Change Tracer: A Protégé Plug-In for Ontology Recovery and Visualization*

Asad Masood Khattak[1], Khalid Latif[2], Zeeshan Pervez[1], Iram Fatima[1], Sungyoung Lee[1], and Young-Koo Lee[1]

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

Abstract. We propose to demonstrate the process of capturing and logging ontology changes and use these logged changes for ontology recovery and visualization. Change Tracer, a Protégé plug-in we have developed for ontology recovery from one state to another (including roll-back and roll-forward) and visualization of ontology changes are structured as RDF graph. Users can also navigate through the history of ontology changes.

1 Demonstration

Knowledge representation and visualization is a collaborative process while dealing with information of high dimensions and complex nature represented in ontology. The goal is to demonstrate our Protégé plug-in i.e., *Change Tracer*, that capture and log changes (happening to these complex ontologies) in a repository i.e., Change History Log (CHL) with conformance to the Change History Ontology (CHO) [1]. On top of these logged changes, applications like ontology change management, ontology recovery, change traceability, and to some extent navigation and visualization of changes and change effects [1, 2] are implemented. The plug-in consist of two main modules;

Recovery module is responsible for rolling back and forwards the applied changes on model in reverse and forward manner for ontology recovery. SPARQL queries are extensively used for the purpose of recovery (see Figure 1-a). For validation of our plug-in (*Change Tracer*), we have checked its change detection accuracy against the *ChangesTab* of Protégé and our plug-in showed good results. Accuracy of high percentage for roll-back and roll-forward algorithm is observed [2].

Visualization module has two sub modules; a) To visualize the ontology and ontology changes in graph like structure. The *TouchGraph* API has been extended for graph drawing. Resources, such as classes, are depicted as nodes and properties as edges. Different resources are represented in different color with respect to one another but in uniform color together. An instance is represented in blue color and expressed in its complete relationships with its associated resources. Numbers of filters are supported in our graph view such as zooming in and out of the graph for detail

* This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under IT/SW Creative research program supervised by the NIPA(National IT Industry Promotion Agency)" (NIPA-2010-(C1820-1001-0001)).

X. Du et al. (Eds.): APWeb 2011, LNCS 6612, pp. 386–387, 2011.

view and Fish-eye view effects. A modified version of the *Spring* graph drawing algorithm is implemented in the visualization that ensures esthetically good looking graph structured and well separated nodes. We have also provided a search facility in case if the ontology is too large to find a resource. The graph and each resource in the graph are also drag-able and the graph refreshes itself.

The second sub module; b) Visually navigate through the history of ontology changes with roll-back and roll-forward operations (see Figure 1-b). Appropriate information capturing is very important for accurate recovery. We implemented different listeners (i.e., *KnowledgeBaseListener*, *ClsListener*, *SlotListener*, *FacetListener*, and *InstanceListener*) that actively listen to ontology changes during ontology engineering. For recovery purpose, first the required *ChangeSet* instance is extracted from the log and then all its corresponding changes are extracted. We used the concept of inverse and reverse changes. First, all the changes are converted to their inverse changes (e.g., *RangeAddition* to *RangeDeletion*) and then implemented in reverse sequence of their occurring order on the current version of ontology to get it in previous state. We have provided the playback and play-forward features where not only the ontology but the changes could also be visually navigated and the trend could be analyzed. Starting from the very first version of the ontology, the user can play the ontology changes and visualize as well as visually navigate the ontology and ontology changes. Rest of details on recovery, visualization, and their implementations is available in [2].

a

```
SELECT   ?changes   ?timeStamp
WHERE
{
?changes   docLog:isPartOf   changeSetInstance .
?changes   docLog:hasTimeStamp   ?timeStamp
}
ORDER BY DESC(?timeStamp)


SELECT   ?changedTarget   ?isSubClassOf
WHERE
{
Resource   docLog:hasChangedTarget
?changedTarget .
Resource   docLog:isSubClassOf   ?isSubClassOf
}
```

b

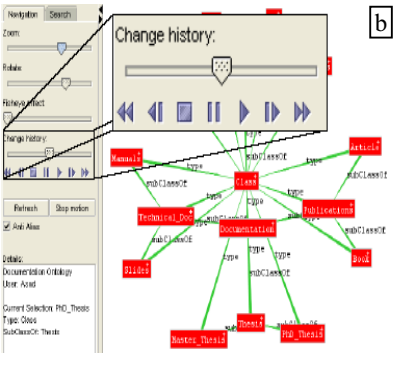

Fig. 1. a) Shows the queries used to extract the changes. b) Visualization of ontology with history playback feature. Users can visually navigate through the history of ontology changes.

The plan is to demonstrate the plug-in features like change capturing, change logging, ontology recovery, ontology visualization, and visual navigation through the history of ontology changes of our developed plug-in.

References

[1] 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)

[2] Khattak, A.M., Latif, K., Lee, S.Y., Lee, Y.K., Han, M., Kim Il, H.: Change Tracer: Tracking Changes in Web Ontologies. In: 21st IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Newark, USA (November 2009)