology versions. The system, called OntoView, provides a transparent interface to different versions of ontologies, by maintaining several aspects of a version relation: the descriptive **meta-data** (i.e., the who, when and why of a change), the **conceptual relations** between constructs in different versions of ontologies, and the **transformations** between the ontology specifications (i.e., a list of update operations). Currently, the system supports RDF-based ontology languages (RDFS, DAML+OIL).

The goal of this system is not to provide a central registry for ontologies, but to allow ontology engineers to store their versions and variants of ontologies and relate them to other (possibly remote) ontologies. The resulting mapping relations between versions can also be exported and used outside the system.

2 Comparing ontologies

One of the central features of OntoView is the ability to compare versions of ontologies. The comparison function is inspired by UNIX `diff`. However, it compares at *structural* level instead of line-level, showing which definitions of ontological concepts or properties have changed.

The comparison function distinguishes between the several types of change. Each change type is highlighted in a different color, and the actually changed lines are printed in boldface.

- Non-logical changes, e.g. in a natural language description. In DAML+OIL, this are changes in the rdfs:label of a concept or property, or in a comment inside a definition.
- Logical definition changes. This is a change in the definition of a concept that affects its formal semantics. Examples of such changes are alterations of subClassOf, domain, or range statements.
- Identifier changes, addition of definitions, or deletion of definitions.

There is also some basic support for the analysis of the effects of changes. On request, OntoView highlights the places in the ontology where conceptually changed concepts or properties are used. In the future, this function should also exploit the transitivity of properties to show the propagation of possible changes through the ontology. Further, we expect to extend the system with a reasoner to automatically verify the changes and the specified conceptual relations between versions.

2.1 Specifying the conceptual implication

The comparison function is an aid for the ontology engineer to find changes in an ontology. To allow for interoperability, the *conceptual implication* of the changes in ontological definitions should be specified. OntoView allows the user to characterize the changes by labelling a changed definition either as "identical" (when the specification has changed but the ontological concept is still meant to be the same), or as "conceptual change" (when the change resulted in a different ontological concept). In the latter case, the logical relation between the two versions can be specified more precisely, e.g. by stating that one version of a concept is a subclass of the other. This information about the conceptual implications of changes complements the descriptive meta-data and the list of change operations.

2.2 Change detection in RDF-based ontologies

To detect changes, we uses the fact that the RDF data model underlies most popular web ontology languages, including RDF Schema and DAML+OIL. The RDF data model basically consists of triples of the form `<subject, predicate, object>`, which can be linked by using the object of one triple as the subject of another. The change detection algorithm splits the ontology in separate definitions, which are then parsed into RDF triples. This results in a set of small RDF graphs, each representing a specific definition of a concept or a property. Next, we locate for each graph in the new version the corresponding graph in the previous version of the ontology. These sets of graphs are then checked according to a number of rules that specify the "required" changes in the triples set (i.e., the graph) for a specific type of change.

3 Conclusion and outlook

OntoView is a system that helps ontology engineers to specify relations between ontology versions in such a way that interoperability is improved. It retains the full transformation of the specification as well as the conceptual relation between versions of concepts and properties. Such support is essential when ontologies are used on the Web and also useful for collaborative development of ontologies. In the future, we would like to add support for other ontology languages and include heuristics to *suggest* the conceptual implications of changes. OntoView will be available at `http://ontoview.org/`.