MedCrave
Step into the World of Research

International Robotics & Automation Journal

Research Article

Open Access

CrossMark

Ontology optimization based on temporal versioning

Abstract

Ontology like any other source of knowledge need to be updated for a bid to adapt the new requirements. Thus, the need for an evolution through temporal versioning is lend itself. This requires a mechanism to implement version management and to identify the links between them. The approach developed in this work allows the management of temporal versioning by preserving the traceability of changes and by applying relevance criteria to the various versions. The relevance is computed based on four criteria: conceptualization, usage frequency, abstraction and completeness. This approach is implemented in a tool dubbed Consistology, which has been developed to assist users in expressing adaptation requirements and managing ontology versions.

Keywords: ontology, temporal versioning, links between versions, relevance criteria, traceability of changes

Volume 4 Issue 1 - 2018

Hanen Jemal

National School of Engineers, Research Groups on Intelligent Machines, Tunisia

Correspondence: Hanen Jemal, National School of Engineers, Research Groups on Intelligent Machines, Tunisia, Email hanen.jemal@ieee.org

Received: December 16, 2017 | **Published:** January 25, 2018

Introduction

During the last decade, much research has been conducted in the engineering of ontology and at different phases of the lifecycle of the ontology. However, most of these studies have focused on the problems of building ontology. Work on the evolution of the ontology is marginal and do not offer so far satisfactory solutions to underlying problems (evolution, versioning, consistency, quality, etc.). Indeed, the fact that the ontology is not static and progress in time, evolution must be considered as a fundamental part of the lifecycle of ontology's. However, to date there is no consensus proposals conducive to manage the problem of evolution through temporal versioning. The problem we are concerned about managing the evolution of ontology's and underlying problems, namely, versioning, relevance and coherence versions of advanced versions. In this work we try to provide some answers to questions raised by the evolution of ontology's, namely:

a. How can we develop an ontology and move from an older version to another version, of course, consistent and appropriate under the new needs of the modeled domain?

b. How to determine the links between the different versions of the ontology?

c. How to introduce the time dimension in the process of versioning?

d. What version should I save throughout the process of versioning?

As a response to the first question, we rely on the evolution operators defined in the literature. The application of consistent sets of changes defined in[1] to produce an advanced version of the consistent ontology. Each version is identified by a number and the date of creation, the operators' change that helped generate it and a link to previous and subsequent versions. When the number of versions exceeds a given threshold versions, the most relevant options have been retained, depending on certain parameters of relevance. In order to achieve these objectives, a method of Advanced Simple Additive Weighting (ASAWA) and the graph of relevance temporal have been achieved. This section contains, the state of the art approach of versioning, also the mechanism for managing relationships between versions and optimization of storage and the extension of the tool dubbed "Consistology".

State of the art

Several approaches to ontology evolution have been proposed in the literature. Stojanovic[2] defines various concepts needed to manage the evolution of ontology. It introduces a taxonomy of changes that has three levels of classification: The basic changes, composite and complex. This approach merely represents the ontology in its latest version. Based on research conducted by Stojanovic,[2] there are terminologies: the management of ontology changes, evolution and versioning:

a. Ontology Management: This is a set of methods and techniques are developed to effectively use multiple variants of ontology's, possibly from different sources to solve different tasks.

b. Changing the ontology management system allows the ontology to make changes on the ontology under consideration, including its state of consistency.

c. Evolution of the ontology management system of the ontology facilitates the modification of ontology by preserving its state of consistency.

d. Version management of ontology (versioning): the management system of the ontology allows the manipulation of the ontology changes by creating and managing its various versions.

The method[3] allows the management and conservation of different versions of the ontology evolution. However, the method does not indicate how the versions of the ontology can be created. A mixed approach that includes the previous two had been proposed. She proceeded to the representation of the evolution of ontology's as directed graphs.[4] It is based on the concepts and techniques developed for temporal databases: changing patterns and versioning databases. This method does not uncover links between the different versions of ontological entities, because the change history of an ontological entity is not required. Other researchers address the problem of the evolution of ontology in connection with its use. In

Luong[5] presents a new approach that supports both the evolution of ontology and semantic annotations produced with this ontology, as part of a corporate semantic web. It develops ontology of evolution which aims to model in a formal way the kinds of changes and information about the evolutionary process. In the study presented in Rogozan[6] integrates the above methodologies (work by Klein and Stojanovic) in a unified methodology applied to online learning systems. This method expands the conceptualization[7] by adding features such as:

a. a typology of complex changes,

b. a description of the effects of changes on the semantic annotation (the formal representation of the contents of a resource),

c. Clarification of the terminology used to characterize the complexity of change.

More recent approaches[8,9] focus on the aspect of the evolution of ontology. In[8] authors construct a definition language changes (Change Definition Language) to formally represent the ontology changes. In Flouris[9] introduce the notion of belief change (Belief change) to address compliance knowledge representation in OWL ontology's.

Proposed approach

In this work an approach for temporal versioning for ontology optimization based on three steps

Step 1: Evolution change

Identification of change: This phase identifies the needs for change following a detection of a new concept to the area where the needs expressed by the user.

Representation of change: in addition to representing the changes (elementary or composite) in a format that clearly expressed formally, we define inverse operators.

Step 2: Consistency of ontology includes the propagation of changes and consistency of analysis

Propagation of changes: during this step determines the types of change: direct and indirect performed on the ontology.

The analysis of consistency: check side effects of each change opposite to anticipate inconsistencies and correction solutions proposed. While applying a set of consistency rules in the form of additional changes, these operations undergo a verification process to ensure that they are causing their share of inconsistencies.

Step 3: Manage versions

At the end of this process, a new version is created: At this time, the designer decides to preserve the old version or deleted.

Versioning optimization

The conservation of a large number of versions of the ontology requires significant memory and causes difficulty in handling the amount of information stored. Thus, we direct our thinking towards conservation of the versions most relevant according to predefined criteria. For each version, we associate a relevance value. When the number of versions becomes very large, the versions that are no longer relevant will be deleted. In case of equal value relevance between two or more versions, we eliminate the oldest. The list of versions kept is a list whose first element is the kernel version and the last is the current version. The remaining elements represent the most relevant versions chronologically ordered. So far, there is no consensus within the engineering community about the ontological relevance criteria versions at the time of archiving. The only criteria are the audit criteria in the existing literature[10,11]concern only the development phase (audit criteria) of the ontology. It is important to respect a number of constraints to obtain a quality, adaptable, maintainable and historicizing. Below a set of criteria that seem particularly appropriate to decide the relevance of models:

a. Conceptualization (criterion 1): the richness of the content design of the ontology (information richness). This helps promote the conservation of versions semantically richer and structurally.[1] Each version will be compared with others to verify the information contained therein already exists in other versions. If this is the case, its removal will not result in loss of information, i.e, all ontological entities created at a given stage of evolution (which existed in earlier versions) can be found and retrieved from other versions if necessary.

b. Frequency of use (criterion 2): to count the number of uses (access) in each release and promote the versions most used and most requested by user queries.

c. Abstraction (criterion 3): the level of abstraction class (generalization / specialization) by considering the depth of hierarchies.

d. Completeness (criterion 4): represents the degree of coverage of sources representative of the model domain and its relevant properties, taking into account the compliance of designations (labels) classes and properties in keywords in the field.

e. Size of the ontology (criterion 5): the number of domain concepts that are structured. To preserve these criteria become very difficult choice for that we offer developers an objective method of fixing the number of versions to historicize and implement an appropriate method.

Advanced simple additive weithing (ASAW)

ASAW (Advanced Simple Additive Weithing) to determine an overall score to the version according to the criteria previously reported.

$$Gs\left(Global\ SAW\right) = \sum \alpha i\ si \qquad (1)$$

αi: weight of criterion i

α: level of importance of each criterion

Note: the criteria must be distinct: no correlation or overlap. The algorithm for calculating score in the SAW method is divided into two stages:

Step 1: creating the decision matrix: The first step is creating the decision matrix that represents the available versions and relevance criteria.

Step 2: Creating the normalized decision matrix: This step is to normalize the weights of criteria in order to obtain values between 0 and 1. As the SAW method does not automatically generate the weights of criteria, we integrate the entropy method to weight the criteria within SAW. The entropy method is an objective technique for weighting criteria[12] the idea is that a criterion j is more important

than the dispersion of valuations of shares is important. Thus, the most important criteria are those that discriminate most between actions: the more a test can distinguish between the scores, the more important.

Step 1: The entropy of a test j is calculated as follows:

$$K\sum_{i=0}^{m} aij \log(aij) \quad (2)$$

aij: score of candidate i under criterion j

i: i version

K: is a constant

It must go through a standardization process of the scores of criteria before calculating overall score for each release.

Step 2: The entropy E is even larger than the values "Ej" are close. Thus, the weights will be calculated according to the measure of dispersion (the opposite of entropy):

$$Dj = 1 - Ej \quad (3)$$

Step 3: the weights are then standardized using the following formula:

$$Wj = \frac{Dj}{\sum_j Dj} \quad (4)$$

Thus, our approach to calculating relevance of versions is based on the following steps:

Step 1: Establish a list of criteria.

Step 2: Generate a table (matrix) of evaluation; the table will contain all values of the scenarios following each test so that the lines represent the versions and the columns represent the criteria.

Step 3: Using the weighting methods (Entropy) and aggregation (SAW) on the scorecard to achieve a standardized table and weighted.

Step 4: Classification of versions, this classification is made by calculating an overall score for each version of the following criteria.

Note: Based on various criteria, one can have stable criteria and other changes in their weight. Moreover, one can distinguish between the dominant criteria and other less dominant. In our work we use the method jointly SAW and the entropy method to determine the overall score of a standardized version of ontology. We also restrict, at the SAW method, the range of values given to the weights of criteria of relevance to the interval [0, 3]: 0 is the minimum weight for a very low value and 3 being the most relevant. This approach dubbed: Advanced Simple Additive Weithing (ASAW).

Consistology prototype

To begin the process of evolution, the loading of the ontology is essential. The loading of the ontology is done with the components and OWL Data Factory OWL Ontology of OWLAPI. After loading the ontology, the user will be prompted to load the metadata (the one corresponding to the ontology). After loading the ontology and metadata the user can start the process of ontology evolution. The interrogation and manipulation of the latter are insured through OWLAPI defines several methods and objects. The expert may express its views on all criteria. We offer a range of weights for each criterion that varies between 0 and 3. The maximum number of version is set at 20 (Figure 1). The introduction of values, the expert clicks the calculate button score, the calculation process Asawa begins and the result is displayed in the window. We also implemented a graphical editor (Figure 2) which displaysall the versions created for each addition or deletion of a version. The developed tool provides access to the XML file of versions (Figure 3). This file contains the score of relevance criteria and the date of creation of each version (Figure 4). In addition, the expert can see the file 'version criteria.txt' that contains the history of all versions (type of changes and creation date of each version).

Please give a weight for each criterion: interval between 0 and 3 (0: Low, 1: Medium , 2: Good 3: High)

Type of change: Remove_Concept

Date and Time 05/05/2011 08:51:01

VERSION	Conceptualization	Use frequency	Completeness	Abstraction	Size of the ontology
Version noyau Create...	3	3	3	3	3
Version1 Created on: ...	3	1	2	3	3
Version2Created on: ...	3	2	3	2	3

Figure 1 Interface weighting of relevance criteria.

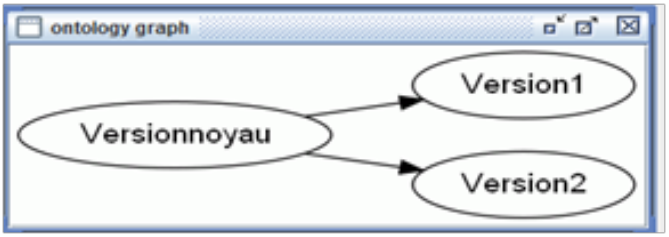

Figure 2 Versions graph.

```
version.xml *
 1  <versions>
 2  <V>
 3  Version noyau   Created on:  21/12/2010
 4  </V>
 5  <conceptualization>3</conceptualization>
 6  <frequency>3</frequency>
 7  <completeness>3</completeness>
 8  <abstraction>3</abstraction>
 9  <size>3</size>
10  <V>
11  Version1Created on:  28/01/2011
12  </V>
13  <conceptualization>3</conceptualization>
14  <frequency>1</frequency>
15  <completeness>2</completeness>
16  <abstraction>3</abstraction>
17  <size>3</size>
18  <V>
19  Version2Created on:  28/01/2011
20  </V>
21  <conceptualization>3</conceptualization>
22  <frequency>2</frequency>
23  <completeness>3</completeness>
24  <abstraction>2</abstraction>
25  <size>3</size>
26  </versions>
```

Figure 3 XML file of each ontology version.

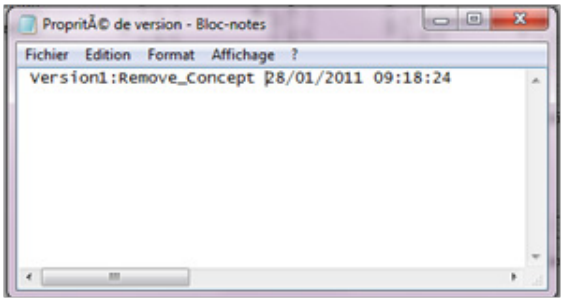

Figure 4 Visualization of the relevance score for each ontology version.

Conclusion

Supporting ontology variation becomes crucial and exceedingly essential, generally when using ontologies in dynamic environments. A main problem when dealing with ontology variation is the management of quite a lot of versions. Ontology versioning is a complex issue as it should take into account the management change, storage of versions and consistency issues, etc. The main purpose of this study is to propose an approach, tool for ontology versioning and to discuss the problem of the ontology versioning and optimization storage. It presents an optimization method dubbed Asawa. We have developed an extension for the Consistology tool: Asawa and the inverse operators of changes in the OWL. We seek in a future work to propose a query language of multi-versions, apply the approach on a geographical ontology. As future works to be done, we propose a medical ontology, as an extension of my previous work.[13–15]

Acknowledgments

None.

Conflicts of interest

None.

References

1 Jaziri W, Sassi N, Gargouri F. Approach and tool to evolve ontology and maintain its coherence. *International Journal of Metadata Semantics and Ontologies*. 2010;5(2):151–166.

2 Stojanovic L. *Methods and Tools for Ontology Evolution*. 2004. p. 1–249.

3 Klein M, Noy N. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*. 2003;6(4):428–440.

4 Eder J, Koncilia C. Changes in Ontologies. *OTM Workshops*. 2004. p. 662–673.

5 Luong P. *Gestion de l'évolution d'un web sémantique d'entreprise*. 2007.

6 Rogozan D. *Gestion de l'évolution des ontologies : méthodes et outils pour un référencement sémantique évolutif fondé sur une analyse des changements entre versions d'ontologie, l'université du Québec à Montréal*. 2008.

7 Plessers P. *An Approach to Web–based Ontology Evolution*. 2006. p. 1–238.

8 *Wetenschappen Web & Information Systems Engineering*. Free University of Brussels, Belgium.

9 Flouris G. *On Belief Change and Ontology Evolution*. 2006. p. 1–192.

10 Mizoguchi R. A step towards ontological engineering. *12th National Conference on AI of JSAI*. 1998. p. 24–31.

11 Uschold M, Gruninger M. *Ontologies: Principles, methods and applications*. 1996. p. 1–69.

12 Entropie de Shannon.

13 Sassi N, Jemal H, Jaziri W. *Optimisation du stockage des versions par calcul de pertinence : la méthode ASAW. 4èmes Journées Francophones sur les Ontologies*. 2011. p. 22–23.

14 Jemal H, Kechaou Z, Ben Ayed M. An enhanced healthcare system in mobile cloud computing environment. *Vietnam Journal Computer Sciences*. 2016;3(4):267–277.

15 Jemal H, Kechaou Z, Ben Ayed M. Agent Technology Based Modelization Systems for Healthcare. *IETE Journal of Research*. 2017;63(5):1–14.