Mash-up Ontology Services for the Semantic Web

Kim Viljanen, Eero Hyvönen, Eetu Mäkelä, Osma Suominen and Jouni Tuominen

Semantic Computing Research Group (SeCo), Laboratory of Media Technology, Helsinki University of Technology, and University of Helsinki, Department of computer Science, P.O. Box 5500, 02015 TKK, Finland; http://www.seco.tkk.fi; email: first.last@tkk.fi

We present ONKI ontology server, a ”mash-up” approach for integrating ontology library services with semantic web applications. The idea of ONKI is to provide applications with ready-to-use ontology service functionalities, such as semantic autocompletion, browsing, and annotation support, at the user interface level using AJAX mash-up technologies. The system is being integrated with various semantic web applications.

Problem: Integrating Ontology Services with Applications

The vision of the Semantic Web is strongly based on the idea of using shared ontologies and metadata for achieving interoperability between organizations or individuals. Ontologies are typically shared by downloading them, and each application must separately implement the ontology support. We argue that one should not only share the ontologies but publish the functionalities for using them as services in an easy and cost-effective way. Ontologies also change in time, when new concepts and relations are introduced or removed, leading to versioning and maintenance problems when several interdependent ontologies are being developed at the same time.

In order to support ontology usage in content annotation, information retrieval, and ontology development we are developing a centralized ontology server system called ONKI [HVS05,K07, SVH07] (fig. 1). In this paper we focus on two services provided for the information indexers, semantic autocompletion and ontology browsing for finding easily annotation concepts and for transferring their URIs and other information into the application.

ONKI introduces a ”mash-up” approach, where ontological support is added to an application (e.g., a cataloging system in a museum or library) at the user interface level by ontology aware user-interface components that communicate asynchronously by AJAX or web service technologies with the shared ontology server. The benefit is that service integration can be done easily by changing only the user-interface component. On the client side this means that, e.g., in the case of AJAX and HTML-pages, only a short snippet of code must be added to the web page for adding ontological functionalities.

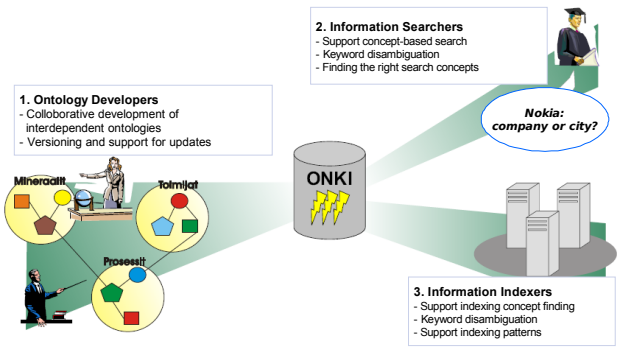

Figure 1. *ONKI for different user groups.*

The ultimate goal of ONKI development is to create a publicly available, national Finnish ontology service that can be easily used to add ontological support to new or existing applications.

ONKI Functionalities

We demonstrate main functionalities of ONKI:

Concept search with concept fetching (fig. 2) is needed when trying to find the concept that matches best the user’s needs and scope. ONKI provides text search with *semantical autocompletion* [HM06] which means that during typing the search string, the system dynamically responds by showing the semantically matching concepts. The text search functionality can be added to any application by using the public ONKI user interface components. When selecting a concept from the result list, the concept's URI and label is fetched to the target application. (Currently implemented as an HTML and AJAX component.)

Concept browsing with concept fetching (fig. 3) can be used both for traversing the ontological hierarchies and between concepts. The browsing feature is implemented as a web application which can be easily linked from any application. If the ONKI browser is used in ”concept fetching” mode, the browser shows a ”Fetch concept” button which, when pressed, transfers the current concept's URI and label to the target application. (Fetch concept functionality currently implemented as an HTML and JavaScript component.)

In addition the ONKI *web service API* contains

methods for finding concepts from the ONKI repository such as search concept by label or URI. The ontologies are also made available as *static files for downloading*. ONKI supports *multilingual ontologies,* has a *multilingual user-interface*, supports loading *multiple ontologies*, and can be *configured* extensively.

Implementation and Evaluation

ONKI is implemented as a Java Servlet application using the Jena semantic web framework for handling RDF content, the Direct Web Remoting (DWR) library for implementing the AJAX functionalities, Dojo javascript toolkit, and Lucene text search engine. Our public test installation is running on an Apache Tomcat, accelerated by Squid web proxy cache, and the public front-end server is Apache (which also serves the static ontology files for downloading).

The testing of the ONKI server in real-world situation is currently beginning in the health promotion domain where the ONKI system is used by several Finnish health promotional organizations for extending their indexing systems with ontological support. The ontologically annotated content is used in the semantical health promotion portal TerveSuomi.fi [SVH07]. First results from the testing will be available by June 2007.

Related Work

Ding and Fensel discuss the need for ontology libraries as an important tool for publishing, managing, re-using, mapping, and versioning ontologies [DF01]. Ding, Finin et al present the semantic web search engine Swoogle that indexes all publicly available semantic web content on the web [DFJ+04]. In our case, the ONKI server is used as a publishing channel for a certain ontology. In addition, ONKI server provides user interface components that can be easily added to existing applications.

Future Work

It should be possible for the end-users to populate the ontology with missing concepts. By adding the concept to the shared ontology server, the once created concept URI could be used by other users as well to avoid (when possible) using different URIs for refering to the same concept. Supporting interlinked ontologies where a domain ontology is mapped to concepts in related other ontologies such as upper ontologies and other domain ontologies.

Acknowledgements

We thank Ville Komulainen for implementing the first version of ONKI [K07]. This work is part of the national semantic web ontology project in Finland (FinnONTO)[1], funded mainly by the National Funding Agency for Technology Innovation (Tekes).

1 http://www.seco.tkk.fi/projects/finnonto/

References

[DFJ+04] Li Ding, Tim Finin et al: Swoogle: A Search and Metadata Engine for the Semantic Web. Proceedings of the Thirteenth ACM Conference on Information and Knowledge Management, November 09, 2004.

[DF01] Ying Ding and Dieter Fensel: Ontology Library Systems: The key to successful Ontology Re-use. In Proceedings of SWWS'01, The First Semantic Web Working Symposium, pp. 93-112, Stanford University, California, USA, July 30-August 1, 2001.

[HM06] Eero Hyvönen and Eetu Mäkelä: Semantic autocompletion. In Proceedings of the first Asia Semantic Web Conference (ASWC 2006), Beijing. Springer-Verlag, New York, August 4-9 2006.

[HVS05] Eero Hyvönen, Arttu Valo, Katri Seppälä, Tomi Kauppinen, Ville Komulainen, Tuukka Ruotsalo, Mirva Salminen and Anu Ylisalmi: Creating a National Content and Service Infrastructure for the Semantic Web. Poster paper, 4th International Semantic Web Conference, Nov, 2005.

[K07] Ville Komulainen: Public Services for Ontology Library Systems. MSc Thesis, University of Helsinki, January, 2007.

[SVH07] Osma Suominen, Kim Viljanen and Eero Hyvönen: User-centric Faceted Search for Semantic Portals. Proceedings of the 4th European Semantic Web Conference ESWC2007, 2007.

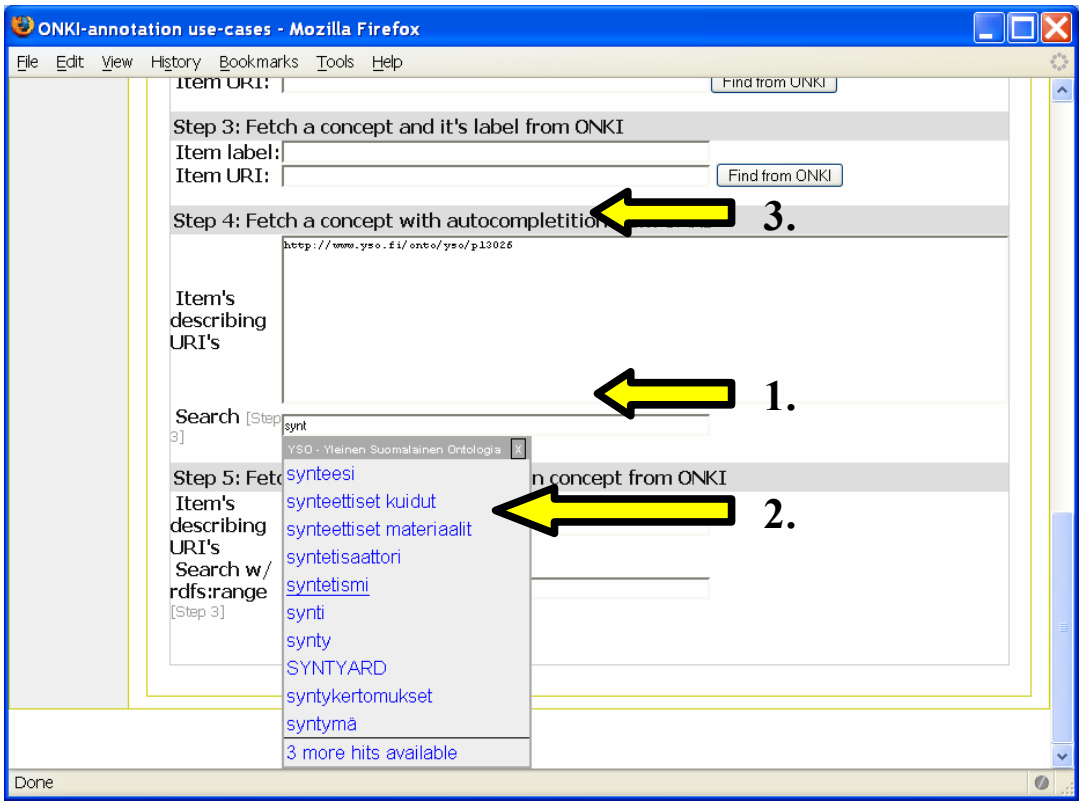

Figure 2: *ONKI semantic autocompletion user-interface component added to an example of an indexing application consisting of HTML form fields. The figure depicts a situation where the user is shown a list of matching concepts (2) to the search string "synt" (1). By clicking on a concept, the URI (optionally more information, such as label) of the concept is returned via Javascript to the target element (3) for further processing.*

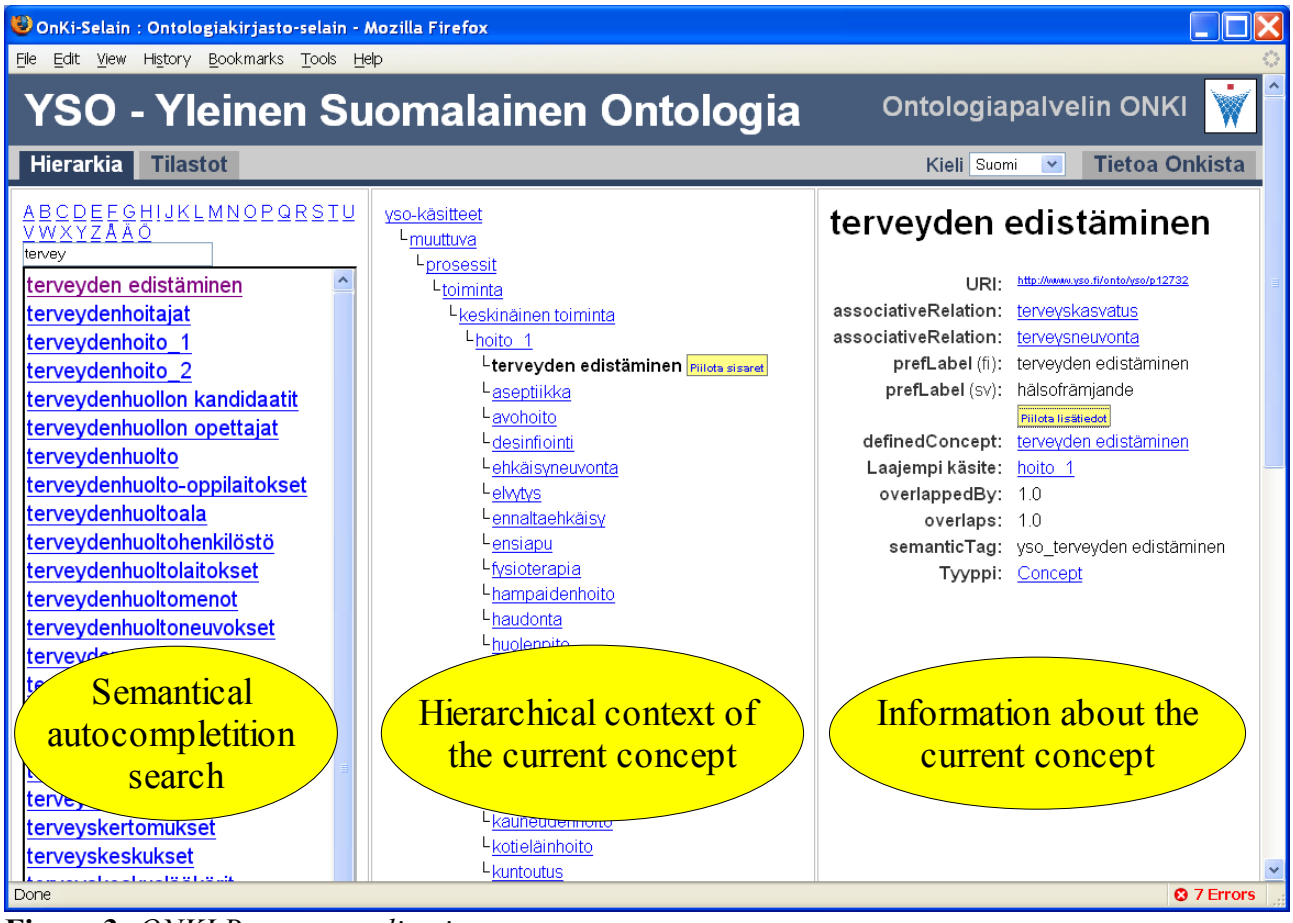

Figure 3: *ONKI Browser application.*