Ontology Evolution

Authors: Raul Palma, Peter Haase

Motivation

Ontologies are dynamic entities that evolve over time. For instance, domains are not static or fixed, and the conceptualization or the formal specification of an ontology may change. The management of ontology dynamics has several challenges associated, ranging from the adequate control of ontology changes to the administration of ontology versions.

What is Ontology Evolution?

Ontology Evolution

Definition

Ontologies evolution refers to the activity of facilitating the modification of an ontology by preserving its consistency; it can be seen as a consequence of different activities during the development of the ontology.

Goal

The goal of ontology evolution is to provide a defined process (potentially with tool support) to perform updates and changes to one or multiple ontologies.

Input

An ontology in a consistent state.

Output

An ontology in a consistent state with the proposed changes implemented.

Who

All ontology engineers that have to perform changes/updates to a deployed ontology.

When

Normally it occurs after the ontology has been deployed and needs to be updated/changed. Changes during the initial creation would be part of the ontology engineering process.

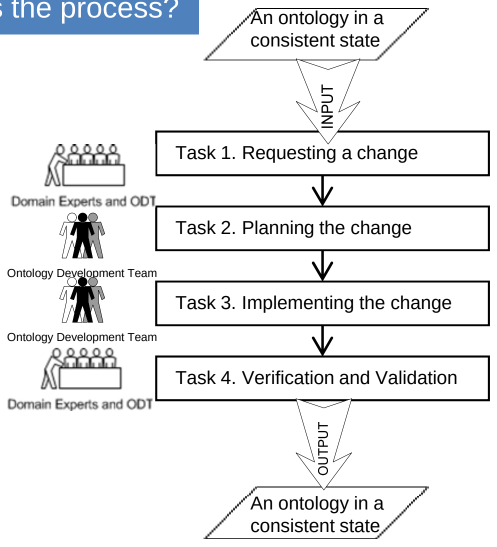
Ontology Evolution or Ontology Versioning?

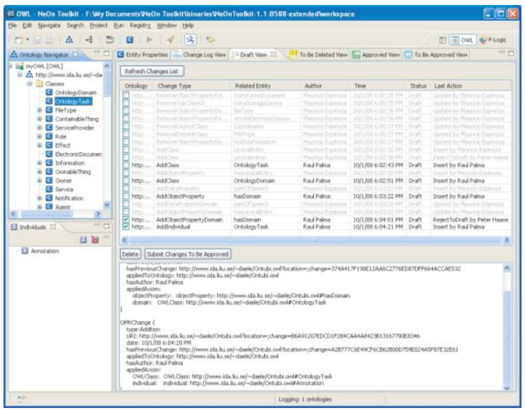
Both ontology evolution and ontology versioning deal with the management of ontology changes. However, they differ in their focus: ontology evolution focus on the modification of an ontology by preserving its consistency, whereas ontology versioning focus on creating and managing different versions of the ontology.

Approach

We propose an ontology-driven approach for the management of ontology changes in distributed and collaborative environments [1,2,3]. At a higher-level, an ontology metadata model allows determining whether an ontology has changed and provides a general overview of how it has changed. At a lower level, a change ontology captures the specific changes that were applied to an ontology. Moreover, a workflow ontology allows representing how changes were applied during the collaborative process, e.g., the policy used for proposals and approvals.

What is the process?


An ontology in a consistent state
INPUT
Task 1. Requesting a change
Domain Experts and ODT
Task 2. Planning the change
Ontology Development Team
Task 3. Implementing the change
Ontology Development Team
Task 4. Verification and Validation
Domain Experts and ODT
OUTPUT
An ontology in a consistent state



Ontology	Change Type	Related Entity	Author	Time	Status	Last Action
http:...	RemoveObjectPropertyRa...	transformDocument	Mauricio Espinoza	10/1/08 6:00:35 PM	Draft	Update by Mauricio Espinoza
http:...	RemoveSubClassOf	DataStorageDevice	Mauricio Espinoza	10/1/08 6:00:35 PM	Draft	Update by Mauricio Espinoza
http:...	RemoveObjectPropertyDo...	FileType	Mauricio Espinoza	10/1/08 6:00:36 PM	Draft	Update by Mauricio Espinoza
http:...	RemoveObjectPropertyRa...	storesElectronicDocum...	Mauricio Espinoza	10/1/08 6:00:36 PM	Draft	Update by Mauricio Espinoza
http:...	RemoveDisjointClass	Coordinates	Mauricio Espinoza	10/1/08 6:00:37 PM	Draft	Update by Mauricio Espinoza
http:...	RemoveDisjointClass	FileType	Mauricio Espinoza	10/1/08 6:00:37 PM	Draft	Update by Mauricio Espinoza
http:...	RemoveObjectPropertyDo...	holdsInformation	Mauricio Espinoza	10/1/08 6:00:37 PM	Draft	Update by Mauricio Espinoza
http:...	AddClass	LexicalEntry	Mauricio Espinoza	10/1/08 6:02:32 PM	Draft	Insert by Mauricio Espinoza
http:...	AddClass	Lexicalization	Mauricio Espinoza	10/1/08 6:02:38 PM	Draft	RejectToDraft by Peter Haase
http:...	AddClass	OntologyTask	Raul Palma	10/1/08 6:02:43 PM	Draft	Insert by Raul Palma
http:...	AddObjectProperty	hasLexicalEntry	Mauricio Espinoza	10/1/08 6:02:50 PM	Draft	Insert by Mauricio Espinoza
http:...	AddClass	OntologyDomain	Raul Palma	10/1/08 6:02:51 PM	Draft	Insert by Raul Palma
http:...	AddDataProperty	partOfSpeech	Mauricio Espinoza	10/1/08 6:03:03 PM	Draft	Insert by Mauricio Espinoza
http:...	AddObjectProperty	hasDomain	Raul Palma	10/1/08 6:03:22 PM	Draft	Insert by Raul Palma
http:...	AddDataPropertyDomain	partOfSpeech	Mauricio Espinoza	10/1/08 6:03:28 PM	Draft	Update by Mauricio Espinoza
http:...	AddObjectPropertyDomain	hasLexicalEntry	Mauricio Espinoza	10/1/08 6:03:52 PM	Draft	Update by Mauricio Espinoza
✓ http:...	AddObjectPropertyDomain	hasDomain	Raul Palma	10/1/08 6:04:01 PM	Draft	RejectToDraft by Peter Haase
✓ http:...	AddIndividual	OntologyTask	Raul Palma	10/1/08 6:04:21 PM	Draft	Insert by Raul Palma

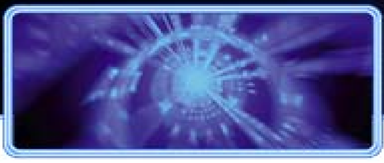
```
  hasPreviousChange: http://www.ida.liu.se/~daele/Ontubi.owl?location=;change=374A417F190E12AA6C2776ED87DFF664ACCAE532
  appliedToOntology: http://www.ida.liu.se/~daele/Ontubi.owl
  hasAuthor: Raul Palma
  appliedAxiom:
    objectProperty:  objectProperty: http://www.ida.liu.se/~daele/Ontubi.owl#hasDomain
    domain:   OWLClass: http://www.ida.liu.se/~daele/Ontubi.owl#OntologyTask
}

OWLChange {
  type:Addition
  URI: http://www.ida.liu.se/~daele/Ontubi.owl?location=;change=B6A91207EDCD1F284CAA4A8423B13167790E8346
  date: 10/1/08 6:04:20 PM
  hasPreviousChange: http://www.ida.liu.se/~daele/Ontubi.owl?location=;change=A28777C6E49CF6CB62B00D759E024A5F87E32E61
  appliedToOntology: http://www.ida.liu.se/~daele/Ontubi.owl
  hasAuthor: Raul Palma
  appliedAxiom:
    OWLClass:   OWLClass: http://www.ida.liu.se/~daele/Ontubi.owl#OntologyTask
    individual:   individual: http://www.ida.liu.se/~daele/Ontubi.owl#Annotation
```

Networked Ontologies

It is worth mentioning that the process in the figure can be applied to networked ontologies since such a process takes into account the existing ontology dependencies with other related artifacts, such as instances, mappings, applications and metadata. In a nutshell, those dependencies are considered during the analysis of the impact and cost in task 2. Furthermore, during the propagation of the changes in task 3, all the ontology related artifacts are updated (if necessary), ensuring the consistency of the networked ontologies. Finally, when assessing the correctness of the evolved ontology in task 4, the verification also takes into consideration the ontology related artifacts to ensure that the whole network of ontologies is behaving as expected (i.e., it is consistent). So, any conflict that may arise can be caught at an earlier stage and this can affect, for instance, the decision of whether a change should be implemented.

NeOn
IST-2005-027595
NeOn-project.org



Ontology Evolution

Tool support in the NeOn Toolkit

Task 1. Requesting a change. *RADON plug-in* is an ontology diagnosis and repair tool that can be used before starting the evolution activity, i.e., before applying changes. The *Workflow Feature* supports the process that coordinates the proposal of changes in a collaborative environment. It supports a top-down/explicit discovery method, i.e., when changes are requested by users/developers. *Evolva plug-in* supports the discovery of changes from external data sources (e.g. text or folksonomies). It supports a bottom-up/implicit discovery method, i.e., when changes are discovered using machine learning techniques.

Task 2. Planning the change. The *NeOn Toolkit* provides simple support when deciding whether to make a change or not. In particular, when a user wants to delete an ontology element, the list of related axioms (the side effect) is shown to the editor, which permits him to verify the cost of implementing the change.

Task 3. Implementing the change. *NeOn Toolkit Ontology Editor* allows the manual application of changes to ontologies.

The *Change Capturing plug-in* supports the logging of changes automatically from the NeOn Ontology Editor. It also supports the application of logs generated by other systems. Additionally ,it is also in charge of propagating changes to the distributed copies of the same ontology. *RADON plug-in* can be used for the management of inconsistencies.

Task 4. Verification and validation. The *Cicero plug-in* supports the justification of changes. The *Workflow Feature* supports the refining of activities.

Example

Experiment conducted with a team of FAO ontology editors in charge of the maintenance of ontologies in the fishery domain. The editors performed collaboratively a set of typical changes and actions to a stable version of a fishery ontology in order to reach a new stable version. In this scenario a central server kept a shared copy of the ontology and the related changes [3].

Task 1. Requesting a change. Initially, FAO experts requested a set of changes to be applied to the current version of the species ontology (Available at http://www.fao.org/aims/neon.jsp). That is, changes were discovered through a top-down/explicit method. Ontology editors were working collaboratively in the implementation of the changes and hence it was not necessary to prioritize such changes(prioritization of multiple changes). The changes were formally represented as individuals of the change ontology (representation of changes). Additionally, since ontology editors were following a well defined process (i.e. workflow) for the coordination of the change proposals, they used the NeOn Toolkit together with the collaborative infrastructure. Consequently, during this task and for every change proposal, the appropriate workflow action (i.e., insert, update, delete) was also formally represented as an individual of the workflow ontology.

Task 2. Planning the change. For this experiment, it was necessary to implement the changes requested regardless of the side effects. Therefore, no analysis of the impact or cost was performed. In fact, the idea of the experiment was to assess the efficiency of the system to support the development of an ontology in a collaborative scenario and not the time or cost of implementing a change.

Task 3. Implementing the change. For this task, it was not necessary any restructuring of the change(s) because, on the one hand ,the changes were not too difficult to implement due to the ontology structure and ,on the other hand, the cost of implementing was not an issue. Additionally, the system (change capturing plug-in) took care of logging automatically the changes proposed (change logging), maintaining the chronological history of the events.

In this experiment, the change(s) did not introduce any inconsistencies in the ontology. However, in case it were necessary to manage inconsistencies, the RADON plug-in, could be used to detect and fix them. As aforementioned, the ontology and related changes were centralized in a server. Furthermore, the ontology used for the experiment was not related to other artifacts at that moment. Hence, it was not necessary any propagation of changes.

Task 4. Verification and validation. During this task, the ontology editors analyzed every change to ensure that the resulting ontology was as expected. To this end, they used the appropriate views of the NeOn Toolkit and the collaborative infrastructure. Additionally, this task was one of the most important of the experiment as it included all the refining activities derived from the workflow that coordinates the proposal of changes. Hence, an ontology validator was in charge of accepting or rejecting changes when necessary by using the appropriate interfaces of the Workflow Feature. Finally, at the moment of the experiment, no support for the justification of changes was used.

Additional information:

Contact person: rpalma@man.poznan.pl

- Palma, R.; Haase, P.; Corcho, O.; Gómez-Pérez, A.; Ji, Q. An Editorial Workflow Approach For Collaborative Ontology Development. 3rd Asian Semantic Web Conference. ASWC 08. Bangkok, Thailand, December 2008. Springer. [1]
- Palma, R.; Haase, P.; Corcho, O.;, Gómez-Pérez, A. Change Representation For OWL2 Ontologies. Proceedings of the Sixth OWLED Workshop, collocated with the ISWC-2009, Washington, USA, October 23-24, 2009. CEUR-WS.org. [2]
- "D1.3.2. Change management to support collaborative workflows ". NeOn Deliverable. December 2008. [3]
- "D5.4.2. Revision and Extension of the NeOn Methodology for Building Contextualized Ontology Networks". NeOn Deliverable. February, 2009. [4]