ONKI Ontology Server —Extending Legacy Systems with Ontology Mash-up Services

Kim Viljanen, Jouni Tuominen and Eero Hyvönen
Semantic Computing Research Group (SeCo)
Helsinki University of Technology and University of Helsinki
Laboratory of Media Technology
P.O. Box 5500, 02015 TKK, Finland
http://www.seco.tkk.fi/
first.last@tkk.fi

ABSTRACT

The Semantic Web is based on using shared ontologies for enabling semantically disambiguated data exchange between distributed systems on the web. This requires, from the ontology publisher's viewpoint, efficient means for publishing ontologies on the web to ensure the availability and acceptance of the ontologies. From the ontology user's viewpoint, support services are needed for utilizing ontologies easily and cost-effectively in the users' own systems that are typically legacy systems without ontology support. This paper presents the ONKI ontology server for addressing these vital needs. For the publisher, ONKI provides a server and a Simple Knowledge Organization (SKOS) compatible lightweight ontology browser with ready-made web interfaces for making ontologies available both for human and machine users. For external legacy and other applications, ONKI provides centralized ontology services for semantic disambiguation, concept finding, and concept fetching. A major contribution of ONKI is to provide these services as ready-to-use functionalities for creating "mash-up" applications very cost-efficiently. Two prototypes of the system—ONKI-SKOS for all kinds of ontologies and ONKI-Geo for geographical ontologies with a map mash-up interface—are operational on the web and are currently being successfully used in several pilot applications.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services; J.0 [**Computer Applications**]: General

General Terms

Ontology servers

Keywords

Ontologies, Semantic Web, Annotation, Legacy systems

1. ONTOLOGIES AS WEB 2.0 SERVICES

.

The Semantic Web[1] introduces a metadata layer on top of the World Wide Web infrastructure for describing its content and services in an explicit, machine "understandable" way using ontologies [7, 22, 6]. When such content is available, semantically aware applications for e.g. searching and browsing the distributed content can be created, as demonstrated in e.g. various semantic portals [20, 11, 13]. Lots of ontologies have been created and are available online in RDF(S) and OWL form today. For example, the Swoogle[2] [2] search engine index contains over 10,000 ontologies on the web.

One of the main lessons learned in our work on creating semantic portals [11, 21, 14, 13] is that metadata in data sources, such as museum databases, are often syntactically heterogeneous and contain typos, are semantically ambiguos, and are based on different vocabularies [9]. This results in lots of tedious syntactic correction, semantic disambiguation, and ontology mapping work when making the contents semantically interoperable, and when publishing them on the Semantic Web. A natural solution to this problem would be to enhance legacy cataloging and content management systems (CMS) with ontological annotation functions so that the quality of the original data could be improved and errors fixed in the content creation phase. However, implementing such ontological functions in existing legacy systems may require lots of work and thus be expensive, which creates a severe practical hinder for the proliferation of the Semantic Web.

This relates to the more general challenge of the Semantic Web today: ontologies are typically published as files without much support for using them. The user is typically expected to open the files using some ontology editor, such as Protégé[3], have a closer look of the ontology, and then figure out whether it is of use to her. Once a suitable ontology is found, lots of programming effort is usually needed to utilize the ontology because it is likely to be used in a specific application and software environment. Even if the same ontological resources and structures could be shared by different content providers for interoperability, like in [11], it is not usually possible to share the *functionalities* of using the ontologies across applications. Instead, each application

[1]http://www.w3.org/2001/sw/
[2]http://swoogle.umbc.edu/
[3]http://protege.stanford.edu/

tends to re-implement generic functions for utilizing ontologies, such as semantic autocompletion and disambiguation [10], browsing and finding concepts, populating ontologies with new instances, etc. It is like re-creating map services from the scratch in different geographical web applications, and not utilizing available services such as Google Maps[4], Yahoo Maps[5], or Microsoft Live Search Maps[6].

This paper presents the idea of publishing and utilizing ontologies as lightweight services on the web for mash-up applications. The idea is to make it possible to publish an ontology, represented in standard SKOS[7] or RDF(S)[8] format, easily without practically any programming in an ontology server called ONKI. ONKI provides the human user with facilities for searching and browsing the ontology semantically using an ordinary web browser. At the same time, the services of ONKI are available to external systems on the web as Ajax[9] services for creating mash-up applications, in the same spirit as e.g. Google or Yahoo Maps are used today. One of our ONKI servers, i.e., ONKI-Geo [8] for geo-ontologies, is actually a "second-order" mash-up service since its map visualization is a Google Maps mash-up. ONKI is a key component of a general vision of creating a national, open source Semantic Web infrastructure on a national level in Finland [12].

In the following, we first ouline the notion of ontology servers and discuss related work. The ONKI ontology server and two prototype implementations of it are then presented. After this, application of ONKI services in external application is discussed by presenting two application scenarios. Finally, contributions, results, and lessons learned are summarized, and directions for further research outlined.

2. ONTOLOGY SERVERS

Ontology servers are intended for managing ontologies, providing support for designing, choosing and accessing ontologies [1, 3, 16]. Different types of ontology servers and their functionalities include the following:

Ontology search engines and ontology libraries. To support the finding and comparing ontologies when choosing what ontologies to use in an application, specialised search engines have been proposed [2]. Such systems maintain an index of ontologies available on the web or a centralized library of the ontologies.

Ontology browsers. Perhaps the most obvious type of ontology services are browsers by which an ontology can be published for human users to view. Practically all major classification schemes, vocabularies, and ontologies can be viewed on the web by using some kind of browser facilitating concept finding (search, indices) and visualizations of the concepts' semantic vicinity (broader terms, related concepts etc.). For example, there are browsers for WordNet[10], for the Getty vocabularies Union List of Artist Names (ULAN)[11], Thesaurus of Geographic Names (TGN)[12], and Art and Architecture Thesaurus[13], for medical thesauri and ontologies, such as the Medical Subject Headings (MeSH)[14], etc.

Application development. Semantic web and ontology enabled applications require specialized functionalities including ontological search, browsing and inference services. Examples of such servers and tools include the KAON Server [18], the semantic web framework Jena[15] for Java, and earlier versions of the ONKI Ontology Library Server [16].

Ontology development and maintenance. Ontologies are complex information artifacts. Specialised tools for developing and maintaining them are needed especially when working collaboratively. The most widely used tool in the field today is the Protégé editor, which contains a server version for collaborative ontology development in addition to the traditional stand-alone application. In addition to single ontology developing, the problems of ontology mapping [5] and maintaining consistency within the ontology [1] have been addressed in the ontology server research.

Lightweight vs. heavyweight services. Ontology servers support either lightweight or heavyweight ontologies where lightweight ontologies support only a few ontological core relations, such as subsumption *rdfs:subClassOf*, instantiation *rdf:type*, and part-of relations, and where the ontology server provides only limited support for inference (e.g., transitive closure over class and part-of hierarchies) [1]. The lightweight ontologies are closer to thesauri and may be presented e.g. using SKOS. Heavyweight ontologies may use e.g. description logic based systems such as OWL for defining concepts.

Our work contributes to previous work on ontology servers in the following ways.

- *Mash-up integration support.* In our work, we concentrate on developing ontology server functionalities for runtime usage, especially for annotation and semantic search, which can be easily integrated with legacy and new applications using the mash-up approach.

- *Semantic autocompletion and disambiguation.* Efficient search functionalities are important when trying to find the semantically correct concepts from large ontologies. Text search boosted up with semantic autocompletion and disambiguation functionalities [10] supports the user in finding the right concept by giving constant feedback of the query, and by helping in disambiguating the intended concept meaning.

- *Concept fetching.* When using ontologies in combination with other applications, the idea of "copying" or "transfering" concepts between applications is important. We propose a concept fetching functionality for moving concepts from the ontology server to the target application, such as a legacy cataloguing or CMS system.

[4]http://maps.google.com/
[5]http://maps.yahoo.com/
[6]http://maps.live.com
[7]http://www.w3.org/2004/02/skos/core/
[8]http://www.w3.org/RDF/
[9]http://en.wikipedia.org/wiki/Ajax_(programming)
[10]http://wordnet.princeton.edu/
[11]http://www.getty.edu/research/conducting_research/vocabularies/ulan/
[12]http://www.getty.edu/research/conducting_research/vocabularies/tgn/
[13]http://www.getty.edu/research/conducting_research/vocabularies/aat/
[14]http://www.nlm.nih.gov/mesh/MBrowser.html
[15]http://jena.sourceforge.net

- *Concept collecting.* Usually no single concept describes all the aspects of the entity that is being described with a certain metadata property such as *dc:subject*. Therefore, it should be possible to collect multiple concepts from the ontology server and return these as a combination value in a specific metadata field to the legacy system.

- *Domain-specific user interfaces.* The concepts of an ontology are typically visualized as a graphical tree or graph visualization of the currently selected concept with its semantic vicinity [4]. As a complement to the abstract visualization of ontological concepts, we propose presenting the content also with domain-specific interfaces, such as a map interface for browsing a geographical ontology.

3. ONKI SERVICE FUNCTIONALITIES

The national semantic web infrastructure model being built by the FinnONTO project in Finland [12] argues that ontology services are needed for three major user groups:

1. *Ontology developers* need a collaborative ontology development, versioning, and publishing environment for ontologies [17].

2. *Content indexers* need services for finding the desired annotation concepts and for transporting the corresponding URIs and other data from the ontology service into external applications.

3. *Information searchers* need services for finding and disambiguating keyword meanings, and for transporting the corresponding URIs into search engines and other applications.

The ONKI Ontology Server is used for publishing ontologies as ready-to-use services for humans and machines. It is targeted for content indexers and information searchers to use for concept disambiguation, searching, and fetching.

3.1 ONKI Functionalities

The main functionalities of the ONKI service are 1) a *web widget for concept searching and fetching*, which is described in this section, and 2) *domain-specific ONKI Browsers*. The ONKI Browsers are user interfaces for searching and browsing ontologies, and they can be used independently or accessed via the web widget. Two ONKI Browser implementations are described later on in this paper.

The general idea of the proposed mash-up approach is to provide the developer with a widget that can utilize ONKI services with minimal changes in the legacy system. In the case of an HTML-based legacy system, just a few lines of JavaScript code need to be added on the HTML page. In the case of other user interface technologies, the Web Service[16] interface can be used. The widget solves the problem of getting right URIs into the application or a database; the actual usage of the acquired semantically correct data is in the responsibility of the target application. This kind of a simple way for getting URIs is crucial e.g. in various content creation systems for the semantic web, such as [9, 13].

The ONKI web widget on an HTML form is illustrated in figure 1. The widget enables the user, e.g. a content annotator, to find correct ontological concepts to describe the content to be annotated. The user is provided with searching and browsing capabilities to aid this task. When the correct concept is found, its URI and label can be fetched to the target application. In addition to annotating content, the web widget can be used for supporting other tasks where ontological concepts need to be searched and fetched, such as content searching.

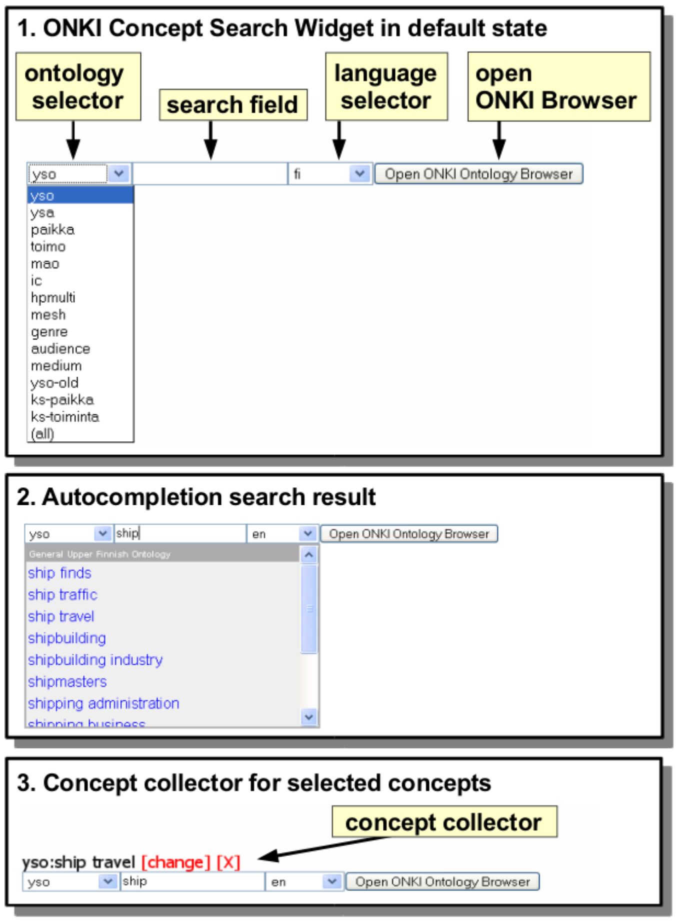


Figure 1: ONKI Concept Search Widget.

Part 1 of figure 1 shows the default components of the widget. The ontology selector can be used to change the ontology used in search. At the moment, there are 14 different vocabularies and ontologies to choose from, including e.g. MeSH, Iconclass[17], the General Finnish Upper Ontology YSO[18] and HPMULTI[19]. The search field is used for inputing the query string. The language of concept labels used in matching the query string can be chosen by using the language selector. The choice of languages depends on the ontology selected. For example, for YSO, English and Swedish is supported in addition to Finnish, and the Finnish Geo-ontology[20] can be used in Finnish, Swedish, and in three

[16]http://www.w3.org/TR/ws-arch/

[17]http://www.iconclass.nl/

[18]http://www.seco.tkk.fi/ontologies/yso/

[19]The European multilingual thesaurus on health promotion in 12 languages http://www.hpmulti.net/

[20]http://www.seco.tkk.fi/ontologies/suo/

dialects of Samish spoken in Lapland. It is possible to use all languages simultaneously by selecting the option “all”. The “Open ONKI Browser” button is used for opening the ONKI browser in a separate window.

The widget supports concept fetching to the target application. This can be done either by semantic autocompletion of the input field, or by pushing the “Open ONKI Browser” button for opening the ONKI Browser:

- *Using semantic autocompletion.* In part 2 of figure 1, the user is typing a search string to the search field of the mash-up component. The system then dynamically performs a query after each input character (here “s-h-i-p-...”) to the ONKI service, which returns the concepts whose labels match the string, given the language selection. The results of the query are shown in the web widget’s result list below the input field. The desired concepts can be selected from the results. When selected, the concept’s URI and label are fetched into the target application. In part 3 of figure 1, the user has selected “ship travel” from the English version of the YSO ontology, and the concept’s URI and label are stored onto the HTML page with the label shown, together with links `change` and `[X]`. In case of a legacy application, which is not capable of handling URIs, only the labels of concepts can be fetched. By clicking the `change` link, it is possible to change the selected concept by using the ONKI browser (whose usage will be illustrated below). The annotation can be removed by clicking s the link `[X]`.
- *Using ONKI Browser.* The alternative for using the autocompletion search, is to use the “Open ONKI Browser” button to search and browse concepts in a new ONKI Browser window. When the desired concept has been found by searching or browsing the ontology, the concept’s URI and label are fetched into the application by pressing the ”Fetch concept” button on the ONKI Browser page corresponding to the concept.

When the URI of a desired annotation concept is fetched, it is stored in a concept collector. The widget provides a default concept collector, but the concept collector can also be implemented in the target application. The default concept collector shows the fetched concepts in the widget’s user interface, and also stores them in hidden input fields. When the form is submitted, the hidden input fields can be processed by the target application. This way the URIs of the annotation concepts can be transfered, e.g., into the database of the application. The body of a HTTP POST request message used to submit the form’s hidden fields to the target application server looks like the following example below where two URIs are selected as values of the “dc:subject” field.

```
dc:subject=http://www.yso.fi/onto/yso/p14629&
dc:subject=http://www.yso.fi/onto/yso/p8038
```

3.2 Implementation

The web widget is implemented as an easily integrable Ajax component. The widget uses HTML and JavaScript for the user interface components, and the Direct Web Remoting (DWR)[21] library for the asynchronous cross-domain Ajax communication between the web browser and the ontology server. The DWR library enables the straightforward use of ontology server’s Java methods in the JavaScript code in the web browser.

[21] http://getahead.org/dwr

An input field in e.g. a cataloging system of a museum could be defined like this:

```
<input id="dc:subject"/>
```

The web widget can be integrated into this example system by adding the following lines of HTML/JavaScript code into the HTML page.

```
1)
<script language="javascript" type="text/javascript"
 src="http://www.yso.fi/onki.js"></script>

2)
<input id="dc:subject"
 onkeyup="onki['yso'].search()"/>
```

The code line 1) is used to load the needed ONKI library files and is typically added to the HEAD section of the HTML page. The code line 2) is added to the BODY section of the HTML page to the locations where the ONKI widget component is desired. The string “yso” in the code line 2) refers to the ontology server instance used in the search.

When a page containing the integration code is accessed, the ONKI JavaScript library files are loaded into the web browser. When loaded, the JavaScript library generates the user interface components of the web widget into the desired locations of the page. In this way, plain input text fields are transformed into ONKI web widgets.

The web widget can be customized e.g. by hiding the ontology or the language selection menus, the search field, or the “Open ONKI Browser” button. The desired customizations can be defined in the integration code[22]. The appearance of the web widget can be modified by CSS rules, e.g. by using the class attributes of the HTML elements of the widget, overriding the default ones.

We have defined an ONKI Java API that has to be implemented by the domain-specific ONKI servers for them to be used with the web widget. The API includes the following methods:

- *search(query, lang, maxHits, type, parent)* - for searching for ontological concepts.
- *getLabel(URI, lang)* - for fetching a label for a given URI in a given language.
- *getAvailableLanguages()* - for querying for supported languages of an ontology.

By implementing the shared API, different domain-specific ONKI servers can be used by one single web widget. The widget can even be used to query multiple ontology servers within the same query.

In addition for the use of the web widget, the ONKI API has been published as a Web Service conforming to the SOAP[23] standard. The Web Service API can be used to

[22] http://www.yso.fi/onki/yso/app/annotation/integration-howto_en.html

[23] http://www.w3.org/TR/soap12-part1/

implement user interface components for applications using other user interface technologies than HTML.

The ontologies in an ONKI service are published also as RDF files to support e.g. semantic web applications that perform complex ontological inferencing. Such applications need access to complete ontology files to be able to process the ontologies in their entirety. When a new version of an ontology is to be published in an ONKI service, the ontology file is tagged with a current date. This date forms part of the URI used for identifying the corresponding version of the ontology file, and this URI can be used as an URL for locating and downloading the RDF file. For every published ontology there is also a static URI which identifies the latest version of the ontology's source file.

4. TWO ONKI SERVERS

Two domain-specific ONKI Servers have been implemented conforming to the general ONKI functionalities described in the previous section. ONKI-SKOS is intended for lightweight ontologies and ONKI-Geo [8] for geo-ontologies. In the following these two systems are shortly described.

4.1 ONKI-SKOS Service

Most of the ontologies developed in the FinnONTO project can be described as lightweight, theusaurus-like ontologies. ONKI-SKOS is an implementation of a general ontology service supporting using thesaurus-like ontologies especially in content indexing. ONKI-SKOS can be used to browse, search and visualize any vocabulary conforming to the SKOS recommendation under preparation at W3C, and RDF(S) ontologies. ONKI-SKOS does simple reasoning (e.g. transitive closure over class and part-of hierarchies). The implementation has been piloted using various ontologies, e.g. MeSH, the General Finnish Upper Ontology YSO and Iconclass.

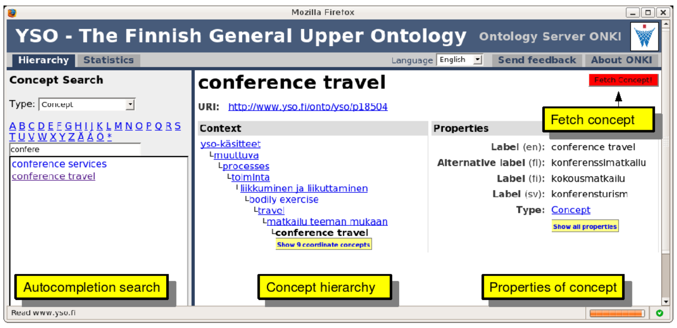
Various configuration properties are specified to enable ONKI-SKOS to process the ontologies as desired. The configurable properties include the ontological properties used in hierarchy generation, the properties used to label the concepts, the concept to be shown in the default view and the default concept type used in restricting the searches.

W3C's SKOS Core[24] is a vocabulary for expressing basic structure and contents of concept schemes, such as thesauri, classification schemes and taxonomies. The concept schemes can be expressed as RDF graphs by using RDFS classes and RDF properties specified in the SKOS Core. SKOS Core defines a suitable model for expressing lightweight ontologies, such as those developed in the FinnONTO project, and therefore ONKI-SKOS is implemented supporting loading of ontologies structured as SKOS concept schemes, with minimal configuration needs. To accomplish this, the SKOS Core structures are configured in ONKI by default. Only the location path of the ontology file to be loaded needs to be configured manually. Because of this strong SKOS binding we call our implementation of the ontology server intended for thesauri and lightweight ontologies ONKI-SKOS.

ONKI-SKOS Browser (see figure 2) is the graphical user interface of the ONKI-SKOS. It consists of three main components: 1) *semantic autocompletion concept search*, 2) *concept hierarchy* and 3) *concept properties*. When typing text to the search field, a query is performed to match the concepts' labels. The result list shows the matching concepts, which can be selected for further examination.

[24]http://www.w3.org/2004/02/skos/core/

When a concept is selected, its concept hierarchy is visualized as a tree structure. ONKI-SKOS Browser supports multi-inheritance of the concepts (i.e. a concept can have multiple parents). Whenever a multi-inheritance structure is met, a new branch is formed to the tree. This leads to cloning of nodes, i.e. a concept can appear multiple times in the hierarchy tree. This increases the overall size of the tree, thus reducing the clarity of the visualization method. The properties of the selected concept are shown in the user interface.



conference travel

URI: http://www.yso.fi/onto/yso/p18504

Context

yso-käsitteet
└muuttuva
 └processes
  └toiminta
   └liikkuminen ja liikuttaminen
    └bodily exercise
     └travel
      └matkailu teeman mukaan
       └**conference travel**

Properties

Label (en):	conference travel
Alternative label (fi):	konferenssimatkailu
Label (fi):	kokousmatkailu
Label (sv):	konferensturism
Type:	Concept

Figure 2: ONKI-SKOS Browser.

When ONKI-SKOS Browser is accessed with no URL parameters, the concept configured to be shown in the default view is selected, and information related to it is shown in the browser. Usually this resource is the root resource of the ontology, if the ontology forms a full-blown tree hierarchy with one single root. In case of a forest of separate smaller subhierarchies, the uppermost concepts of the subhierarchies can be added as children of a virtual root resource. This way all the concepts are in one full-blown hierarchy and can be accessed by browsing the concept hierarchy view. In SKOS concept schemes the root resource is the resource representing the concept scheme itself, i.e. the resource of type *skos:ConceptScheme*.

The name of the ontology, shown in the top of the ONKI-SKOS Browser, is fetched from the ontology being currently browsed. In case of SKOS Core, the name is the label of the resource representing the concept scheme. Similarly, in RDF(S)/OWL ontologies, the name is the label of the resource representing the ontology itself (its URI is the namespace URI of the ontology).

The concept hierarchy of a concept is generated by traversing the configured transitive properties. In SKOS Core these properties are *skos:narrower* and *skos:broader*. The children nodes of the root resource of a concept scheme are the top concepts of the concept scheme. They are defined with property *skos:hasTopConcept*.

Labels of concepts are needed in visualizing search results, concept hierarchies, and related concepts in the concept property view. The label to be attached to a concept is the value of the configured property used for labeling concepts. In SKOS this property is *skos:prefLabel*. The label is of the same language as the currently selected user interface language, if such a label exists. Otherwise any label is used.

The semantic autocompletion search of ONKI-SKOS works by searching for concepts whose labels match the search string. To support this, the labels of the concepts are indexed. The indexed properties can be configured. In SKOS these properties are *skos:prefLabel*, *skos:altLabel* and *skos:hiddenLabel*. When the user searches e.g. with the search term “cat”, all concepts which have one of the forementioned properties with values starting with the string “cat” are shown as the search results. The autocompletion search also supports wildcards, so a search with a string “*cat” returns the concepts which have the string “cat” as a part of their label.

The search can be limited to certain types of concepts only. To accomplish this, the types of the concepts (which are expressed with property *rdf:type*) are indexed. It is also possible to limit the search to a certain subtree of the concept hierarchy by restricting the search to the children of a specific concept. Therefore also the parents of concepts are indexed.

A concept can be selected to be visualized by clicking on a corresponding link in the search results, a concept link in the concept hierarchy, or a concept link shown as a property value in the property views. In a concept property view, all properties selected in the configuration used are shown. These properties consist of RDF statements that have the selected concept as a subject. In SKOS ontologies the property *skos:hiddenLabel* is not shown in the view because it is intended only for free text search operations, not for visual displays of resources.

In several cases, concepts may have some interesting properties, which are not represented as direct properties of the concept, i.e., they are not expressed by simple RDF statements that have the concept as a subject. For example, concepts can be grouped in SKOS Core to collections that are expressed as concepts of the type *skos:Collection*, and the property *skos:member* is used to indicate the members of a collection. Thus, when a concept of the type *skos:Collection* is selected, the members of the collection should be shown in the concept property view. Also, it is beneficial to show the property *skos:member* inversed in the concept property view of a concept that belongs to a collection, so that the associated collection can be seen. Support for showing similar inverse or otherwise indirect properties can be implemented in each separate case with little extra of work by creating a new Java class, which we call a “property fiddler”.

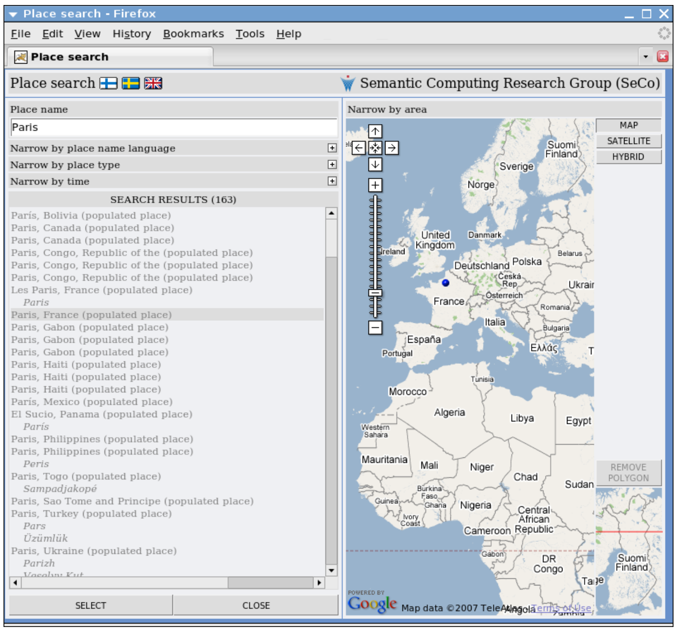
ONKI-SKOS is implemented as a Java Servlet using the Jena Semantic Web Framework[25], the DWR library and the Lucene[26] text search engine.

4.2 ONKI-Geo Service

ONKI-Geo [8] is an ontology service specialized for geographical data. It is based on the Finnish Place Ontology SUO (Suomalainen Paikkaontologia) [15] being developed in FinnONTO. The SUO ontology has currently been populated with 1) place information from the Geographic Names Register (GNR) provided by the National Land Survey of Finland[27] and with 2) place information from the GEOnet Names Server (GNS)[28] maintained by the National Geospatial-Intelligence Agency (NGA) and the U.S. Board on Geographic Names (US BGN). GNR contains about multilingual 800,000 resources of natural and man-made features in Finland, including data such as place type or feature type and the coordinates of a place. The GNS register contains similar information of about 4,100,000 places around the world.

The ONKI-Geo Browser is depicted in figure 3. For visualization of the places, the ONKI-Geo Browser uses Google Maps and its API for mash-ups. The application is useful e.g. for disambiguating homonymous place names: there are e.g. hundreds of places in Finland with the name “Isosaari”. This kind of service is needed when annotating resources using unambiguous place identifiers (URIs), or with coordinate information about arbitrary points or polygons, and when searhing for information about particular places.

ONKI-Geo Browser contains several facets for narrowing the search. One of the facets is the map on which one can draw a polygon that defines the area in which to search. The other facets are an ontology of geographic feature types to search (e.g., lake, city, etc), a list of languages of the place names to search, and a substring of the name of the place. All these facets form a combination of narrowing search criteria by which to search the ONKI-Geo database for place instances.



Place name

Paris

Narrow by place name language

Narrow by place type

Narrow by time

SEARCH RESULTS (163)

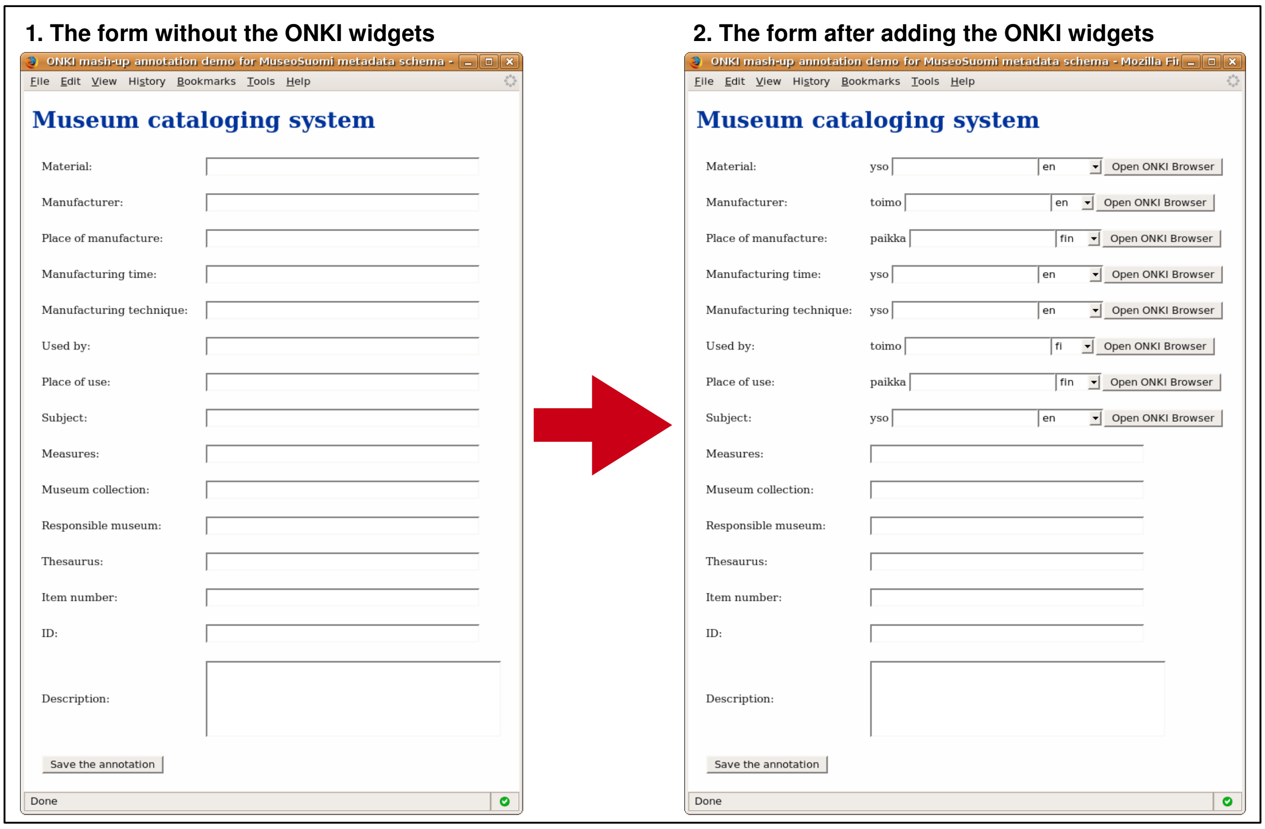
```
París, Bolivia (populated place)
Paris, Canada (populated place)
Paris, Canada (populated place)
Paris, Congo, Republic of the (populated place)
Paris, Congo, Republic of the (populated place)
Paris, Congo, Republic of the (populated place)
Les Paris, France (populated place)
   Paris
Paris, France (populated place)
Paris, Gabon (populated place)
Paris, Gabon (populated place)
Paris, Gabon (populated place)
Paris, Haiti (populated place)
Paris, Haiti (populated place)
Paris, Haiti (populated place)
París, Mexico (populated place)
El Sucio, Panama (populated place)
   París
Paris, Philippines (populated place)
Paris, Philippines (populated place)
   Peris
Paris, Togo (populated place)
   Sampadjakopé
Paris, Sao Tome and Principe (populated place)
Paris, Turkey (populated place)
   Pars
   Üzümlük
Paris, Ukraine (populated place)
   Parizh
```

Figure 3: ONKI-Geo Browser. Search can be constrained by using the facets on the left or by drawing a polygon on the map. By pushing the “Select” button in the left bottom corner, the found concept or selected coordinate information is transferred into the mash-up widget in a legacy application.

In our work, ONKI-Geo Browser was integrated with the ONKI mash-up widget by implementing the DWR interface[29]. By selecting the ontology “paikka” in the ontology selector in figure 1 (and optionally the language), semantic autocompletion can be performed using ONKI-Geo Browser’s autocompletion function. In the same way, push-

[25]http://jena.sourceforge.net/
[26]http://lucene.apache.org/java
[27]http://www.maanmittauslaitos.fi/
[28]http://earth-info.nga.mil/gns/html/

[29]Integration at the ONKI-Geo side was implemented by Robin Lindroos and Tomi Kauppinen.


1. The form without the ONKI widgets
2. The form after adding the ONKI widgets
ONKI mash-up annotation demo for MuseoSuomi metadata schema - Mozilla Fi
File Edit View History Bookmarks Tools Help
Museum cataloging system
Save the annotation
Done

Material:	yso	en	Open ONKI Browser
Manufacturer:	toimo	en	Open ONKI Browser
Place of manufacture:	paikka	fin	Open ONKI Browser
Manufacturing time:	yso	en	Open ONKI Browser
Manufacturing technique:	yso	en	Open ONKI Browser
Used by:	toimo	fi	Open ONKI Browser
Place of use:	paikka	fin	Open ONKI Browser
Subject:	yso	en	Open ONKI Browser
Measures:			
Museum collection:			
Responsible museum:			
Thesaurus:			
Item number:			
ID:			
Description:			

Figure 4: A museum cataloguing system before and after integrating the ONKI widgets.

ing the "Open ONKI Browser" button opens in this case the ONKI Geo-Browser of figure 3 with an additional "Fetch concept" button. The communication between the widget and the geo-ontology server is implemented using the same Ajax techniques as when connecting the widget with the ONKI-SKOS server (figure 2).

5. INTEGRATING ONKI WITH APPLICATION SYSTEMS

In the following we describe use cases of the ONKI system.

5.1 Integrating ONKI with a Cataloging System

To demonstrate how to add ONKI functionalities to a legacy system, we created a simple web form (part 1 of figure 4) presenting the MuseumFinland [11] metadata fields. By adding the ONKI Concept Search Widget to the fields filled with ontological resources (URIs), the possibility of using ontologies in annotating museum collection items is made possible. Part 2 of figure 4 depicts the original form after adding the widgets. The changes can be done very fast, in few minutes.

After this the form can be used for creating semantically annotated metadata. Metadata field values are stored into collectors from where they can be stored into a database. If the form is attached to a legacy system, URIs or terms fetched from ONKI have to either be in existing tables, or the underlying database system has to be modified to handle new kind of information. This depends on the legacy application.

In addition to storing data from a web form, also the functionality of filling up a form based on a database is often needed, for example, when editing an existing metadata record. In such cases, the data from the database has to be read into the collector that can be used by the ONKI widget. In below, we present a more advanced example for this kind of ONKI service usage.

5.2 An Annotation Editor Based on ONKI Ontology Services

To test the ONKI solution, we have integrated ONKI widgets and the mash-up functionalities to the browser-based annotation editor SAHA[30] [19, 23]. SAHA is a generic annotation system supporting distributed collaboration in creating annotations, and hiding the complexity of the annotation schema and the domain ontologies from the annotators. SAHA adapts flexibly to different metadata schemas, which makes it suitable for different applications. Support for using ontologies is based on ONKI ontology services. The system is being tested in various practical semantic portal projects. Figure 5 illustrates the usage of ONKI in SAHA.

The metadata field elements are implemented using the standard ONKI mash-up widget, as discussed above. Depending on the field, different ONKI servers are used as specified in the SAHA configuration in use. In this case the

[30]http://www.seco.tkk.fi/services/saha/

Finnish General Upper Ontology YSO [12], published as an ONKI service[31], is used and has been added as a mash-up component to SAHA for selecting annotation concepts. SAHA can also make use of our automatic text extraction component POKA in extracting potential annotations from web resources [19].

Annotations created with SAHA are stored in a centralized database, from which they can be retrieved for editing or to be used in applications such as semantic portals. It is possible to view and edit existing annotations by reading the metadata fields in corresponding widget collectors. Furthermore, SAHA supports population of its own annotation ontologies by new resources. In this way, different persons creating annotations collaboratively can share new resources created by anyone, e.g., instances of new works of art or other artifacts. ONKI service ontologies themselves cannot be populated with new resources wihout reloading a new version of the ontology by the service provider.

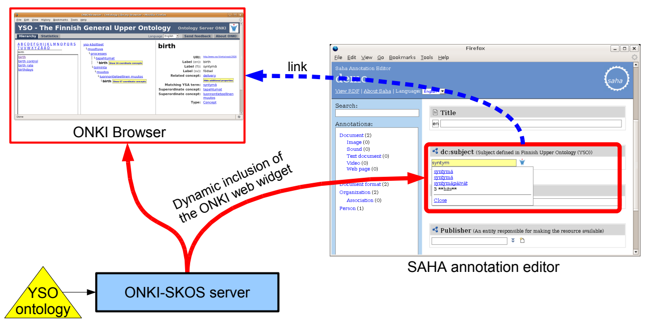


Figure 5: ONKI integrated with the SAHA annotation editor.

5.3 ONKI Concept Fetching for Content Searching

A complementing use-case of concept fetching for annotation purposes is to use the ONKI Concept Search Widget in content searching in legacy systems. The concepts can be selected from ontologies and disambiguated using the Concept Search Widget. After this, the URIs of the chosen concepts can be used in the query to the underlying legacy system. If the content has been annotated using URIs, they can be matched to the query URIs using string matching if each concept has a single URI.

6. DISCUSSION

The main contribution of this paper is to present the idea of publishing *ontologies as mash-up services* that can be integrated in a lightweight fashion to legacy systems on the user interface level. To demonstrate the applicability of the idea, we presented the ONKI service and two implementations of the ONKI interface: the general ontology server ONKI-SKOS and the geographical ontology server ONKI-Geo. The two ONKI implementations also demonstrated the idea of creating domain specific user interfaces to better support the usage of different types of ontologies. A practical contribution of the paper was to introduce the idea of concept fetching between applications and the need for concept collecting when using an ontology server for annotation purposes. Finally, semantical autocompletition was proposed and implemented in the user interface components to provide an efficient method for finding and disambiguating concepts.

A lesson learned from implementing the concept fetching functionality as a web browser application was that special gimmics is needed to transfer data between browser windows loaded from different domains. Despite security is an important concern, we suggest, that browsers should provide some standardized solution for communication between domains.

In the near future, we plan to investigate how the autocompletion concept search component could support showing relations between concepts. This would help the user in disambiguating concept meanings without opening the ONKI Browser. In the current implementation, the mash-up components are created using HTML and JavaScript. However, in some legacy systems also other user interface environments should be supported, e.g., a Java Swing component connected to ONKI using the web service API. Our current focus has been on supporting semantic annotation. However, mash-up ontology support for content search in legacy systems will be researched in more detail. Finally, according to our vision of a national ontology service, the ontologies in such a service should be extensively and mutually interlinked to support creating cross-domain applications. Therefore, the question of how to support developing and using mutually interlinked ontologies is on our research aganda, too.

7. ACKNOWLEDGMENTS

We thank Ville Komulainen for his work on the first version of the ONKI server, and Robin Lindroos and Tomi Kauppinen for collaboration in ONKI-Geo development. Our research is a part of the National Semantic Web Ontology Project in Finland[32] (FinnONTO) 2003–2007 funded by the Finnish Funding Agency for Technology and Innovation (Tekes) and 36 companies and public organizations.

8. REFERENCES

[1] M. N. Ahmad and R. M. Colomb. Managing ontologies: a comparative study of ontology servers. In *ADC '07: Proceedings of the eighteenth conference on Australasian database*, pages 13–22, Darlinghurst, Australia, Australia, 2007. Australian Computer Society, Inc.

[2] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: a search and metadata engine for the semantic web. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659, New York, NY, USA, 2004. ACM Press.

[3] Y. Ding and D. Fensel. Ontology library systems: The key to successful ontology reuse. In *Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, USA*, pages 93–112, 2001.

[4] P. Eklund, N. Roberts, and S. Green. Ontorama: Browsing rdf ontologies using a hyperbolic-style

[31]http://www.yso.fi/onto/yso/

[32]http://www.seco.tkk.fi/projects/finnonto/

browser. In *First International Symposium on Cyber Worlds, CW02, Theory and Practices, IEEE Press.*, pages 405–411, 2002.

[5] J. Euzenat and P. Shvaiko. *Ontology matching.* Springer-Verlag, Berlin, 2007.

[6] D. Fensel. *Ontologies: Silver bullet for knowledge management and electronic commerce (2nd Edition).* Springer-Verlag, 2004.

[7] T. R. Gruber. A translation approach to portable ontology specification. *Knowledge Acquisition*, 5(2):199–220, June 1993.

[8] E. Hyvönen, R. Lindroos, T. Kauppinen, and R. Henriksson. An ontology service for geographical content. In *Poster Proceedings of the 6th International Semantic Web Conference (ISWC/ASWC 2007), Busan, Korea*, Nov 2007.

[9] E. Hyvönen, M. Salminen, S. Kettula, and M. Junnila. A content creation process for the Semantic Web, 2004. Proceeding of OntoLex 2004: Ontologies and Lexical Resources in Distributed Environments, May 29, Lisbon, Portugal.

[10] E. Hyvönen and E. Mäkelä. Semantic autocompletion. In *Proceedings of the first Asia Semantic Web Conference (ASWC 2006), Beijing.* Springer-Verlag, New York, August 4–9 2006.

[11] E. Hyvönen, E. Mäkelä, M. Salminen, A. Valo, K. Viljanen, S. Saarela, M. Junnila, and S. Kettula. Museumfinland—Finnish museums on the semantic web. *Journal of Web Semantics*, 3(2):25, 2005.

[12] E. Hyvönen, K. Viljanen, E. Mäkelä, T. Kauppinen, T. Ruotsalo, O. Valkeapää, K. Seppälä, O. Suominen, O. Alm, R. Lindroos, T. Känsälä, R. Henriksson, M. Frosterus, J. Tuominen, R. Sinkkilä, and J. Kurki. Elements of a national semantic web infrastructure—case study finland on the semantic web (invited paper). In *Proceedings of the First International Semantic Computing Conference (IEEE ICSC 2007), Irvine, California*, September 2007. IEEE Press, forth-coming.

[13] E. Hyvönen, K. Viljanen, and O. Suominen. Healthfinland—Finnish health information on the semantic web. In *Proceedings of the 6th International Semantic Web Conference (ISWC 2007), Busan, Korea.* Springer-Verlag, Nov 2007.

[14] T. Känsälä and E. Hyvönen. A semantic view-based portal utilizing Learning Object Metadata, August 2006. 1st Asian Semantic Web Conference (ASWC2006), Semantic Web Applications and Tools Workshop.

[15] T. Kauppinen, R. Henriksson, J. Väätäinen, C. Deichstetter, and E. Hyvönen. Ontology-based modeling and visualization of cultural spatio-temporal knowledge. In *Developments in Artificial Intelligence and the Semantic Web. Proceedings of the 12th Finnish AI Conference STeP 2006*, October 26–27 2006.

[16] V. Komulainen. Public services for ontology library systems. Master's thesis, University of Helsinki, Department of Computer Science, January 2007.

[17] V. Komulainen, A. Valo, and E. Hyvönen. A tool for collaborative ontology development for the semantic web. In *Proc. of the International Conference on Dublin Core and Metadata Applications (DC 2005)*, Nov 2005.

[18] D. Oberle, R. Volz, B. Motik, and S. Staab. An extensible ontology software environment. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, chapter III, pages 311–333. Springer, 2004.

[19] O. A. Onni Valkeapää and E. Hyvönen. Efficient content creation on the semantic web using metadata schemas with domain ontology services (system description). In *Proceedings of the European Semantic Web Conference ESWC 2007, Innsbruck, Austria.* Springer, June 4–5 2007.

[20] D. Reynolds, P. Shabajee, and S. Cayzer. Semantic Information Portals. In *Proceedings of the 13th International World Wide Web Conference on Alternate track papers & posters*, New York, NY, USA, May 2004. ACM Press.

[21] T. Sidoroff and E. Hyvönen. Semantic e-goverment portals - a case study. In *Proceedings of the ISWC-2005 Workshop Semantic Web Case Studies and Best Practices for eBusiness SWCASE05*, Nov 2005.

[22] S. Staab and R. S. (eds.). *Handbook on ontologies.* Springer-Verlag, 2004.

[23] O. Valkeapää and E. Hyvönen. A browser-based tool for collaborative distributed annotation for the semantic web. In *Proceedings of the Semantic Authoring and Annotation Workshop, 5th International Semantic Web Conference*, November 2006.