An Approach to Cope with Ontology Changes for Ontology-based Applications

Yaozhong LIANG, Harith ALANI, David DUPPLAW and Nigel SHADBOLT

Intelligence, Agents and Multimedia Group, School of Electronics and Computer Science, University of Southampton, Highfield, Southampton, SO17 1BJ, U.K.
{y.david.liang, h.alani, dpd, nrs}@ecs.soton.ac.uk

Abstract. Keeping track of ontology changes is becoming a critical issue for ontology-based applications because updating an ontology that is in use may result in inconsistencies between the ontology and the knowledge base, dependent ontologies and dependent applications/services. Current research concentrates on the creation of ontologies and how to manage ontology changes in terms of the attempts to ease the communications between ontology versions and keep consistent with the instances, and there is little work available on controlling the impact to dependent applications/services which is the aims of the system presented in this paper. The approach we propose in this paper is to manually capture and log ontology changes, use this log to analyse incoming RDQL queries and amend them as necessary. Revised queries can then be used to query the knowledge base of the applications/services. We present the infrastructure of our approach based on the problems and scenarios identified within ontology-based systems. We discuss the issues met during our design and implementation, and consider some problems whose solutions will be beneficial to the development of our approach.

1 Introduction

Ontologies are quickly becoming indispensable parts of the Semantic Web. The number of ontologies that are being developed and used by various applications is continuously increasing. One of the major problems with ontologies is change! Ontologies may change for a variety of reasons, such as when the domain itself or our understanding of it changes, when applying modelling corrections, expanding the domain representation, etc.

Ontology changes may cause serious problems to its data instantiations (the knowledge base), the applications and services that might be dependent on the ontology, as well as any ontologies that import that changed ontology [5]. This becomes even more problematic if the applications are not built or controlled by one single party.

There has been much work within the last few years on managing how ontology change, so that such updates can be logged and used to provide better

maintenance and accessability. Most work so far are focused on ways to handle ontology change, such as change characterisation [5], ontology evolution [6], ontology versioning [3], consistency maintenance [8, 10, 13], etc.

However, not much has been done with respect to using change-tracks to eliminate or reduce any impact that ontology change can have on any dependent applications and services. It would be very costly and perhaps even unrealistic to expect all parties that could be affected by a change to coordinate any such changes [2]. Therefore, we believe that it would be very beneficial to have a system that could track such changes, relate changes to incoming queries, amend such queries accordingly, and inform the query source of those changes and actions taken.

In this paper we describe a prototype system that tries to do the above mission. The system logs ontology change and uses it to amend RDQL queries sent to the ontology as necessary. Such a system could save application development many hours by not only updating their queries automatically and maintaining the flow of knowledge to their applications as much as possible, but also to inform the developers of such changes in the ontology related to their queries.

2 Ontology Change Management is Needed for the Semantic Web

To our current knowledge, the interoperability-enabled knowledge represented by the different ontologies and the different versions of the same ontologies is necessary to the Semantic Web. It is therefore important to manage the ontology changes effectively to maintain the relations that specify how the knowledge is related between the different versions of the same ontologies or the various ontologies efficiently to avoid wrong interpretations.

There have been a lot of research activities related to ontology changes. Most of them could fall into one or more of three groups as following:

- Detection and characterisation of change;
- Ontology versioning and evolution;
- Handling inconsistency introduced by ontology change.

To handle ontology changes, detection and characterisation will be the first task. Comparison could be the direct and efficient method to find out the changes between the different versions of the same ontology. Currently, Ontoview [10] and Promptdiff [8] of Protégé are two popular systems to locate the changes between the variants of the same ontology. The granularity level of changes detected by the former (RDF statements) is lower than the latter (structural level).

Currently, there is no agreed versioning and evolution methodology for ontologies on the Web [5]. Within this research area, most of work focuses on

tracking ontology change during evolution process [9, 6] using changes identified by comparison, introducing evolution strategies to allow the developers to specify complex effects of changes [1, 11] and the efforts on defining change operations for ontology language, in particular for OKBC, OWL [4, 7] and for KAON ontology language [12].

The inconsistency introduced by ontology changes will bring unexpected consequences to related ontologies and dependent applications. However, this realm has received little attention so far. It is identified that the impact of a change in the ontology on the function of the system is hard to predict and strongly depends on the application that uses the ontology. Part of the problem is the fact that ontologies are often not just used as a fixed structure but as the basis for deductive reasoning. The functionality of the system often depends on the result of this deduction process and unwanted behavior can occur as a result of changes in the ontology. Haase, et al's work [13] addresses this problem by introducing a common formal basis for comparing the existing approaches to dealing with the inconsistency in terms of a set of elementary definitions.

The ideas that our system is based on fall between the last two groups we described above. Our system is designed to diagnose and fix the inconsistency happening between two versions of the same ontology with respect to the dependent applications/services by using change-tracks in order to eliminate or reduce any impact that ontology change can have on the dependent applications or services.

3 Approach

In this section, we will describe the important components within our approach. In Section 1, we have stated that it could be costly and perhaps impossible to populate every change brought by the dependent ontologies to the applications and services. Our system aims at taking advantage of change-tracks to reduce any impacts on the dependent applications and services brought by the changes of the underlying ontologies. An overview of the approach is shown in Figure 1.

The description will be summarised as follows:

3.1 Approach Description

Problems and Scenarios The main vision of the Semantic Web is that it could provide the end-users with more intelligent services by the means that the machine could understand and communicate with each other based on the knowledge about how to use and combine the information presented by the ontologies. The ontologies with the knowledge they presented are crucial elements in the ontology-based applications in the scenario of the Semantic Web. However, neither the date the ontologies would encode, nor the ontologies themselves are permanent and stable.

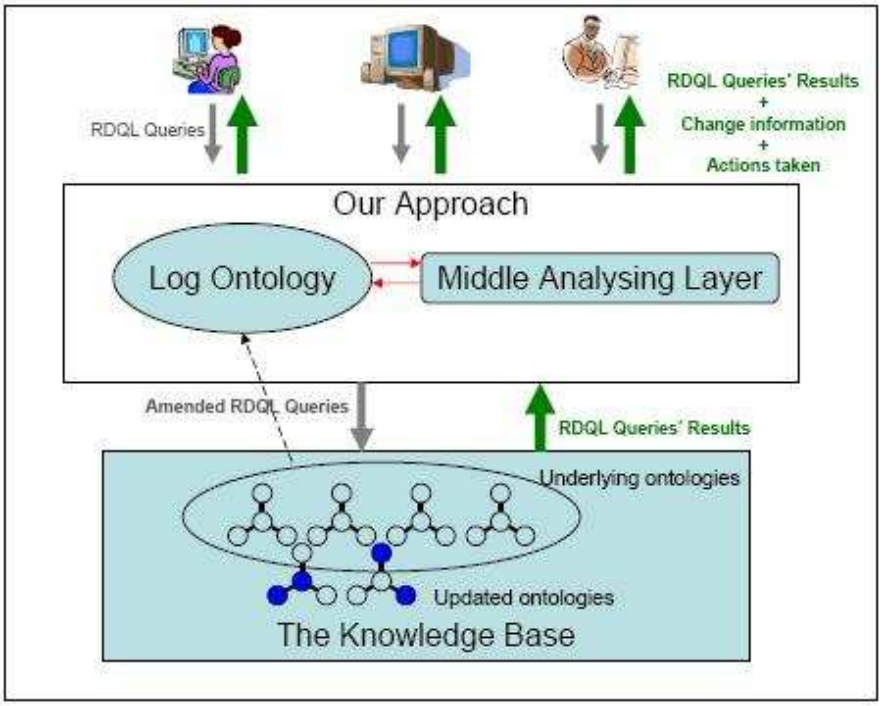

Fig. 1. An overview of the Approach

Ontology change is important to consider for a ontology-based application because changes effect the way data should be interpreted and handled. From the perspective of the end-users, they expect to receive continuous and high-quality services as usual, especially for some particular end-users in the crucial industry where the service could be a part of the service-chain within an application. Any unavailable service at any time would break down the whole system. The direct and indirect consequences could be costly. In this scenario, end-users anticipate that the services would be continuously available within 24*7. In addition, as the end-users, they normally care about the knowledge delivered by the system more than the background supporting elements, such as version update of the underlying ontologies. When the end-users have no knowledge of the updating ontologies, they might have no awareness that the results delivered by the applications and services could be changed as well. In this scenario, end-users anticipate not only obtaining the right knowledge but also to be informed of the updated domain knowledge.

In summary, the end-users expect that the applications and services in the Semantic Web could continually deliver the right knowledge at right time in

an intelligent fashion. Consistently and efficiently coping with ontology changes would be critical for the applications and services to achieve this requirement.

Research Question The central research question in our discussion is the following:

> Which approach is required to cope with ontology changes consistently in order to not only keep the ontology-based applications to provide continuous and updated services as usual, but also make the end-user's domain knowledge up-to-date?

We have identified two sub-questions to assist us to identify the research question more clearly, as follows:

- What is the adequate representation of ontology change?
- Which approach could be developed to resolve the problems caused by the ontology changes identified in our scenarios?

Our Solution for the Approach Within our research, we try to achieve a better understanding of a complicated problem. We developed our understanding by analyzing the context of our problem and comparing it with the related works in the area. Based on this, our approach was introduced to explore a number of techniques that is useful to solve some of the problems we identified in the scenarios described above.

Fig. 2. The Solution of the Approach

Figure 2 shows the solution to tackle the problems identified in our scenarios. The description of each stage is as follows:

1. **Capture**: the changes made between two versions of the same ontology was captured in this stage. Currently, we manually locate changes by comparing two variants using Protégé. In the future, it could be implemented to capture changes by using certain script language.
2. **Manage**: based on the identified changes in the first stage, an appropriate representation of ontology change, called *Log Ontology*, was produced in this stage.

3. **Analyse**: Queries submitted from the applications/services were analysed to find out their effectiveness and usability by checking each entity within the queries coordinating with Log Ontology.
4. **Access**: if certain entities within the users' queries were found being updated to the new versions, they would be replaced by their relevant new versions to form the new queries with updated entities, and then submitted to applications/service for execution.
5. **Response**: after the new-formed queries were submit to the applications or service for execution, the results would returned back as anticipated. In the meantime, change/update information, including the new entity name, domain or range information, and change operations performed during the update process, would also be returned back to the end-users with the query results so as to make users informed of the updated domain knowledge in due course.

3.2 Example Applications – CRM

The application we used within our prototype system intends to be the applications which use CIDOC Conceptual Reference Model [1] (CRM) as the backbone ontology. CRM provides a common language and semantic framework for the domain experts and developers in the culture heritage documentation to share the understanding of the culture heritage information. In this way, any cultural heritage information can be mapped to this framework. CRM acts as a semantic glue to mediate between the different sources of cultural heritage information.

CRM provided detailed documentation with each of its fertile releases of versions, which is helpful to our research on the change process among the different versions of the same ontology. This distinguishing advantage made to choose CRM as the test-bed of our prototype system. We use released version 3.3.2 and 3.4.

4 Discussion and Future Work

In this paper, we discussed the ontology changes as an open problem with respect to their serious effects on the underlying applications/services. We proposed an approach for coping with ontology changes consistently by means of using change-tracks to eliminate or reduce any impact that ontology change can have on the dependent applications and services. For this purpose, we defined the Log Ontology that captures and manages the change information between two versions of CRM ontology. We developed the Middle Analysing Layer that analyses the end-user's queries, amends the entities within the queries accordingly related to the change information captured in the Log Ontology, and informs the end-user of the changes and actions taken. We have implemented a prototypical

[1] The CIDOC Conceptual Reference Model: http://zeus.ics.forth.gr/cidoc/index.html

implementations of this middle layer infrastructure for ontology-based systems and successfully tested it on CRM ontology. We showed that with the extra support of the middle layer ontology-based system could provide continuous and unchanged services to the end-users without the deleterious effects brought by the changes of underlying ontologies, in the mean time, the interested end-user could as also get the knowledge of up-to-date changes taken on the underlying ontologies.

There are some of the ideas that we would like to research and discuss further. By implementing these ideas appropriately in the near future, we believe that our approach will provide a promising method to cope with ontology changes.

- Log Ontology: the Middle Analysing Layer depends on the designed structure and change information organisation of Log Ontology in our approach. The modification to the structure of Log Ontology will bring the essential needs to change the implementation of the Middle Analysing Layer as well. This demands us to figure out a method to increase the flexibility of the implementation of the Middle Analysing Layer. The solution of this issue could add the extra power to the Middle Analysing Layer to make it a crystal interface for another ontology-based applications/services.
- Application/Service: we choose CRM ontology as the underlying ontology for the ontology-based systems. Comparing with the other ontologies we found, though it provides much more available versions, it does not reach our demands as expected. Due to the quantity and quality of changes captured in the Log Ontology, the methods to cope with the different kinds of ontology changes could not be a complete list in our prototype system. We need the other applications/serives to enlarge the scope of our research related to the possible types of ontology changes so as to make our system more fully functional.
- The issues about changes: by analysing the fashion of representation of change information delivered by Log Ontology, two issues attracted our more attentions. For example when is an appropriate time to inform the end-user of the changes besides those changes related to the entities within end-users' queries returned with the results of the queries in the final stage. Because the knowledge represented by the ontologies are correlated, is it a necessity to inform the end-users of any changes happened on these correlations? If so, when to inform them? Also, the change issue we discuss here mainly focuses on the change process, i.e. from the point of view of query, our focus is on the changes represented by Log Ontology during the course of processing the query, not on the query result. What if the knowledge delivered by the query result were changed as well between two different query times? Should we inform the end-users of these type of changes as well? And how could we judge whether the end-users need this information?

Acknowledgement This work has been supported under the Advanced Knowledge Technologies Interdisciplinary Research Collaboration (AKT IRC), which is sponsored by the UK Engineering and Physical Science Research Council under grant number GR/N15764/01.

References

1. Stojanovic, L., et al. User-driven ontology evolution management. In *Proceeding of the 13th International Conference on Knowledge Engineering and Knowledge Management, Ontologies and the Semantic Web*, pages 285–300, 2002.
2. Heflin, J. and Hendler, J. Dynamic ontologies on the web. In *Proceeding of the 17th American Association for Artificial Intelligence Conference (AAAI)*, pages 443–449, Menlo Park, CA, US, 2000. AAAI/MIT Press.
3. Huang, Z. and Stuckenschmidt, H. Reasoning with multi-version ontologies: A temporal logic approach. In *Proceeding of the 4th International Semantic Web Conference (ISWC)*, Galway, Ireland, 2005.
4. Klein, M. *Change Management for Distributed Ontologies.* PhD thesis, Vrjie Universiteit, Amsterdam, 2004.
5. Klein, M. and Fensel, D. Ontology versioning on the semantic web. In *Proceeding of International Semantic Web Working Symposium (SWWS)*, Stanford University, California, U.S.A, 2001.
6. Noy, N.F., Kunnatur, S., Klein, M., and Musen, M.A. Tracking changes during ontology evolution. In *Proceeding of the 3rd International Semantic Web Conference (ISWC2004)*, Hiroshima, Japan, November 2004.
7. Mostowfi, F. and Fotouhi, F. Change in ontology and ontology of change. In *Proceeding of K-CAP 2005 Workshop on Ontology Management for Searching, Selection, Ranking and Segmentation*, Banff, Canada, October 2005.
8. Noy, N.F., and Musen, M.A. Promptdiff: A fixed-point algorithm for comparing ontology versions. In *Proceeding of the 18th National Conference of Artificial Intelligence (AAAI)*, pages 744–750, Edmonton, Alberta, Canada, 2002.
9. Ognyanov, D. and Kiryakov, A. Tracking changes in rdf(s) repositories. In *Proceeding of the 13th International Conference on Knowledge Engineering and Management, Ontologies and the Semantic Web*, Spain, 2002.
10. Klein, M., Kiryakov, A., Ognyanov, D., and Fensel, D. Ontology versioning and change detection on the web. In *Proceeding of 13th International Conference on Knowledge Engineering and Management*, Siguenza, Spain, 2002.
11. Plessers, P., and Troyer, O.De. Ontology change detection using a versioning log. In *Proceeding of the 4th International Semantic Web Conference*, Galway, Ireland, 2005.
12. Stojanovic, L. *Methods and Tools for Ontology Evolution.* PhD thesis, University of Karlsruhe, 2004.
13. Haase, P., Harmelen, F.van, Huang, Z., Stuckenschmidt, H., and Sure, Y. A framework for handling inconsistency in changing ontologies. In *Proceeding of the 4th International Semantic Web Conference (ISWC)*, Galway, Ireland, 2005.