A Layered Framework for Pattern-based Ontology Evolution

Muhammad Javed , Yalemisew M. Abgaz , Claus Pahl

Centre for Next Generation Localization (CNGL)
School of Computing, Dublin City University, Dublin 9, Ireland
{mjaved,yabgaz,cpahl}@computing.dcu.ie

Abstract. The challenge of ontology-driven modelling of information components is well known in both academia and industry. In this paper, we present a novel approach to deal with customisation and abstraction of ontology-based model evolution. As a result of an empirical study, we identify a layered change operator framework based on the granularity, domain-specificity and abstraction of changes. The implementation of the operator framework is supported through layered change logs. Layered change logs capture the objective of ontology changes at a higher level of granularity and support a comprehensive understanding of ontology evolution. The layered change logs are formalised using a graph-based approach. We identify the recurrent ontology change patterns from an ontology change log for their reuse. The identified patterns facilitate optimizing and improving the definition of domain-specific change patterns.

Key words: pattern-based ontology evolution, ontology change operators, layered change logs

1 Introduction

Ontology-driven modelling is beneficial for a wide range of information systems aspects. Ontology-based approaches can be used to capture the architecture and process patterns [1]. Ontology-based software models helped researchers to take a step forward from traditional content management systems (CMS) to conceptual knowledge modelling to meet the requirements of the semantically aware software systems. Domain ontologies have become essential for knowledge sharing activities in dynamic enterprise software system. Ontologies can convey the useful semantic information for software developers to understand and process.

Ontology change management is a challenging area. The dynamic nature of knowledge in every conceptual domain requires ontologies to change over time. The operationalisation of changes to ontologies is one of the vital parts of ontology evolution. An ontology may evolve due to the change in the domain, the specification, the conceptualization or any combination of them [2].

We present an approach to deal with ontology evolution through a framework of compositional operators and change patterns. We introduce the notion of layered change logs for explicit operational representation of ontology changes. Some central features of our approach are:

– The representation of changes at a higher level of granularity using domain-specific change pattern.
– The explicit representation of the intent of an ontology change, at a higher level, using layered change logs.
– The discovery of recurring content change patterns provides an opportunity to define reusable domain-specific change patterns.

A brief description of our empirical study of ontology evolution is given in section 2. Layered change logs for ontology change representation are introduced in section 3. We discuss metadata and storage aspects for change logs in section 4. A short evaluation is given in section 5 and we end with some discussion.

2 Empirical Study of Evolution of Domain Ontologies

We studied the evolution of ontologies empirically in order to investigate the relationships between generic and domain-specific changes and to determine common patterns of change. In this paper, we provide a brief description of our empirical study – details can be found in [4].

As a case study, the domains *University Administration* and *Database Systems* were taken into consideration. The former is selected because it represents an organisation involving people, organisational units and processes. The latter is a technical domain that can be looked at from different viewpoints for instance, being covered in a course or a textbook on the subject. We observed that the changes in the database system can be identified by taking different perspectives into account. In teaching, the course content changes almost every year introducing new concepts, theories and languages. In publishing, new database books in the area appear every couple of years resulting in addition of new chapters, merging or removal of existing chapters and changing of the structure of the topics within and among chapters. In industry, new technologies and languages are emerging. These changes result both in structural and instance level change. In university ontology, changes are frequent at instance level due to people joining or leaving, the introduction of new courses etc., but do also occur, albeit more irregular at concept level. Domain experts in both areas have contributed to the study [3]. Based on our observation of common changes in all

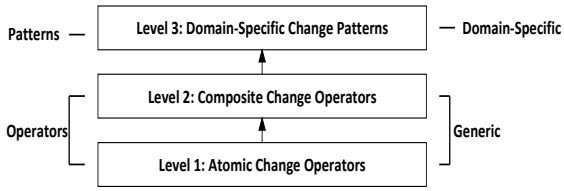

Fig. 1. Layered Framework of Change Operators and Patterns

ontologies, we studied the patterns they have in common, resulting in a layered framework of change operators (Figure 1): *level one* captures elementary changes which are atomic tasks, *level two* captures aggregated changes to represent composite, complex tasks, and *level three* captures domain-specific change patterns.

3 Layered Logs for Ontology Change Representation

Change logs can provide operational as well as analytical support in ontology evolution process. If there is a need to reverse a change, we use the change log to undo/redo the applied changes. In collaborative environments, change logs are used to keep the evolution process transparent and centrally manageable. It captures all changes ever applied to any entity of the ontology. However, we focus on a mechanism of representing ontology changes expressively at different levels of granularity (i.e. fine-grained changes such as the creation of a single class and also coarse-grained changes such as merging two sibling classes) [5].

Representing the change log at the elementary level does not suffice. As the intent of the ontology change is missing from such change logs (and mostly specified at higher domain-specific level of granularity), the ontology engineer is unable to understand why changes were performed. Whether it is an elementary level change or a part of composite change and what the impact of such changes is. We attempt to mine valuable information from a change log, making it easy for the ontology engineer, (other) users and machines to understand and interpret the ontology modifications. We propose a layered change log model, containing two different levels of granularity, i.e. a Basic Change Log (BCL) and a Pattern Change Log (PCL), shown in Figure 2.

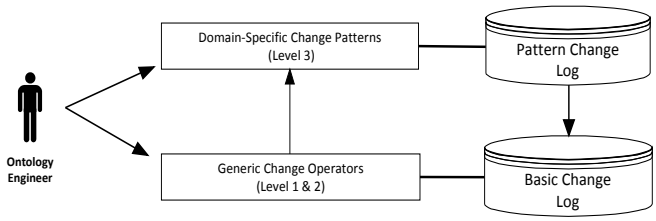

Fig. 2. Layered Change Log

Layered change log work inline with the change operator framework presented in figure 1. The basic change log contains generic level changes and the pattern change log contains the information about the domain-specific change patterns. Using pattern change logs, one can capture the objective of the ontology changes at a higher level of abstraction and will help in comprehensible understanding of ontology evolution. Storing ontology changes at two different levels of abstraction help in identifying recurring domain-specific change patterns from low level logs. We discuss this *pattern discovery* and *pattern matching* in Section 3.2.

3.1 Graph-based Representation for Layered Change Logs

A graph-based representation is an operational representation for the layered change logs. Graphs enable efficient search and analysis and can also communicate information visually. The benefit of a graph-based representation is the

availability of well established algorithms and its well known characteristics such as performance, which can be used for querying the change logs effectively.

Our graph is linear sequential, i.e. there are no concurrent change operations reflected in the graph. We used attributed graphs which are typed over an attributed type graph (ATG) with node and edge attribution. Attributed type graphs ensure that all edges and nodes of a graph are typed over the ATG and each node is either a source or a target connected by an edge. BCL and PCL are typed by a generic ATG where attributes carry labels. Types of nodes refer to the respective ontology elements and types of edges refer to the change operation applied. Several instances of a PCL can occur in a BCL (i.e. a PCL could be a subgraph of BCL expressing that potentially several patterns can capture the same sequence of elementary changes). Figure 3 shows a portion of

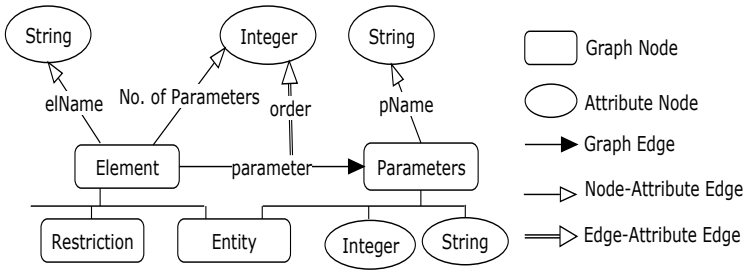

Fig. 3. Attributed Type Graph (ATG) Representation of Change Log

an Attributed Type Graph for a BCL. Each attribute node is named after the data type an instance can have and each graph node represents a conceptual representation of a change log entity. An attributed graph typed over ATG is given in Figure 4. The types defined on the nodes and edges can be represented as t(E1) = Entity, t(Class) = String, t(1) = Integer and t(P1) = String.

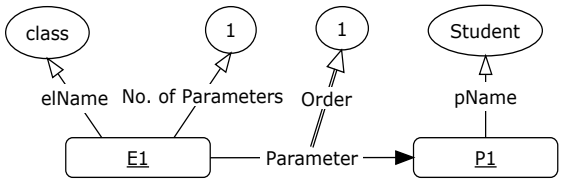

Fig. 4. Typed Attributed Graph of Change Log

3.2 Identification of Recurrent Patterns in Change Log

During our empirical case study, we observed that a number of sequentially ordered change operations are exercised by the users repeatedly during the evolution of domain ontologies. Such a sequenced bundle of change operations are

presented multiple times as a chain of single atomic changes in a basic change log. We are interested in identifying such frequent recurring change patterns automatically. The motivation behind it is the reusability of domain-specific change patterns, in line with the idea of managing change and maintaining consistency through pattern-based ontology evolution.

We considered identifying recurring sequenced change operations from a change log as a problem of recognition of frequent pattern in a graph i.e. graph-based pattern discovery and pattern matching.

- *Pattern Discovery*: As discussed in section 2, domain-specific change patterns can provide guidelines to content change management and support for evolution of information systems. Discovering the recurring (but not yet explicitly defined) change patterns can provide an opportunity to define reusable domain-specific change patterns that can be implemented encapsulating existing information systems.
- *Pattern Matching*: A user can also search for already defined domain-specific change patterns in layered change logs for better understanding on how the ontology evolves through time and in which segments such domain-specific change patterns had been used.

The result of the change pattern identification is a set of subgraphs (change patterns). Based on the resulting subgraphs, a user can select the potential change pattern candidates and store them for further reuse. User may also customise the candidate change patterns by adding/deleting or editing the change operations. Details of these algorithms can be found in [1].

4 Metadata and Storage System for Ontology Change Logs

In order to conceptualise the ontology changes, we constructed a *metadata ontology* by looking into concrete structure of OWL-DL syntax-based domain ontologies. The metadata ontology represents different categories of ontology changes based on our layered change operator framework, types of ontology elements (such as concept, axioms, restriction etc.) and other concepts such as change, users, timestamp etc. Each instance of the change log is of type *Change*, available in the metadata ontology.

In order to implement a uniform and efficient storage solution, we used RDF triple stores. We used sesame native triple store for storage of the domain ontologies, static metadata ontology and change logs. Sesame provides an open source API for fine-grained access to the repository. It offers methods to infer the knowledge which is not explicitly given in the ontology. SPARQL format queries are used to extract the data from the triple store repository.

We identified two types of information, which are essential to be stored into a change log instance i.e., static properties and change properties. Table 1 shows an example of single ontology change log instance *Add subClassOf ("PhDStudent", "Student")*, stored in the triple store. Our ontology editing framework (OnE)

offers a graph API which is used for generating and reading graphs (of type GraphML) from change log triples, extracted from sesame repository.

Table 1. An Example of Change Log Triple stored in a Triple Store

Subject	**Predicate**	**Object**
Static Properties		
Change:142845	rdf:type	Metadata:Change
Change:142845	xsd:ID	"142845"
Change:142845	Metadata:hasCreator	Metadata:Javed
Change:142845	Metadata:Timestamp	xsd:Jan 18 16:28:14 GMT 2011
Change Properties		
Change:142845	Metadata:hasOperation	Metadata:Add
Change:142845	Metadata:hasAxiom	Metadata:subClassOf
Change:142845	Metadata:hasParameter1	University:PhdStudent
Change:142845	Metadata:hasParameter2	University:Student

5 Evaluation

We have looked at practical validity and adequacy of the pattern framework as evaluation criteria. The change operators and patterns we found are based on changes actually carried out by users and ontology engineers and observed by us in both the university administration and database systems ontologies. Pattern change log support higher level of abstraction of ontology changes which are not visible at lower levels. Though it is not expected to be exhaustive, we found that a significant portion of ontology change and evolution is represented in our layered framework, making the supported operators and patterns valid from a practical perspective. Our empirical study results confirm that the lower-level change operators are useful to ontology engineers to suitably define their own change operations, i.e. provide an adequate customisation solution. Domain experts can use the patterns and alter them to meet their requirements by varying the sequence of the content elements.

We conducted experiments on a number of change log case scenarios empirically in order to identify the frequent change patterns. We found that the identified patterns capture the core segments of the ontology evolution and can be reused to construct new domain-specific change patterns. The results acknowledged that the proposed pattern identification framework facilitates a structured evolution process and reduces effort in terms of time.

6 Discussion

The presented work continues our previous research [4] [6] by adding operational, pattern-based change representation and analysis mechanisms. The approach focuses on three levels of change operators which are based on granularity and domain-specificity. Multilayered change logs have been proposed for ontology

change representation and a graph-based representation has been suggested as the formalism to capture and analyse ontology change logs.

Our framework deal with structural and semantic changes at separate levels without loosing their interdependence. Furthermore, it enables us to define a set of domain-specific changes. The empirical study indicates that the solution is valid and adequate to efficiently handle ontology evolution. We have investigated application scenarios where ontologies are used to annotate content. Our observation here is that ontology changes affect the content and vice versa. In these larger systems, change is very frequent and combined with the volume of information affected, makes automated tool support to manage change at the right (i.e. higher, application domain) level of abstraction highly beneficial.

In terms of future work, an optimisation of these in terms of better capturing structural ontology aspects is planned. We will also supplement our technique with improved impact determination and consistency management.

Acknowledgment

This research is supported by the Science Foundation Ireland (Grant 07/CE/I1142) as part of the Centre for Next Generation Localisation at Dublin City University.

References

1. Gacitua-Decar, V., Pahl, C. Ontology-based Patterns for the Integration of Business Processes and Enterprise Application Architectures. In G. Mentzas et al. (Eds). Semantic Enterprise Application Integration for Business Processes: Service-Oriented Frameworks. IGI Pub. 2009.
2. Noy, N.F., Klein, M.: Ontology evolution: Not the same as schema evolution. SMI technical report SMI-2002-0926, 2002
3. Boyce, S., Pahl, C.: The development of subject domain ontologies for educational tech. systems. In Journal of Educational Tech. & Society **10**(3) (2007), pp. 275–288.
4. Javed, M., Abgaz, Y., Pahl, C.: A pattern-based framework of change operators for ontology evolution. In 4th International Workshop on Ontology Content. Volume 5872 of Lecture Notes in Computer Science., Springer (2009), pp. 544–553.
5. Plessers, P., De Troyer, O.: Ontology change detection using a version log. In: Proc. of the 4th International Semantic Web Conference, Springer (2005), pp. 578–592.
6. Abgaz, Y., Javed, M., Pahl, C.: Empirical analysis of impacts of instance-driven changes in ontologies. In: 6th International Workshop on Ontology Content. Lecture Notes in Computer Science, Springer (2010)
7. Gruhn, V., Pahl, C., Wever, M.: Model Evolution as Basis of Business Process Management. In: Proceedings of the 14th Int. Conference on Object-Oriented and Entity-Relationship Modelling (OOER '95). Springer (1995), pp. 270–281.