CHANGE MANAGEMENT: THE CORE TASK OF ONTOLOGY VERSIONING AND EVOLUTION

Y Liang, H Alani, N R Shadbolt
Intelligence, Agents and Multimedia Group, School of Electronics and Computer Science, University of Southampton
Highfield, Southampton SO17 1BJ
{yl504r | ha | nrs}@ecs.soton.ac.uk

Key words to describe the work: Ontology change, Ontology versioning, Ontology evolution, Ontology management

Key Results: The key issue in the support of evolving ontologies is to distinguish and recognize the changes during the process of ontology evolution. We are proposing an approach to get the evidences of ontology changes, keep track of them, and manage them in an engineering fashion.

How does the work advance the state-of-the-art?: Most of the current popular work on ontology versioning do not keep a record of the changes in the ontology, thus preventing the user from tracking those changes back and forward, or to at least understand the rational behind those changes. Our approach will "force" ontology changes to be fully recorded for future ontology management tasks.

Motivation (problems addressed): Change management as a key issue in ontology versioning and evolution is still not fully addressed, which to some extent forms a barrier against the smooth process of ontology evolution.

Introduction

Ontology in computer science is a borrowed word from philosophy. An ontology defines a common vocabulary of terms, some specification of the meaning of the terms, and a shared understanding for people and machines. Figure 1 shows an example of an ontology.

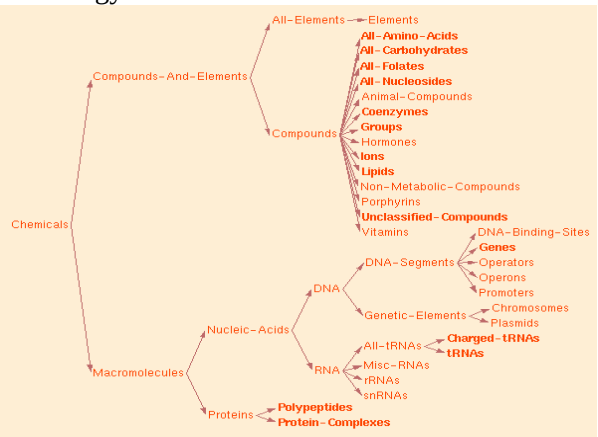

Figure 1 An example of an ontology from EcoCyc[1]

Multiple versions of the same ontology are bound to exist and must be supported appropriately. Some applications keep their ontologies up-to-date, while others may continue to use the old version ontologies and upgrade them at their own pace. These situations may exist because developers can not normally get a whole view of how the ontologies have changed and what were the impacts of those chances. Therefore, ontology change management becomes the key issue in the support of evolving ontologies.

In current stages most of the works in the ontology versioning and ontology evolution borrowed their ideas from the schema versioning (provide access to both old and new data through different interfaces) [1] and schema evolution (provide access to both old and new data through the new schema) [1] in database. However, the usage and content of ontologies are more complicated than database schemas. For example, ontologies incorporate much more semantics than database schema's which might help to solve some integration problems. And ontologies are often reused and distributed to a much greater extent than database schema. Also the data models of ontologies are much richer than those of database schemas. [2] The characteristics of ontologies decided that the concepts of versioning and evolution in database schema cannot be applied directly in ontologies.

The approach we are proposing in this paper is to assist the developers not only to ease the management process of the different versions of an ontology, but also to get the evidences of the changes and keep track of them in order to make the ontology versioning and evolution process go more smoothly.

[1] http://ecocyc.org/

Incompatibility: direct consequence of changes

Versioning support in ontology management is necessary because changes during the process of versioning may cause incompatibilities between ontologies, where old version of an ontology could not be replaced by the changed version without causing some side effects. What constitutes compatibility between the different versions of the same ontology? This question could be answered by looking at the side effects caused by the changes to the ontology during ontology versioning. The side effects can be divided into three main types from the ontology point of view:

- Incompatibility of instance data: The changed version of the ontology could not conform to the meaning of data which the old version depicts.
- Incompatibility of related ontologies: If an ontology (Onto A) is used to build the other ontology (Onto B), then the changes to the source ontology (Onto A) could bring effects to the meaning of the result ontology (Onto B).
- Incompatibility of the related applications: Applications use the ontology to specify the conceptual knowledge that is necessary for the required tasks. The changed version of the ontology used within the application may hamper the usage of the application due to this incompatibility.

Not knowing what caused the changes and how they happened on the ontology hinders reusing those ontologies and causes difficulties for applications to switch to a new version of that ontology. We think that, in an ideal scenario, the developers of the ontology should not only manage and maintain the different versions of their ontology, but also keep track of the detail and rational on how the various versions differ and whether or not it is compatible to switch from an old version to this version. In practice, it is still not very easy for the developers or any ontology management system to achieve those goals simultaneously. Most ontology management systems, for example PROMPT [4] and OntoView [5], have the ability to identify the changes (strictly, identify the differences), and manage the different versions of an ontology, but they can not trace or keep record of the changes made on the ontologies from the beginning, i.e. the changes are untraceable. For the good of our research work, the ontology versioning mechanism and the solution to the issue of compatibility in SHOE [3] have to be mentioned and analysed. To our knowledge, SHOE is currently the only ontology specification language supporting ontology versioning, although it cannot keep track of the evidence of changes from one version to another version.

Some initial ideas on change management and next steps

The efforts of trying to solve the compatibility issue caused by the ontology changes appeared in SHOE. But it was not enough to help tracking or explaining the changes during ontology versioning and evolution processes. In our approach, we propose to keep track of, and record, ontology changes within the ontology itself, i.e. we will add the extra functions to the ontology specification language to enable them to record and keep track of the ontology changes.

We think that a tracable change is composed of a series of the operations acted on the targeted ontologies, for example, add or delete a class, attach a slot to a class, change restriction to a slot, move a class (divided into *add* and *delete* actions respectively). To get the evidence of the changes is transferred to getting the evidence of a series of operations performed on the ontology. But it is important to get those evidences depending on getting rid of any incompatibility issues stated above. Networks are very good for representing and expressing the relationships among the different objects. Our initial idea was enlightened from the family tree which is a kind of simple network. Because it can represent what happened, how happened and what will happen, we hope to put the ontology changes into this kind of evolving network, make a series of changes between the different versions of the same ontology connected with one another which is easy to keep track of. This initial idea is still at the abstract level. More investigation and research is needed to study this approach and test its feasibility.

References

[1] RODDICK, J.F., *A Survey of Schema Versioning Issues for Database Systems.* Information and Software Technology, 37(7): pp383-393

[2] NOY, N.F., KLEIN, M., *Ontology Evolution: Not the Same as Schema Evolution.* Knowledge and Information Systems, 5, 2003

[3] HEFLIN, J., HENDLER, J., *Dynamic Ontologies on the Web*, In: Proceedings of American Association for Artificial Intelligence Conference (AAAI-2000), 2000

[4] NOY, N.F., MUSEN, M.A. *Ontology Versioning in an Ontology Management Framework.* IEEE Intelligent Systems, July-August 2004, 19 (4) pp.6-13

[5] KLEIN, M., *et.al, Finding and characterizing changes in ontologies*, The 21st International Conference on Conceptual Modeling, Tampere, Finland, October, 2002