Knowledge-Based Systems 37 (2013) 1–18

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Change management in evolving web ontologies

Asad Masood Khattak [a], Khalid Latif [b], Sungyoung Lee [a,*]

[a] Ubiquitous Computing Lab, Department of Computer Engineering, Kyung Hee University, Global Campus, South Korea
[b] School of Electrical Engineering and Computer Science, National University of Sciences and Technology, H-12 Campus, Islamabad, Pakistan

ARTICLE INFO

Article history:
Received 17 November 2011
Received in revised form 11 May 2012
Accepted 14 May 2012
Available online 7 June 2012

Keywords:
Ontology Change Management
Ontology Evolution
Ontology Recovery
Change History Ontology
Change Navigation

ABSTRACT

Knowledge constantly grows in scientific discourse and is revised over time by different stakeholders, either collaboratively or through institutionalized efforts. The body of knowledge gets structured and refined as the Communities of Practice concerned with a field of knowledge develop a deeper understanding of the issues. As a result, the knowledge model moves from a loosely clustered terminology to a semi-formal or even formal ontology. Change history management in such evolving knowledge models is an important and challenging task. Different techniques have been introduced in the research literature to solve the issue. A comprehensive solution must address various multi-faceted issues, such as ontology recovery, visualization of change effects, and keeping the evolving ontology in a consistent state. More so because the semantics of changes and evolution behavior of the ontology are hard to comprehend.

This paper introduces a change history management framework for evolving ontologies; developed over the last couple of years. It is a comprehensive and methodological framework for managing issues related to change management in evolving ontologies, such as versioning, provenance, consistency, recovery, change representation and visualization. The Change history log is central to our framework and is supported by a semantically rich and formally sound change representation scheme known as change history ontology. Changes are captured and then stored in the log in conformance with the change history ontology. The log entries are later used to revert ontology to a previous consistent state, and to visualize the effects of change on ontology during its evolution. The framework is implemented to work as a plug-in for ontology repositories, such as Joseki and ontology editors, such as Protege. The change detection accuracy of the proposed system *Change Tracer* has been compared with that of *Changes Tab*, *Version Log Generator* in Protege; *Change Detection*, and *Change Capturing* of NeOn Toolkit. The proposed system has shown better accuracy against the existing systems. A comprehensive evaluation of the methodology was designed to validate the recovery operations. The accuracy of Roll-Back and Roll-Forward algorithms was conducted using different versions of SWETO Ontology, CIDOC CRM Ontology, OMV Ontology, and SWRC Ontology. Experimental results and comparison with other approaches shows that the change management process of the proposed system is accurate, consistent, and comprehensive in its coverage.

1. Introduction

Ontologies are formal descriptions of a shared conceptualization of a domain of discourse [14]. Their usage is wide spread in information systems, especially when building a *lingua franca* for resolving the terminological and conceptual incompatibilities between information networks of varying archetype and different provenance [41].

One of the crucial tasks faced by practitioners and researchers in knowledge representation area is to efficiently encode human knowledge in ontologies. Maintenance of usually large and dynamic ontologies, and in particular adaptation of these ontologies to new knowledge, is one of the most challenging problems in Semantic Web research [10,26]. Due to the uncontrolled, decentralized, and complex nature of the Semantic Web, ontology change management is a complicated and multifaceted task. Ontology change management deals with the problem of deciding which modifications to perform in an ontology; implementation of these modifications; and the management of their effects in dependent data structures, ontologies, services, applications, and agents [11]. This has led to the emergence of several different, but closely related, research areas, such as ontology evolution, versioning, integration, and merging [11,15,34]. Ontology evolution research also deals with other associated problems, such as ontology

* Corresponding author. Tel.: +82 31 201 2514.
E-mail addresses: asad.masood@oslab.khu.ac.kr (A.M. Khattak), khalid.latif@seecs.nust.edu.pk (K. Latif), sylee@oslab.khu.ac.kr (S. Lee).

0950-7051/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2012.05.005

matching, which otherwise are fundamentally different [22]. Ontology evolution covers modifications in ontology when there is a certain need for a change as *Communities of Practice* concerned with the field of knowledge develop a deeper understanding of the domain of discourse. Ontology evolution takes place when the perspective under which the domain is viewed has changed [35].

The ontology evolution process deals with the growth of the ontology as well as capturing and accommodating new information [28,43]. As an ontology evolves to a new state, the dependant ontologies and services may become invalid [3,11,22]. Consequently, ontology change management solutions have to answer a number of questions [12]. One such question is, "how to maintain all the changes in a consistent and coherent manner?". During ontology enrichment, modifications are made to the ontology and it evolves from one consistent state to another without preserving information about the previous state. This makes it harder to refer to the previous state unless the changes are preserved in some form. Moreover, the role of change history information becomes critical when an unauthorized user makes changes, or when an ontology engineer wants to revert the changes. A storage structure for such information is also crucial for effective retrieval. Other aspects, such as change traceability and effective visualizations of ontology changes, must also be taken care of by a comprehensive change management framework.

The general aim of this research work, is to provide a mechanism for temporally tracking down changes to an ontology throughout it's life span. In particular the objective is to work on ontology change management, recovery, and visualization of changes and their effects on an ontology to understand the ontology's evolution behavior. To achieve this, all these changes are maintained and managed in a coherent manner.

A Semantically enriched Change History Ontology (CHO) is developed and used to record the ontology changes in a Change History Log (CHL). For proof of the concept, we have developed the system as a plug-in for the ontology editor Protege, that listens and logs all of the ontology changes in CHL. Afterwards, these logged changes are used for ontology recovery (Roll Back and Roll Forward) purposes. We have designed and implemented the Roll Back and Roll Forward algorithms. The logged changes are also used for visualization of changes and their effects at different stages of the evolving ontology. A play back feature is provided to navigate the ontology history for a better understanding of its evolution behavior.

To verify and validate the developed system, we have compared the change capturing ability of the developed plug-in i.e., *ChangeTracer*, with the *ChangesTab* and *VersionLogGenerator* of Protege, *ChangeDetection*, and *ChangeCapturing* of NeOn Toolkit. The results showed that our developed plug-in has better accuracy than the *ChangesTab*, *VersionLogGenerator*, and *ChangeDetection*; whereas having almost same accuracy as *ChangeCapturing*. Moreover, the proposed system uses *difference( )* method to capture and log all the missing changes, whereas, the other systems do not. For validating recovery algorithms (i.e., *RollBack* and *RollForward*), we have tested them on standard data sets, i.e., Semantic Web Technology Evaluation Ontology (SWETO) [1], CRM Ontology [6], Semantic Web for Research Communities (SWRC) Ontology [44], and Ontology Metadata Vocabulary (OMV) Ontology [16]. A high accuracy for both algorithms was observed in these tests. The overall working shows that the proposed system is accurate, consistent, and comprehensive in nature.

The rest of this paper is organized as follows: Section 2 sets the stage by providing the background subject knowledge about the change management and recovery techniques available in ontology and its sibling domains. We have also contrasted our work with other related approaches. A comprehensive description of the semantic structure developed for maintaining the ontology changes is presented in Section 3. Section 4 presents the different possible applications of CHL. Section 5 discusses the recovery algorithms with a running example and presents implementation details. Section 6 presents the details on system implementation. In Section 7, a comparative analysis and validation of results of the proposed methodology using Semantic Web Technology Evaluation Ontology (SWETO), CIDOC Conceptual Reference Model, SWRC Ontology, and OMV Ontology is presented. Finally, we have concluded the research work and have presented an outlook to future research aspects in Section 8.

2. Background and review

Ontology change management deals with the problem of deciding which modifications to perform in ontology in response to a certain need for change [9]. This mechanism ensures that the required changes are reflected in the ontology, and that it is in a consistent state. It deals with four different aspects [10,11,38]. (1) *Ontology evolution* is the process of modifying ontology in response to a certain change in the domain or its conceptualization. (2) *Ontology versioning* is the ability to handle an evolving ontology by creating and managing different versions of it. (3) *Ontology integration* is the process of composing the ontology from information found in two or more ontologies covering related domains. (4) And lastly, *ontology merging* is the process of composing the ontology from information found in two or more ontologies covering highly overlapping or identical domains.

To support the dynamic nature of the Semantic Web, there must be some mechanism to cope with the continuous evolution of domain models and knowledge repositories. Therefore, it is important to manage the ontology changes effectively, and to maintain the relationship between the changes and models [30]. A lot of research on schema evolution has been carried out in relational databases. Schema evolution handles changes in a schema of a populated database without losing data, and provides transparent access to both old and new data through the new schema. It is a complicated task considering the dynamics of ontologies [34], and is also critical to the support of networked ontologies. Ontology evolution and versioning could be amalgamated in a holistic approach to manage ontology changes as well as their effects.

2.1. Ontology evolution process

The process of ontology evolution has the following two variants: *Ontology Population* and *Ontology Enrichment*. New instances of prior coded concepts can be introduced or existing instances can be updated. As a result, the A-Box is changed and reflects new realities. This is called ontology population. Where as in *Ontology Enrichment* process, new domain concepts; properties; or restrictions are introduced or existing ones are updated. This later variant refers to the changes in the schema or T-Box. Overall, the process of evolution takes ontology from one consistent state to another [4,43]. Fig. 1 depicts an overview of this process and shows an interconnection of needed building blocks. In a holistic manner, these components ensure that the ontology has evolved to a consistent new state, incorporating all the required changes. These components are comprehensively discussed in the subsequent sections.

2.1.1. Change detection and description

The first step in the process is to detect changes, whether the suggested changes are already present in the target ontology. Additionally, schema and individual level differences can be detected effectively, as reported in [46]. In case the concept in focus is totally new and there is no additional information, then the H-Match