Supporting evolving ontologies on the Internet

Michel Klein

Vrije Universiteit Amsterdam
De Boelelaan 1081a
1081 HV Amsterdam, the Netherlands
Michel.Klein@cs.vu.nl

Abstract. The idea of a "Semantic Web" has created a lot of interest in the use of ontologies — formal descriptions of a part of the world — for describing the meaning of information on the web. However, when ontologies are used on the web, several problems appear: how can different ontologies be combined, what happens when they are changed, and how should they be adapted for new tasks. Those questions require the management of ontologies, their use and their evolution. This paper describes a research project that will investigate the management of ontologies in an web-based setting.

1 Ontologies on the Web

In the last few years, there has been put a lot of effort in the development of techniques that aim at the "Semantic Web". This envisioned next generation of the World Wide Web will enable computers to *use* the information on the internet. This means that computers will not only distribute and render the information, but also select, combine, transform and relate data, to support humans in performing tasks. Such a web will be more targeted at problem solving and query answering than at mere information retrieval.

A lot of those newly developed techniques require a formal description of a part of *our human* environment, i.e., a description of a part of the real world. Such descriptions, in various degrees of formalness and specificity, are often called *ontologies* [4]. Nowadays, people are taking the first steps towards the "Semantic Web" by annotating (web)data with standard terminologies and other semantic meta data. This is providing us with a lot of freely accessible domain specific ontologies. However, to form a real *web of semantics* — which will allow computers to combine and infer implicit knowledge — those separate ontologies should be linked and related to each other. Adaptation of existing ontologies, and composition of new ontologies from ontologies that are already around will play a central role in this process, as it will both simplify the specification of new knowledge and leverage the interconnection.

The research that is described in this paper will investigate the management of ontologies in a distributed, web-based setting. Ontology management covers several aspects, which are described in more detail in the next section. In section 3, we describe two important building blocks. Finally, we present some ideas on practical validation in section 4.

2 Ontology Management

Ontology management is the whole set of methods, methodologies, and techniques that is necessary to efficiently use multiple variants of ontologies from possibly different sources for different tasks. There are several reasons for ontology management.

2.1 Combination of ontologies

In the envisaged distributed context of a Semantic Web, ontologies from different sources will be linked and combined. However, the reuse of existing ontologies is often not possible without considerable effort [11]. The ontologies either need to be *integrated* [9], which means that they are merged into one new ontology, or the ontologies can be kept separate. In both cases, the ontologies have to be *aligned*, which means that they have to be brought into mutual agreement. The problems that underlie the difficulties in integrating and aligning are the *mismatches* that may exist between separate ontologies. Ontologies can differ at the language level, which can mean that the syntax in which they are represented disagrees, or that the expressivity is different. Ontologies also can have mismatches at the model level, for example in scope, granularity, paradigm, or modeling style [5].

The large number of possible mismatches that can exist between ontologies require that the ontology combination process is guided. Ontology management methods should help to decide which ontologies can be combined and should determine the required adaptations and the possible consequences of the combination.

2.2 Evolving ontologies

Ontologies are often developed by several persons and continue to evolve over time. Domain changes, adaptations to different tasks, or changes in the conceptualization might cause modifications of the ontology. Therefore, both the development and the maintenance of ontologies require advanced versioning methods. Configuration management, that takes care of the identification, relations and interpretation of ontology versions, is necessary.

Related to this is the fact that the evolution of ontologies also causes interoperability problems. The impact of ontology revisions on conforming data or applications that use them is an important aspect of ontology management [6]. Incompatibility for ontologies means that the original ontology can not be replaced by the changed version without causing side effects. The side effects depend on the use of the ontology and the relations that it has.

1. When an ontology is used to specify the meaning of data, this data may get an different interpretation or may use unknown terms. An example of this use is a web page which content is annotated with terms from an ontology.
2. If ontologies are built from other ontologies, changes to the source ontology may affect the meaning of the resulting ontologies.
3. Applications that use the ontology may also be hampered by changes to the ontology. In the ideal case, the conceptual knowledge that is necessary for an application should be merely specified in the ontology; however, in practice applications also use an internal model. This internal model may become incompatible with the ontology.

The interpretation of compatibility is different for each of those types of usage. In the first case, compatibility means the ability to interpret all the data correctly through the changed ontology. This is much like the interpretation of compatibility in database schema versioning. Compatibility here means “preservation of instance data”.

In the second case, the effects of the changes on the logical model that the ontology forms are often important. Other ontologies that import an ontology might depend on the conclusions that can be drawn from the it. A change in the ontology should not make previous conclusions invalid. In this case, compatibility means “consequence preservation”.

Applications that use the ontology might depend on the logical model, but also on the characteristics of the ontology itself. For example, a web site that use an ontology for navigation can depend on the fact that there are only four top-level classes, or that the hierarchy is only three levels deep. A change that does not invalidate queries to instance data or the logical model might invalidate queries to the ontology itself. This interpretation of compatibility is “preservation of answers to ontology queries”.

Changing ontologies thus requires management of the use of ontologies and the effects on the compatibility.

2.3 More complicated than database schema versioning and integration

A lot of the research that has been done out in database schema versioning and integration is relevant and useful for ontology management. Research in this area includes analysis of causes of change [12], effects of different operations on the data [1], frameworks for handling different versions coherently [10] and schema integration methodologies [2]. However, in practice there are significant differences between ontologies and database schemas from the point of view of evolution and versioning. The content and the usage of ontologies are often more complex than that of database schemas. For example, ontologies often incorporate much more semantics than database schema's which might help to solve some integration problems. Also, because their primary goal was knowledge sharing, ontologies are often reused and distributed in a much greater extent than databases. In practice, the data models of ontologies are also richer then those of database schemas. A more detailed analysis of differences can be found in [8]. Ontologies turn some of the theoretical problems and opportunities of database schema versioning into real ones.

2.4 Research question

The central question of this research project can be formulated as follows:

> Which mechanisms and methods are necessary to support the combination of ontologies from different sources and the changes to them in an open, distributed environment.

In the next section, we will describe which building blocks of an ontology management methodology will be developed in this project.

3 Elements for an ontology management framework

.

The contribution of this research project will consist of several aspects. A method for change specification of ontologies on the web will be developed. This will coincide with a methodology for combining, changing and updating online ontologies. The results will be implemented in an automated ontology management system. In this section, we will describe the basis of a change specification method and the first implementation of a web-based ontology management server.

3.1 Change specification mechanism

To determine the requirements for a change specification mechanism, it is important to realize what a *change relation* actually is. Ontologies usually consist of a set of class definitions, property definitions and axioms about them. The classes, properties and axioms are related to each other and together form a model of a part of the world. A change constitutes a new version of the ontology. This new version defines an orthogonal relation between the definitions in the original version of the ontology and those in the new version.

The relations between concepts inside an ontology, e.g. between class A and class B, is thus a fundamentally different relation than the update relation between two versions of a concept, e.g. between class $A_{1.0}$ and class $A_{2.0}$. In the first case, the relation is a purely conceptual relation in the domain; in the second case, however, the relation describes meta-information about the change of the concept.

We distinguish the following properties that are associated with an update relation:

– **transformation** or **actual change**: a specification of what has actually changed in an ontological definition, specified by a set of change operations (cf. [1]), e.g., change of a restriction on a property, addition of a class, removal of a property, etc.;

- **conceptual relation**: the logical relation between constructs in the two versions of the ontology, e.g., specified by equivalence relations, subsumption relations, logical rules, or approximations. The conceptual relation between two versions of a concept is basically a human decision. It might be that the logical definition of a concept has changed, while the definition still refers to the same concept. For example, if a property ‘SSN’ is added to a concept ‘Person’, it is still meant to be the same concept. However, it is also possible that a concept is narrowed or broadened. This should be decided by the person that specifies the conceptualization, i.e. the ontology engineer.
- descriptive meta-data like **date**, **author**, and **intention** of the update: this describes the when, who and why of the change;
- **valid context**: a description of the context in which the update is valid. In its simplest form, this might consist of the date when the change is valid in the real world, conform to *valid date* in temporal databases [10] (in this terminology, the “date” in the descriptive meta-data is called *transaction date*). More extensive descriptions of the context, in various degrees of formality, are also possible.

A well-designed ontology change specification mechanism should take all these characteristics into account.

A desired feature of a language for change specification is the “partial understanding” principle (see also [3]). This principle builds on the idea that a change specification language will extend an existing ontology language. In the case that a specific agent is unable to understand the meta-information about changes (i.e., the extension), it should still correctly handle the conceptual information about the changes, i.e. the **conceptual relation** from the above list.

3.2 Ontology Management Server

In this project, an Ontology Management Server will be developed. The system, called OntoView, will provide a web-based system to manage changes in ontologies. Its main function is to provide a transparent interface to arbitrary versions of ontologies. To achieve this, the system maintains an internal specification of the relation between the different variants of ontologies. It keeps track of the **meta-data**, the **conceptual relations** between constructs in the ontologies and the **transformations** between them. This specification is partly based on change specifications and the versions of ontologies themselves, but also uses additional human input about the meta-data and conceptual implications of changes. It allows users to differentiate between ontologies at a conceptual level and to export the differences as adaptations or transformations.

The system will provide various functions:

- **Reading changes and ontologies.** OntoView will accept changes and ontologies via several methods. Currently, ontologies can be read in as a whole, either by providing a URL or by uploading them to the system. The user has to specify whether the provided ontology is new or that it should be considered as an update to an already known ontology. In the future, OntoView will also accept changes by reading in transformations, mapping ontologies, and updates to individual definitions. These update methods provides the system with different information than the method described above. For that reason, this also requires an adaptation of the process in which the user gives additional information.
- **Identification.** Identification of versions of ontologies is very important. Ontologies describe a consensual view on a part of the world and function as reference for that specific conceptualization. Therefore, they should have a unique and stable identification. A human, agent or system that conforms to a specific ontology, should be able to refer to it unambiguously.

- **Comparing ontologies.** One of the central features of OntoView is the ability to compare ontologies at a conceptual level. This is inspired by UNIX `diff`, but the implementation is quite different. Standard `diff` compares versions at line-level, highlighting the lines that textually differ in two versions. OntoView, in contrast, compares version of ontologies at a *structural* level, showing which definitions of ontological concepts or properties are changed.
  The comparison function distinguishes between the following types of change:
  - Non-logical change, e.g. in a natural language description. In DAML+OIL, this are changes in the rdfs:label of an concept or property, or in a comment inside a definition.
  - Logical definition change. 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. Additions or deletions of local property restrictions in a class are also logical changes. The second and third change in the figure is (class "Male" and property "hasParent") are examples of such changes.
  - Identifier change. This is the case when a concept or property is given a new identifier, i.e. a renaming.
  - Addition of definitions.
  - Deletion of definitions.

  Each type of change is highlighted in a different color, and the actually changed lines are printed in boldface.
  The comparison function also allows the user to *characterize* the conceptual implication of the changes. For the first three types of changes, the user is given the option to label them either as "identical" (i.e., the change is an explication change), or as "conceptual change". In the latter case, the user can specify the conceptual relation between the two version of the concept. For example, by stating that class $A_{1.0}$ is a subclass of $A_{2.0}$.
- **Analyzing effects of changes.** Changes in ontologies do not only affect the data and applications that use them, but they can also have unintended, unexpected and unforeseeable consequences in the ontology itself [7].
  OntoView provides some basic support for the analysis of these effects. First, on request it can also highlight the places in the ontology where conceptually changed concepts or properties are used. For example, if a property "hasChild" is changed, it will highlight the definition of the class "Mother", which uses the property "hasChild". In the future, this function should also exploit the transitivity of properties to show the propagation of possible changes through the ontology.
- **Exporting changes.** The main advantage of storing the conceptual relations between versions of concepts and properties is the ability to use these relations for the re-interpretation of data and other ontologies that use the changed ontology. To facilitate this, OntoView can export differences between ontologies as separate mapping ontologies, which can be used as adapters for data sources or other ontologies. They only provide a partial mapping, because not all changes can be specified conceptually.

4 Experimental validation

The developed ideas for ontology versioning and management will be used in two case studies to guide the research and to function as a kind of "validation". A first experiment is the integration of literature references inside a research community. Several people will be asked to describe a particular piece of information in a semantic way, i.e. via an ontology. Then, the different ontologies will be related and aligned to create a "web of literature references". The goal is to find out which theoretical and practical problems occur when ontologies developed by different people with different perspectives are used for information integration.

As as second experiment, the UNSPSC classification of productswill be used to test the change management on large and simple ontologies. The current UNSPSC standard consists of an hierarchy of about 16.000 product categories. It is heavily used by the industry to classify their product data to allow e-commerce. However, due to the democratic nature of the standard, it changes very frequently. Every two weeks a change description is published which alters between 100 and 600 concepts. This causes serious problems for companies. The experiment will show whether this can be helped by advanced change specifications and automatic support for transparent version access.

5 Discussion and conclusion

Ontology (change) management shares some characteristics with database management and database schema versioning. However, there are also some important differences [8]. The most important ones are the richness of the data model of ontologies, the distributed and re-usable nature of ontologies, and the fact that ontologies often incorporate semantics themselves. All these aspects increase the effects of ontology changes and makes the necessity for ontology management stronger.

In this research project, the peculiarities of ontology management will be analyzed in more detail and the achievements of the work in the database area will be used to developed advanced ontology management methodologies and support. This will be an important building block for the envisaged “web of semantics” that will leverage the use of information on the internet.

References

1. J. Banerjee, W. Kim, H.-J. Kim, and H. F. Korth. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. *SIGMOD Record (Proc. Conf. on Management of Data)*, 16(3):311–322, May 1987.
2. C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies of database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
3. J. Broekstra, M. Klein, S. Decker, D. Fensel, F. van Harmelen, and I. Horrocks. Enabling knowledge representation on the web by extending RDF schema. In *Proceedings of the 10th World Wide Web conference*, Hong Kong, China, May 1–5, 2001.
4. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 1993.
5. M. Klein. Combining and relating ontologies: an analysis of problems and solutions. In A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt, and M. Uschold, editors, *Workshop on Ontologies and Information Sharing, IJCAI'01*, Seattle, USA, Aug. 4–5, 2001.
6. M. Klein and D. Fensel. Ontology versioning for the Semantic Web. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, Stanford University, California, USA, July 30 – Aug. 1, 2001.
7. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 483–493, San Francisco, 2000. Morgan Kaufmann.
8. N. F. Noy and M. Klein. Ontology versioning and database schema versioning, 2001. submitted.
9. H. S. Pinto, A. Gómez-Pérez, and J. P. Martins. Some issues on ontology integration. In *Proceedings of the Workshop on Ontologies and Problem Solving Methods during IJCAI-99*, Stockholm, Sweden, Aug. 1999.
10. J. F. Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, 37(7):383–393, 1995.
11. M. Uschold, M. Healy, K. Williamson, P. Clark, and S. Woods. Ontology reuse and application. In N. Guarino, editor, *Formal Ontology in Information Systems (FOIS'98)*, Treno, Italy, June 6-8, 1998. IOS Press, Amsterdam.
12. V. Ventrone and S. Heiler. Semantic heterogeneity as a result of domain evolution. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 20(4):16–20, Dec. 1991.