A Collaborative Ontology Development and Service Framework ONKI

Ville Komulainen, Arttu Valo, and Eero Hyvönen
Semantic Computing Research Group
University of Helsinki and Helsinki University of Technology
P.O. Box 5500, 02015 TKK, FINLAND
http://www.cs.helsinki.fi/group/seco/, firstname.lastname@cs.helsinki.fi

ABSTRACT

We present a national ontology library development and service framework ONKI under development in Finland. The idea behind the system is to start and support a national collaborative effort for developing mutually interoperable ontologies for the Semantic Web. ONKI supports collaborative development of interdependent ontologies and their usage as a web service in applications such as library or museum cataloging systems and in information retrieval systems for formulating meaningful concept-based queries.

1. THE GOALS

The Semantic Web will not emerge without publicly available ontologies that can be made mutually interoperable. Furthermore, services facilitating their usage in semantic web applications are needed. This paper presents an ontology library service ONKI for satisfying these needs. The goals for developing the ONKI system are: 1) Enable distributed development of multiple interdependent vocabularies in machine-readable and -usable form. 2) Provide support mechanisms for versioning, maintenance, and distribution of ontologies. 3) Provide a semantic browser for searching and utilizing the information within the system. 4) Provide the access services of ontological information as machine-usable web services interfaces for external applications.

In the following we first describe how ONKI supports ontology development. After this ONKI browser service for using the ontologies in applications is described.

2. ONTOLOGY DEVELOPMENT PROCESS

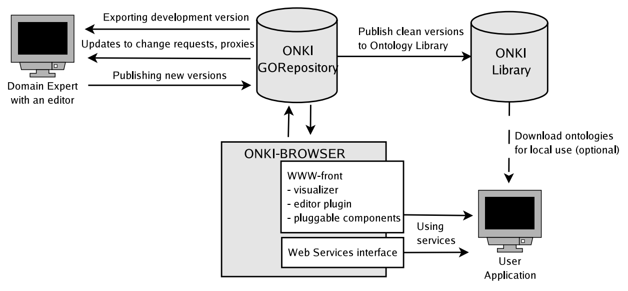

Figure 1: ONKI Architecture and publishing process.

The ONKI architecture and publishing process is depicted in figure 1. Three core components of the system are the development repository (ONKI GORepository) for ontologies being edited, the public ontology library containing the set of published interrelated ontologies (ONKI Library), and the browsing service (ONKI Browser) for using the ontologies.

ONKI separates the development process into two major parts: the *development loop* (cf. the arrows between Domain Expert and ONKI GORepository in figure 1) and the *publishing push* (cf. the bottom arrow from Domain Expert to ONKI GORepository and the arrow from there to ONKI Library in figure 1).

The development loop is based on exporting an ontology with versioning metadata and then occasionally polling the development repository for seeing there have been affecting changes in other ontologies. Pull is used to download the affecting changes made in related ontologies because it has been identified as a better mechanism for keeping distributed ontology copies and development synchronized [2, p. 152].

When editing the ontology, all changes, e.g., the introduction of a new concept in a subsumption hierarchy, are documented as instances of an RDF change ontology and are stored as metadata of the ontology development version. Concepts from related ontologies can be imported by a proxy-mechanism that creates a copy of the borrowed concepts and keeps track of their origin. Having a securely contained development copy of imported concepts allows the ontology developer to focus on her modeling work without constant worries of changes in dependencies with other ontologies in the library.

Once an ontology editor decides that the current development version is mature enough for publishing, the development changes are sent to the development repository. The development repository keeps track on individual concepts and ontologies that contain them. Both concepts and ontologies are versioned.

When publishing an official version of the ontology, the publishing push is automatically activated at the development repository. In publishing push, the development metadata that was created during development is separated from the clean ontology version. This results in two published packages: 1) A cleaned-up, readily-usable ontology — in our case an RDF Schema file. 2) An RDF file containing the change instances from the previous version to this just published latest version. This change metadata can be used when the developers of the other ontologies what to upgrade their own ontologies to meet the changes.

3. DEVELOPMENT REPOSITORY FUNCTIONALITY

The development repository maintains metadata of versions for concepts and ontologies, and tells to what ontologies concepts belong to. The metadata of changes during ontology development is based on two mechanisms: *changes* and *proxies*.

3.1 Describing Changes

There are two kind of change descriptions. First, metadata of concept and ontology changes in the edited ontology is maintained. Second, the system also records *change requests* imposed by changes made in other ontologies. For example, if the original version of a borrowed concept is moved from an ontology to another, then the new origin information should be updated in all ontologies using the concept.

Knowing the change history of ontologies is important in synchronizing ontology development of related ontologies and in keeping the versions interoperable with each other. In PROMPTdiff [1], ontology changes are identified automatically by comparing two versions and then deducing the changes. Since this approach cannot necessarily identify and describe all chances accurately, we decided that the editor should record and explain the changes explicitly during the development process in terms of change metadata.

Change requests differ from changes only in that they have been imposed by changes made in other related ontologies. Pending change requests are visible in the development repository through the ONKI Browser. The editor can then decide whether to turn the request into an official change, to remove the request entirely, or to remain undecided and let the change request lay in the development repository.

3.2 Using Proxies

Proxies are a crucial mechanism for separating individual ontologies for distributed development. A proxy is a local representation of a remote entity, in our case a concept. When an ontology is exported for development, its references to other ontologies within the development repository are replaced by proxies. Thus, instead of referring to external concepts, the references are be made to temporarily inserted proxies.

A proxy can have all or none of the properties of the entity it represents—for development time the editor is free to modify its properties just as those of her own concepts'. If and when the properties of a concept are changed in its own home ontology, polling can be used to update to the latest version. Thus, the developer of an ontology can have a isolated development version and work with it independently.

It is the duty of the development repository import and export functions to retain unique URIs for the concepts during development. Upon publishing, temporary metadata such as proxies are removed, URIs are reverted to point to the actual entities, and official change set is made publicly available with the new version.

4. ONKI BROWSER AND INTERFACES

ONKI Browser is used for illustrating, finding, and importing concepts from the ONKI system ontologies (cf. the arrow Service utilization in figure 1). It consists of three components: 1) Connector to ontology repository that has utilities for knowledge-base information retrieval processes. 2) Visualizer for the semantic data, collecting the data from Connector according to parameters given. 3) Web Service interface for intelligent agents and applications. This interface is as a wrapper to the Connector providing access to ontological information.

An ontology library such as ONKI can contain lots of separate ontologies. A domain expert editing one ontology with an ontology editor, such as Protege-2000, typically can not see the other related ontologies stored in the library. Thus, it is essential to have a tool for gaining a clear perception of the ontology library as a whole. Although visualizing complex semantic data in HTML is challenging, HTML is a good platform since there is no need for any additional software installation or plugins for the end user. Therefore ONKI Browser is implemented by server side programming providing visualization of ontological data for all devices equipped with an ordinary HTML-browser. Interface also offers search engines an option of querying ONKI and look for synonyms or closelely related concepts matching the specified search-criteria and later guide Search engines user towards the answers he/she was really looking for.

Machine understandable interfaces are required for sharing and using knowledge stored in ontologies. External applications can use ONKI by the Simple Object Access Protocol (SOAP) over HTTP. SOAP enables applications to connect to ONKI and execute queries defined by SOAP-methods on the knowledge base. This mechanism gives, for example, the possibility to annotate external application data with the concepts in ONKI by implementing SOAP-calls to the server. ONKI Browser can be used for finding and selecting the annotation concepts. The benefits of such a centralized ontology service are clear: Firstly, external applications can reuse the ONKI Browser functionality. Secondly, concept labels and especially the underlying complex URIs can be imported easily into applications. Thirdly, the service on the web always contains up-to-date versions of the ontologies.

5. DISCUSSION

The need for ontologies is clearly visible, but the tools for the development, management, and publishing ontologies are just being developed. Limited interoperability between knowledge editors and management systems and a generic fragmentation of approaches is still a major obstacle for developing interdependent ontologies. ONKI system tries to alleviate these by providing a non-intrusive framework for distributed collaborative development, straightforward publishing, and web service based usage of ontologies.

Acknowledgements

This work is funded mainly by the National Technology Agency Tekes.

6. REFERENCES

[1] Michel Klein. *Change Management for Distributed Ontologies*. PhD thesis, SIKS, the Dutch Graduate School for Information and Knowledge Systems, 2004.

[2] Ljiljana Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, Universität Karlsruhe, 2004.