Dominik Ślęzak Tai-hoon Kim
Jianhua Ma Wai-Chi Fang
Frode Eika Sandnes Byeong-Ho Kang
Bongen Gu (Eds.)

Communications in Computer and Information Science 62

U- and E-Service, Science and Technology

International Conference, UNESST 2009
Held as Part of the Future Generation
Information Technology Conference, FGIT 2009
Jeju Island, Korea, December 2009, Proceedings

Table of Contents

Ontology Evolution: A Survey and Future Challenges

Asad Masood Khattak[1], Khalid Latif[2], Songyoung Lee[1], and Young-Koo Lee[1]

[1] Department of Computer Engineering, Kyung Hee University, Korea
{asad.masood, sylee}@oslab.ac.kr, yklee@khu.ac.kr
[2] School of Electrical Engineering and Computer Science, NUST, Pakistan
khalid.latif@niit.edu.pk

Abstract. Ontology used in many Information Systems and Knowledge Sharing Systems to represent the domain knowledge. As use of ontology increased significantly in recent years that gives importance to proper maintenance of ontology. Ontology change management is a multifaceted and complex task incorporating research areas like; ontology engineering, evolution, versioning, merging, integration, and maintenance. Ontology evolves from one state to another state in response to the changes requested. Crucial task is how to accommodate the new changes while preserving its consistency. This paper provides state of the art analysis of existing approaches covering ontology evolution, and their critical analysis. Pending/Unsolved challenges that need to be address in order to get the process done automatically are also discussed.

Keywords: Ontology Evolution, Emerging Concepts, Change History Ontology, Change History Log.

1 Introduction

Ontologies are formal description of shared conceptualization of a domain of discourse. Ontology serves as back bone of many Information Systems and Knowledge Sharing Systems representing domain knowledge. Ontologies are complex and often large structured, their maintenance give rise to interesting research problems like; evolution, versioning, merging, and integration [FPA06]. The ontology based systems need to have complete and accurate information. So there is a need to keep the ontology up-to-date and should accommodate all the new changes which make the ontology to evolve. Ontology evolves as communities of practices concerned with knowledge develop better understanding of their perceived knowledge [SMM02]. The evolution process deals with the growth of ontology. More specifically, ontology evolution means *modifying or upgrading the ontology when there is a certain need for change or there comes a change in the domain knowledge*.

The process of evolution takes ontology from one consistent state to another [CFH06]. Ontology change management handles the evolution process which may involve different strategies like merging and integration (fundamentally different) [FPA06]. It has several subtasks (see Figure 1); 1) *Capture all the required change(s)* to be applied to ontology and is known as change request. 2) The required *changes are represented* using a common representation format. This representation may be using semantic structure/schema [KLK08a and KLL09] or simple text representation.

D. Ślęzak et al. (Eds.): UNESST 2009, CCIS 62, pp. 68–75, 2009.

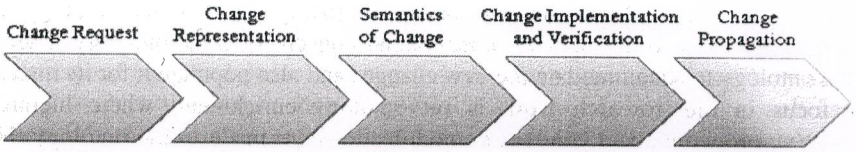

Fig. 1. Process of Ontology Evolution

3) The change effects are tested on the ontology for *consistency* and if required some *deduced changes* are also included in the change, which becomes part of the required changes. 4) The complete change request (modified) is executed by *implementing the changes* in ontology. *Change verification* subtask validates the subject ontology to confirm that the requested changes have been committed to the ontology [KLL09]. 5) Finally, the changes are propagated to dependent data, applications, and ontologies.

Different researchers have provided overlapping solution to the problem of ontology evolution. These approaches do have some pragmatic advantages, but also have several weaknesses, such as: manual specifications of new changes, manually resolving inconsistencies (selecting deduced changes from available alternatives), and also the absence of proper and complete undo and redo facilities in case we want to recover the ontology [KLL09]. To automate the process, these weaknesses need to be eliminated. The automation is also necessary because human intervention is time consuming and error prune. Goal of this research is to provide the survey of ontology evolution approaches. We highlight the main features of all the approaches with the limitations in those systems. After that we discuss some open problems that need to be addressed in order to completely automate the evolution process.

This paper is arranged as follows: Section 2 describes general terms related to ontology change management. Section 3 presents different ontology evolution approaches. In Section 4 we present the challenges still needs to be worked out for automation of evolution procedure. Finally we conclude our discussion in Section 5.

2 Ontology Change Management

Ontology change management activities are sometimes confused together while they are fundamentally different activities [FPA06]. 1) *Ontology Evolution:* is the process of *modifying the ontology when there is a certain need for change or a change in the domain knowledge.* 2) *Ontology Versioning:* is the process of modifying ontology while keeping the original version intact. Mostly used in CVS systems. It handles an evolving ontology by creating and managing different versions of it [FPA06]. 3) *Ontology Merging:* is composition of new ontology from two or more ontologies covering highly overlapping or identical domains, e.g. merging ontologies ACM hierarchy and MSC hierarchy. 4) *Ontology Integration:* is composition of a new ontology from two or more ontologies covering related domains, e.g. integration of health ontology and crime ontology for information of a person.

The evolution is mainly because of new changes and evolution can be of two types. i.e. 1) *Ontology Population:* When we get new instances of concept(s) already present in the ontology. Only the new instance(s) are added and the ontology is populated. 2)

Ontology Enrichment: When we get concept(s), totally new for our ontology or it does have some sort of changes from its counter concept(s) in the ontology. Then we enrich ontology to accommodate the new changes and also populate it for its instance. Our focus in this research work is on ontology enrichment, where hierarchy, concept(s), properties, and constraints modifications are made to the ontology. Here we briefly highlight some of the critical changes while most of these changes are discussed in [CFH06 and Kle04].

- *New Concept:* This is the most common change in any ontology. New concepts emerge and have to be accommodated in the concept hierarchy.
- *Concept Hierarchy:* In this case the concept in focus might have different hierarchical position to the existing one.
- *Concept with Changed Properties:* When the concept in focus is already present in the ontology but its properties are different from the existing one.
- *Concept with Changed Restrictions:* In this case, the concept in focus having restrictions that are dissimilar from those associated with existing concepts.
- *Simple vs. Aggregated Concept:* The concept in focus might be a combination of multiple existing concepts (or vice versa). The ontology evolution framework(s) shall properly detect and act accordingly to accommodate these types of changes.
- *Concept vs. Property:* The concept can either be a class in OWL or used as a property of some other existing class. For example, the concept *deliverable* could be a separate class or could be modeled as property of the concept *project*. In the first case it could have been implemented as a subclass of *document* and in the second case it could take the instances of *software* as its value.

Understanding of change types is necessary to correctly handle explicit and implicit change requirements, and these are the changes that introduce deduced changes in change request.

3 Ontology Evolution Approaches

Ontology over time needs to be updated to accommodate changes in domain, user requirements, and to incorporate incremental improvement in the system. In this section we discuss and critically analyze different ontology evolution approaches.

3.1 User Driven Evolution

L. Stojanovic et al. in [SMM02] proposed a four phase ontology evolution process which copes with ontology changes due to business requirements and dynamic environment. As ontologies are frequently used for information interchange among organizations, so changes in ontology also have ripple effects on the other side use of the same ontology. A process-oriented evolution procedure is presented that focus on the consistency of the ontology during evolution for complex changes. The main modules are; 1) Change representation, the business requirements for change (i.e. change request) is described in formal representational format. 2) Semantic of change, the requested changes are checked on ontology for consistency. It is checked for the actual effects on the dependent data (i.e. instances), applications, and the source

ontology due to these changes. If it results in inconsistency, then the user introduce deduced changes to resolve the inconsistencies. 3) Implementation, the complete change request (included deduced changes) is applied to the source ontology and the ontology evolves to another state. The local instances of ontology as well as the instances of concepts in the ontology are all updated. 4) Propagation, propagation of these changes to dependent data, applications, and ontologies is very important. All the changes are propagated to remote instances as well as applications in a consistent and coherent manner. Change propagation for instances are done in a way that out-of-date instances are simply replaced with the up-to-date instances.

3.2 Evolution Framework for Distributed Ontologies

Change and Annotation Ontology presented in [KlN03] to represent ontology changes, while complete detail of number and types of changes are discussed in [Kle04]. They developed the idea on change ontology to drive various ontology evolution tasks [KlN03]. The idea of change representation is borrowed from [Kle04], but has made extensions like; identification of a changed resource, change author, timestamp, and link to annotations about change. A component based framework for distributed ontology evolution is presented using change representations that formally describe ontology changes required to perform in evolution of ontology.

In [NCL06], the authors on top of their previous work (discussed above) presented different scenarios for ontology maintenance and evolution in distributed and collaborated ontologies. Several features and high-level tasks that an editing environment must support for these scenarios are defined. For this, a unified set of tools for different scenarios and user navigation to different modes are provided.

3.3 Discovery Driven Evolution

In [CFH06], the authors presented an idea of evolution based on the changes recognized by the system after analyzing some domain artifacts. The new concept(s) discovery process proposed supports ontology enrichment activity for multimedia ontology. A multimedia object is first fragmented and then new resources are detected from the fragmented parts. Automatic discovery of change(s) using ontology matching techniques from multimedia objects with additional domain specific metadata is the main achievement. For discovery of new concepts from the fragmented objects, H-Match is used for finding the match for the new concept from the already existing concepts in the ontology. *WorldNet* thesaurus is used for additional metadata. For accuracy improvement, they have implemented four matching models namely; surface, shallow, deep, and intensive discussed in [CFH06].

3.4 Integrated Framework for Evolution Management

We proposed an integrated framework for ontology evolution management [KLL09] with four main modules that supports automatic evolution of ontology from one consistent state to another. 1) Change detection and description, the new changes are automatically detected. These changes are because of emerging concept(s) (single concept, group of concepts and concepts in a hierarchical structure). After detection, these changes are represented in a semantically sound structure (i.e. *Change History*

Ontology (CHO) [KLK08a]) to represent and log all the ontology changes. 2) Inconsistency detection, all possible syntactic and semantic inconsistencies due to change request are resolved here. For this resolution, deduce changes are introduced in the change request. For its implementation, we used the technique of KAON API [GSV04]. 3) Change implementation and verification, here complete change request is applied to the ontology. We focus on changes at atomic level and it completes in isolation. If after applying certain changes we get the ontology inconsistent then there is a loop back facility to inconsistency detection to make the ontology consistent by resolving inconsistencies. After implementation, all the changes are logged in *Change History Log (CHL)* [KLK08a] for undo/redo and recovery purpose. Verification is made to verify that all the requested changes are implemented properly by consulting the change request. 4) Change log, a repository that keeps track of all the changes applied. We have developed a semantic structure i.e. *Change History Ontology* [KLK08a] as a storage structure for ontology changes. It provides facilities like properly managing changes, undo/redo of changes, recovery, visual navigation of change effects on ontology, and visual navigation of ontology changes [KLK08b].

3.5 Critical Analysis

Most of the existing ontology evolution systems do not support automatic ontology evolution except the one proposed in [KLL09] for emerging concepts. In [PIT05 and SMM02], user manually creates requests for changes due to business requirements, while in [OVM04, PIT05, and SMM02], experts are required for conflict resolutions in case of inconsistencies. The system discussed in [CFH06] mostly focused on the discovery of the new change and afterwards the system needs expert to insert the concept at suitable place suggested by the system.

In [KLL09], work related to change detection, its implementation and conflict resolutions are all done automatically. The main concerns in this system are, still a best matching resource search problem for the newly detected change. The second problem is inconsistency resolution. This is the toughest problem that needs special attention. We have to train the system for different deduced changes but training a system for different changes is a tough job. After proper training the system results may not be according to user intentions, as for one conflict resolution there might be many alternative deduced changes. Secondly selecting a deduced change from alternative also needs some predefined criteria.

4 Open Challenges

In this section we will discuss some of the challenges that still need to be addressed to achieve the automated ontology evolution procedure.

4.1 New Change Detection

To detect new changes among the emerging concept(s) (single concept, group of concepts, and concepts in a hierarchical structure), different correspondence, difference, and matching [CFM06 andOVM04] techniques are applied. These techniques suggest the

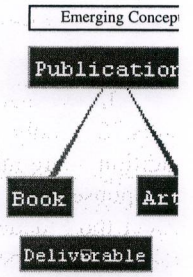

Fig. 2. Emerging

relevance of new er
resources is checked
relevant concept(s) i
should be inserted. H

- *Relevance Dete*
matching algor
results of these
these algorithms
- *Selection amon*
Figure 2, where
with *Book* and *A*
which the chang
concept of *Doc*
alternatives; 1)
make *Deliverabl*
as a property of
is more obvious
should be imple
tough task as
information repr

4.2 Conflict Resolu

Introducing deduce
consistent is the mc
proper solution otl
SMM02, and StM0
evolution procedure
1) to become a *proj*
concept in the sour
then the alternative
the system is traine

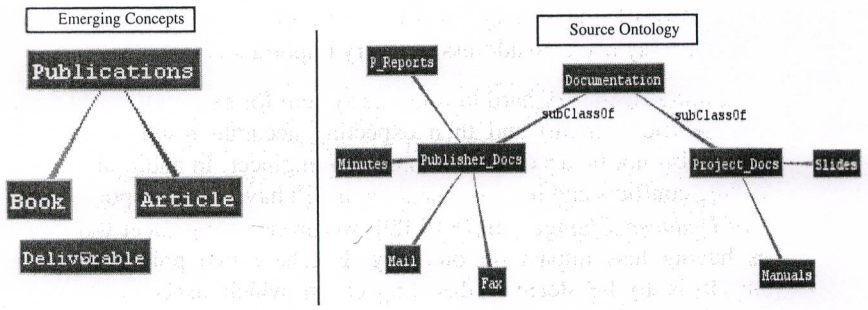

Fig. 2. Emerging concepts, the reason for change in *Documentation* (source) ontology

relevance of new emerging concepts to the source ontology. Then existence of new resources is checked, and if it does not then matching process starts to detect the most relevant concept(s) in the source ontology [CFH06 and KLL09] where the new concept should be inserted. Here we encounter two problems;

- *Relevance Detection:* best and mostly used difference, correspondence, and matching algorithms are defined in [CFM06, OVM04, and HuQ07]. But the results of these algorithms are still not matured for diverse domains, so using these algorithms are not completely reliable and user intervention is required.
- *Selection among Newly Detected Changes:* To understand this issue look at the Figure 2, where we have emerging concepts i.e. *Deliverable* and *Publications* with *Book* and *Article* as sub-concepts, and source ontology of *Documentation* to which the changes will be made. The *Publications* sub-tree is added as a sub-concept of *Documentation* concept. For concept *Deliverable,* we have three alternatives; 1) make *Deliverable* as sub-concept of *Documentation* concept, 2) make *Deliverable* as sub-concept of *Project_Docs* concept, 3) make *Deliverable* as a property of *Project_Docs*. An ontology expert knows that second alternative is more obvious, but the decision is to be made by the system. Proper heuristics should be implemented or system should be trained for such situations. But it is a tough task as ontology is very much different in its nature than any other information representation schemes [NoK04]. These issues are stilled unfolded.

4.2 Conflict Resolution

Introducing deduced changes in order to resolve conflicts and make ontology consistent is the most highlighted problem in ontology evolution literature without proper solution other then involving ontology expert [CFH06, KlN03, PlT05, SMM02, and StM02]. In KAON [GSV04], rules are specified prior to the start of evolution procedure. For example, if there are two alternatives for a concept change, 1) to become a *property* of some concept, and 2) to become a *sub-concept* of some concept in the source ontology (like *Deliverable* concept case in previous section), then the alternative of *sub-concept* should be selected. In [KLL09], we proposed that the system is trained for different types of deduced changes, and then accordingly

select that alternative (deduce change) that have less impact on ontology. To resolve the conflicts in this way needs to address two very important issues.

- *System Training:* It is very hard to train the system for exhaustive list of changes (even of specific domain) and then expecting accurate results. Secondly, the results may also not be acceptable to ontology engineer. In addition, there might be cascading conflicts and in result the system will have weak response time.
- *Impact of Deduced Changes:* In [KLL09], we proposed to select those deduced changes having less impact on ontology. But here two points needs special attention. It is to be decided that impact on which aspect of ontology is considered for deduced changes. There are changes that have large change impact on the structure of ontology but have less impact on the semantics of resources in the ontology. For example (see Figure 2), adding the concept *Publications* as a sub-concept of *Documentation* have large structural impact but less semantic impact. The same way if we make *Publisher_Docs* (see Figure 2) disjoint with its sibling concepts, then this change have less structural impact but can have very large impact on semantic of the resources as its effects will also be reflected on the sub-concept(s) of all the disjoint concept(s).

Currently, these issues are still not resolved properly for automated ontology evolution from one consistent state to another.

5 Conclusions and Step Ahead

Ontology evolution is a collaborative process incorporating work from other related fields such as ontology matching, merging, integration, and reasoning. In this paper we talked about the change management activities and based on these changes the ontology evolves from one state to another state is discussed in detail. We discussed different approaches followed by the research community to handle the evolution process with their pros and cons. At the end we elaborated some open challenges still unhandled to fully automate the process of ontology evolution. Currently, we are working on solutions to these challenges and also on reconciliation of mapping in evolving ontologies.

Acknowledgement

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Advancement)" (IITA-2009-(C1090-0902-0002)) and was supported by the IT R&D program of MKE/KEIT, [10032105, Development of Realistic Multiverse Game Engine Technology].

This work also was supported by the Brain Korea 21 projects and Korea Science & Engineering Foundation (KOSEF) grant funded by the Korea government(MOST) (No. 2008-1342).

References

[CFH06] Castano, S., Semantic W Workshop, I

[CFM06] Castano, S., systems: Te Catarci, T., 25–63. Sprir

[FPA06] Flouris, G., In: The Pos 3rd Italian S

[GSV04] Gabel, T., S D3.1.1.a, SE 2004)

[HuQ07] Hu, W., Qu, Semantics (2

[KLK08a] Khattak, A.M Ontologies. 347–350 (20 Grid (2008)

[KLK08b] Khattak, A.M In: 4th Intern 96 (2008)

[KLL09] Khattak, A.M Framework f Global Econ Malaysia (Jur

[KlN03] Klein, M., N Proceedings CEUR-WS, v

[Kle04] Klein, M.: Ch of Computer S

[NCL06] Noy, N.F., Ch Collaborative Schwabe, D., vol. 4273, pp.

[NoK04] Noy, N.F., K Knowledge ar

[OVM04] Oberle, D., environment. Information S

[PlT05] Plessers, P., I Gil, Y., Mott vol. 3729, pp.

[SMM02] Stojanovic, L evolution man LNCS (LNAI)

References

[CFH06] Castano, S., Ferrara, A., Hess, G.: Discovery-Driven Ontology Evolution. In: The Semantic Web Applications and Perspectives (SWAP), 3rd Italian Semantic Web Workshop, PISA, Italy, December 18-20 (2006)

[CFM06] Castano, S., Ferrara, A., Montanelli, S.: Matching ontologies in open networked systems: Techniques and applications. In: Spaccapietra, S., Atzeni, P., Chu, W.W., Catarci, T., Sycara, K. (eds.) Journal on Data Semantics V. LNCS, vol. 3870, pp. 25–63. Springer, Heidelberg (2006)

[FPA06] Flouris, G., Plexousakis, D., Antoniou, G.: A Classification of Ontology Changes. In: The Poster Session of Semantic Web Applications and Perspectives (SWAP), 3rd Italian Semantic Web Workshop, PISA, Italy (2006)

[GSV04] Gabel, T., Sure, Y., Voelker, J.: KAON – ontology management infrastructure. D3.1.1.a, SEKT Project: Semantically Enabled Knowledge Technologies (March 2004)

[HuQ07] Hu, W., Qu, Y.: Falcon-AO: A Practical Ontology Matching System. Journal of Web Semantics (2007)

[KLK08a] Khattak, A.M., Latif, K., Khan, S., Ahmed, N.: Managing Change History in Web Ontologies. In: International Conference on Semantics, Knowledge and Grid, pp. 347–350 (2008); Fourth International Conference on Semantics, Knowledge and Grid (2008)

[KLK08b] Khattak, A.M., Latif, K., Khan, S., Ahmed, N.: Ontology Recovery and Visualization. In: 4th International Conference on Next Generation Web Services Practices, pp. 90–96 (2008)

[KLL09] Khattak, A.M., Latif, K., Lee, S.Y., Lee, Y.K., Rasheed, T.: Building an Integrated Framework for Ontology Evolution Management. In: 12th Conference on Creating Global Economies through Innovation and Knowledge Management (IBIMA), Malaysia (June 2009)

[KlN03] Klein, M., Noy, N.F.: A component-based framework for ontology evolution. In: Proceedings of the (IJCAI 2003) Workshop on Ontologies and Distributed Systems, CEUR-WS, vol. 71 (2003)

[Kle04] Klein, M.: Change Management for Distributed Ontologies. PhD Thesis, Department of Computer Science, Vrije University, Amsterdam (2004)

[NCL06] Noy, N.F., Chugh, A., Liu, W., Musen, M.A.: A Framework for Ontology Evolution in Collaborative Environments. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 544–558. Springer, Heidelberg (2006)

[NoK04] Noy, N.F., Klein, M.: Ontology evolution: Not the same as schema evolution. Knowledge and Information System 6(4), 428–440 (2004)

[OVM04] Oberle, D., Volz, R., Motik, B., Staab, S.: An extensible ontology software environment. In: Handbook on Ontologies. Series of International Handbooks on Information Systems, pp. 311–333. Springer, Heidelberg (2004)

[PlT05] Plessers, P., De Troyer, O.: Ontology change detection using a versioning log. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 578–592. Springer, Heidelberg (2005)

[SMM02] Stojanovic, L., Madche, A., Motik, B., Stojanovic, N.: User-driven ontology evolution management. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 285–300. Springer, Heidelberg (2002)