Journal of Universal Computer Science, vol. 13, no. 12 (2007), 1835-1853
submitted: 15/4/06, accepted: 31/7/07, appeared: 1/12/07 © *J.UCS*

An Adaptable Framework for Ontology-based Content Creation on the Semantic Web

Onni Valkeapää
(Helsinki University of Technology (TKK), Finland
onni.valkeapaa@gmail.com)

Olli Alm
(University of Helsinki and
Helsinki University of Technology (TKK), Finland
olli.alm@tkk.fi)

Eero Hyvönen
(Helsinki University of Technology (TKK) and
University of Helsinki, Finland
eero.hyvonen@tkk.fi)

Abstract: Creation of rich, ontology-based metadata is one of the major challenges in developing the Semantic Web. Emerging applications utilizing semantic web techniques, such as semantic portals, cannot be realized if there are no proper tools to provide metadata for them. This paper discusses how to make provision of metadata easier and cost-effective by an annotation framework comprising of annotation editor combined with shared ontology services. We have developed an annotation system Saha 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 ontology services, such as concept searching and browsing, concept URI fetching, semantic autocompletion and linguistic concept extraction. The system is being tested in various practical semantic portal projects.

Keywords: Semantic Web, Ontologies, Annotation, Metadata, Information Extraction
Categories: H.3.1, H.3.2, H.3.4

1 Introduction

Currently, much of the information on the Web is described using only natural language, which can be seen as a major obstacle in developing the Semantic Web. Since the annotations describing different resources are one of the key components of the Semantic Web, easy to use and cost-effective ways to create them are needed, and various systems for creating annotations have been developed [Reeve and Han 05, Uren et al. 06]. However, there seems to be a lack of systems that 1) can be easily used by annotators unfamiliar with the technical side of the Semantic Web, and that 2) are able to support distributed creation of semantic metadata based on complex metadata annotation schemas and domain ontologies [Valkeapää and Hyvönen 06, Valkeapää et al. 07].

Metadata descriptions are usually based on ontologies of two kinds. First, an annotation ontology, i.e. a metadata schema, tells what kind of properties and value types should be used in describing a resource. For example, the Dublin Core schema[1] uses 15 elements, such as `dc:title`, `dc:creator`, `dc:subject`, etc. Second, a set of domain ontologies are used to define vocabularies by which the values for metadata properties are given. This suggests that three kinds of tools are needed to address the problems of metadata creation. First, an annotation editor supporting the usage of different metadata schemas is needed. Second, we need services for supporting the usage of the domain ontologies (vocabularies) that are employed for the annotations. Third, tools for automating the creation of actual metadata descriptions in various ways, e.g., for finding suitable values for the elements, must be developed.

To test this idea, we have developed a system of three integrated tools that can be used to efficiently create semantic annotations based on metadata schemas, domain ontology services, and linguistic information extraction. These tools include, at the moment, an annotation editor system Saha[2] [Valkeapää and Hyvönen 06], an ontology service framework Onki[3] [Komulainen et al. 05, Viljanen et al. 07] and an information extraction tool Poka[4] for (semi)automatic annotation. The annotation editor Saha supports collaborative creation of annotations and it can be connected to Onki servers for importing concepts defined in various external domain ontologies. Saha has a browser-based user interface that hides complexity of ontologies from the annotator, and adapts easily to different metadata schemas. The tool is targeted especially for creating metadata about web resources. It is being used in different applications within the National Semantic Web Ontology Project in Finland[5] (FinnONTO) [Hyvönen et al. 07b].

2 Saha Annotation System

2.1 Requirements for the System

In order to support the kind of annotation that is required in our project, we identified the following basic needs for an annotation system. These were also features that we felt were not supported well enough in many of the current annotation platforms:

- **Simplicity**. The system should, as a rule, hide technical concepts related to markup languages and ontologies from its user.
- **Adaptivity**. The system should be adaptable to different annotation cases with different kinds of contents to be described.
- **Quality**. When annotation is done by hand, the annotator should be guided to produce annotations in qualified and pre-defined form, if needed.

[1] http://dublincore.org/
[2] http://www.seco.tkk.fi/services/saha/
[3] http://www.seco.tkk.fi/services/onki/
[4] http://www.seco.tkk.fi/tools/poka/
[5] http://www.seco.tkk.fi/projetcs/finnonto/

- **Collaboration**. The system should support collaborative annotation, where the annotation process can be shared among different annotators at different locations.
- **Portability**. The annotator should be able to use the system at any location without installing any special software.

2.2 Utilizing Annotation Schemas

Ontologies may be used in two different ways in annotation: they can either serve as a description template for annotation construction (annotation schemas/ontologies) or provide an annotator with a vocabulary which can be used in describing resources (reference/domain ontologies) [Schreiber et al. 01]. An annotation schema has an important role in expressing how the ontological concepts used in annotations are related to the resources being described. Without annotation schemas, the role of these concepts would remain ambiguous. In addition to explicitly expressing the relation between a resource and an annotation, the schema helps the annotator to describe resources in a consistent way and it can be effectively used to construct a generic user-interface for the annotation application.

Saha uses an approach similar to the one introduced in [Kettler et al. 05] to form its user interface according to an annotation schema loaded into it. Saha does not use any proprietary schemas, but instead will accept any RDF/OWL-based ontology as a schema. By schemas we mean a collection of classes with a set of properties. An annotation in Saha is an instance of a schema's class that describes some web resource and is being linked to it using the resource's URL (in some cases, URI). We make a distinction between an annotation of a document (e.g. a web page) and a description of some other resource (e.g. a person) that is somehow related to the document being annotated. Accordingly, we can divide classes of a schema to those that describe documents and those that describe some other resources[6]. An annotation schema can be seen as a basis for a local knowledge base (KB) that contains descriptions of different kinds of resources that may or may not exist on the web.

Figure 1 illustrates how an annotation describing a document could be related to different kinds of resources. Properties and classes in the figure are only examples of types of resources that an annotation schema might contain. Saha itself does not define any classes or properties, but instead, they are always expressed by the schema. In figure 1, an annotation is connected to a document using the property `saha:annotates`. The property `dc:subject` points to a class of an external (domain/reference) ontology and the property `dc:creator` to a KB-instance.

Each annotation schema loaded to Saha forms an annotation project, which can have multiple users as annotators. In practice, an annotation project consists of Jena's[7] *ontology model* stored in a database and of *settings* defining how the schema should

[6] We call the classes describing documents *annotation classes* and the classes describing other resources *reference classes*. It should be pointed out, however, that this division is mainly used in order to clarify the way how annotation schemas are designed and utilized. A schema may well be designed so that some or all of its classes are used for describing different types of resources (i.e. documents and other resources). In that case, the division to annotation and reference classes may be less clear or cannot be made at all.

[7] http://jena.sourceforge.net/

be used in the project. The database is used solely for storing the RDF triples of the ontology model. Settings of a project are stored in an RDF file, which is read each time the project is being loaded. An ontology model can be serialized to RDF/XML in order to use the annotations in external applications.

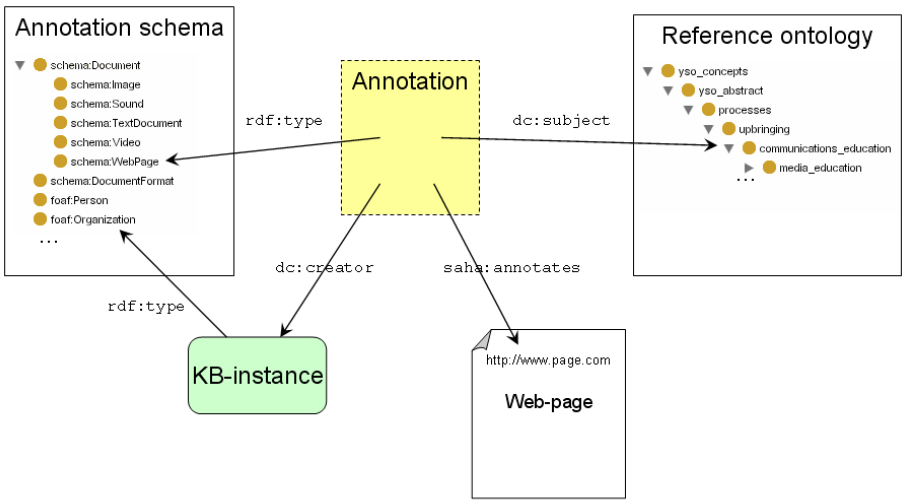

Figure 1: Annotations in Saha

One of the purposes of an annotation schema is to guide an annotator in creating annotations. In order to ease the annotation we prefer using as simple schemas as is practical for each annotation case, containing rather tens than hundreds or thousands of classes and properties. Because of this and due to manual work required in defining settings for an annotation project, Saha is not designed, nor meant to be used with very large annotation schemas. This is opposed to reference ontologies which are used to define vocabularies for the annotations. They are typically more complex and much larger in size when compared to annotation schemas. In order to conveniently utilize these kinds of reference ontologies in annotations, we are providing annotator with browsing and searching capabilities of the Onki ontology library system.

Since the schemas used in Saha may contain any kinds of resources (classes, properties, and instances) and relations between them, the following aspects must be considered, among others, when a schema is used in annotation:

- Which of the schema's classes and properties an annotator can use in annotations and how are these resources shown to her?
- To what class(es) a property is attached, if the relations are not explicitly stated using restrictions, such as `rdfs:domain`?
- How are external ontologies and information extraction components attached to different resources of the schema?

To facilitate the use of arbitrary annotation schemas and address the questions stated above, we need a way to configure an annotation tool for various annotation schemas. The idea similar to this is discussed in [Handschuh and Staab 02], where it is proposed that the design of ontologies should be separated from the way they are used in annotation. In Saha, the rules describing how an annotation schema is to be used are defined by the administrator of an annotation project, when the project is being created. For this task, Saha offers a simple administrative interface.

2.3 Architecture and User Interface

The main difference between Saha and ontology editors such as Protégé [Noy et al. 01] is that Saha offers the end-user a highly simplified view of the underlying ontologies (annotation schemas). It does not provide tools to modify the structure (classes and properties) of ontologies (creating new sub classes for existing classes is possible), but rather focuses on using them as a basis for the annotations.

Saha is a web application implemented using the Apache Cocoon[8] and Jena frameworks. It uses extensively techniques such as JavaScript and Ajax[9]. The basic architecture of Saha is depicted in figure 2. It consists of the following functional parts: 1) annotators using web browsers to interact with the system, 2) Saha application running on a web server, 3) applications using the annotations created with Saha, 4) the Onki ontology service, 5) PostgreSQL database used store the annotations, and 6) the Poka information extraction tool.

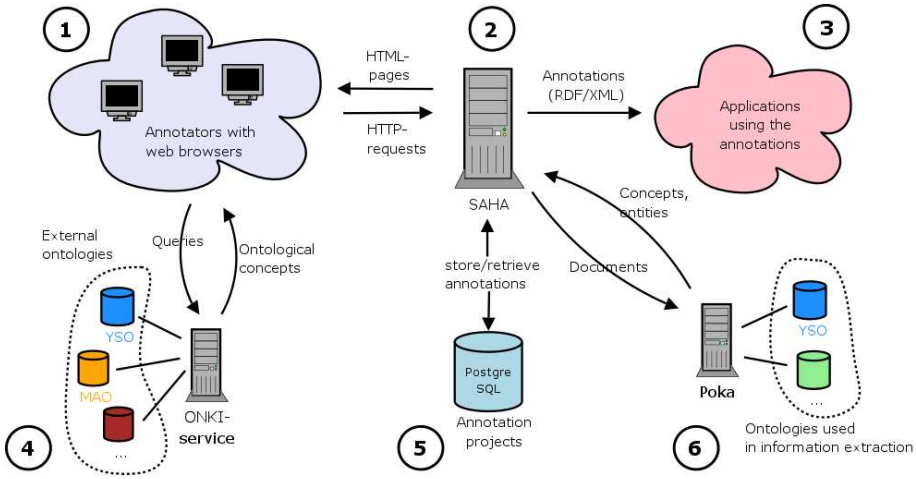

Figure 2: Architecture of Saha

[8] http://cocoon.apache.org/
[9] http://en.wikipedia.org/wiki/Ajax_%28programming%29

The user interface of Saha, depicted in figure 3, provides an annotator with a view of the classes and properties of an annotation schema. The annotator can choose a class from the class hierarchy (left side of the screen), view the annotations/KB-instances and create new ones. The lower part of the screen views the resource being annotated. In figure 3, an annotation belonging to class "Web page" is being edited. The properties of the annotation, such as "Title", as well as fields to supply values for them are shown on the right side of the class hierarchy.

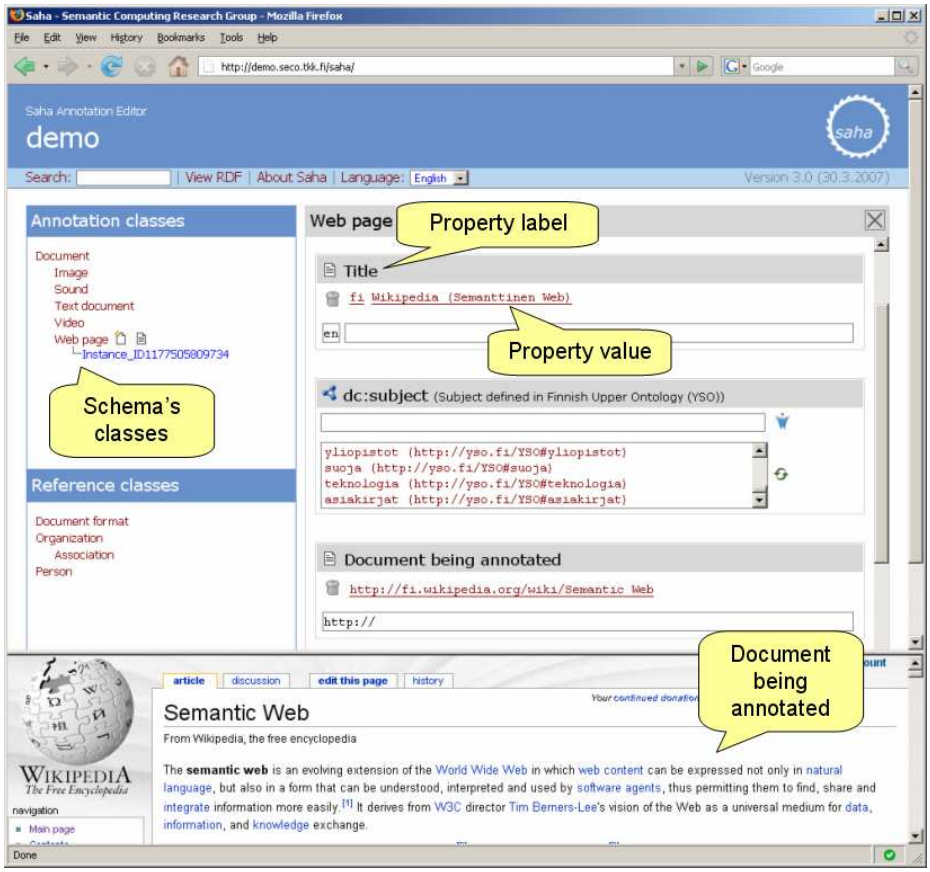

Figure 3: The User Interface of Saha

Properties of an annotation schema accept either literal or object values. In the latter case, values are KB-instances or concepts of some external domain ontology. KB-instances can be chosen using semantic autocompletion [Hyvönen and Mäkelä 06]. Here, the user types in a search word and selects a proper instance from the list populated dynamically by the system after each input character. If the proper KB-instance does not exist, user may also create a new one. `rdfs:range` or `owl:Restriction` is used to define the types of things that are allowed as values.

2.4 Setting Up an Annotation Project

Saha's annotation cycle starts by loading an annotation schema to Saha server and defining settings for the schema. The whole process of creating a project is done using an administrative interface of Saha with a common web-browser. The settings for an annotation project will define 1) the way how the schema is visualized for the annotator, 2) how human readable labels (`rdfs:label`) are automatically created for new annotations and KB-instances, and 3) how different property fields are filled in the annotations. By visualization, we refer to e.g. defining a subset of schema's classes that are shown in the editor's class-hierarchy, or defining the order of the properties of a class in which they are shown to the annotator. The idea of setting the layout for the schema's classes is similar to the *forms* used in Protégé [Noy et al. 01]. Human readable labels, in turn, are needed when annotations or instances are represented in the user-interface of an annotation editor or some application using the annotations. These labels can be, in many cases, formed automatically using property-values supplied by the annotator for the annotation/KB-instance. For example, we can state in the settings of an annotation project, that the value of a property `foaf:name` should be used as a *rdfs:label* of the instance of the class `foaf:Person`, to which the property `foaf:name` belongs to.

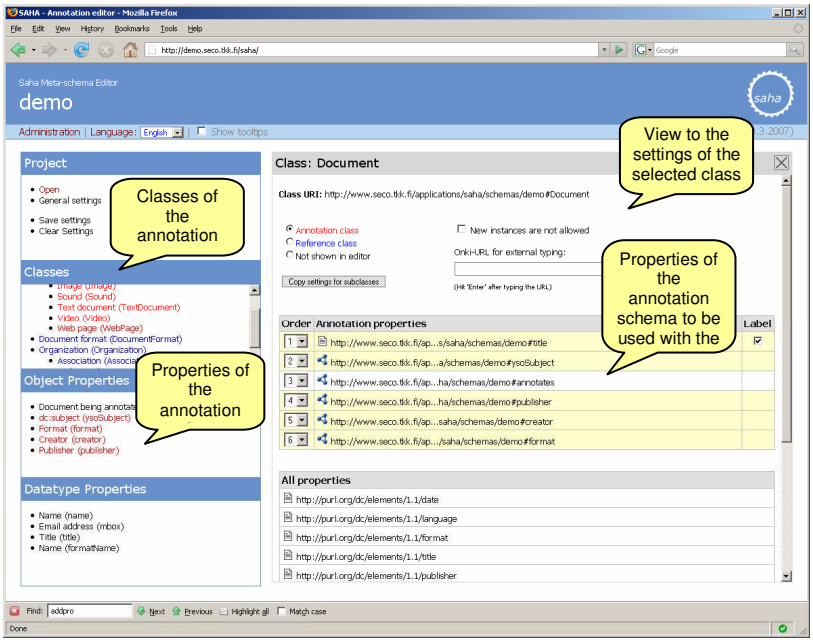

Figure 4: Defining the settings for an annotation project

In Saha, property values can be filled in manually or by using integrated ontology services, which include the ontology server system Onki and the automatic information extraction tool Poka. When using these services, we map a property of an annotation schema to the desired service. In the case of Onki, the values of the

property will be concepts defined in some external domain ontologies, selected by an annotator using a dedicated Onki-browser. When Poka is used, values are ontological concepts or literals provided by the extraction tool. For example, an extraction component recognizing person names could be coupled with the property `dc:creator`.

Settings for an annotation project are done using a dedicated browser-based user interface, which provides an administrator of an annotation project a view to the classes and properties of an annotation schema. Using the interface (depicted in figure 4), an administrator can define the basic settings for the project, such as choosing the classes of the schema to be used in annotations.

3 Onki Ontology Services

One of the key features of Saha is its ability to connect to centralized ontology server system Onki [Komulainen et al. 05, Viljanen et al. 07]. The idea of Onki is to provide applications with ready-to-use ontology service functionalities such as ontology browsing and searching annotation concepts using semantic autocompletion. In addition to offering browse and search capabilities, static ontology files are made available for downloading in RDF/XML. Here, recipes 5 and 6 of the best practices for publishing RDF vocabularies recommended by W3C [W3C, 06], are followed.

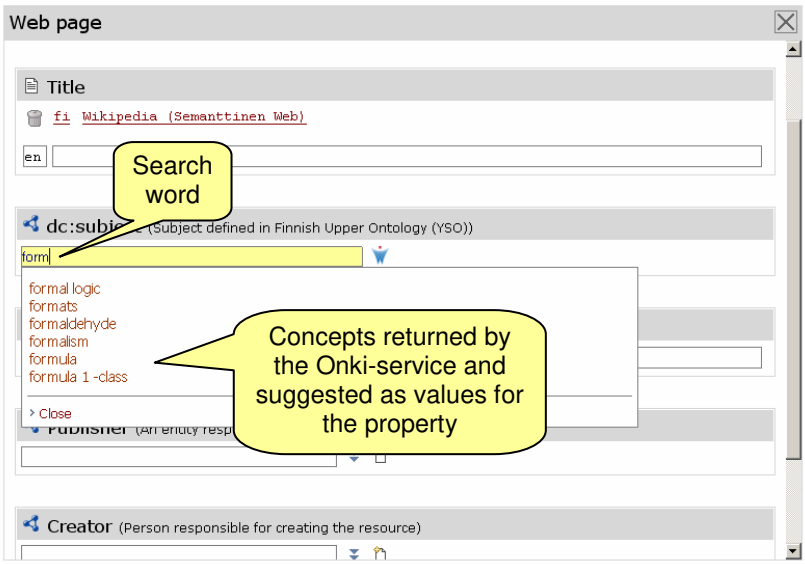

Figure 5: Fetching concepts from Onki ontology service using semantic autocompletion

In Saha, concepts of external domain ontologies can be used as values of an annotation schema's properties using services provided by Onki. Concepts are made available for the annotators through Onki's two interfaces to ontological information:

searching (see figure 5) and browsing (see figure 6). The first one is similar to the instance KB search for choosing values to object properties. When using it, the annotator types a search word which is sent to the Onki ontology server character by character and matched with the concepts in the underlying ontology. Concepts matching to the query will be sent back to Saha and shown below the search field from which they can be selected by the user. The other option, depicted in figure 6, is to use a browser view of the Onki system. It is practical when the annotator does not get agreeable results using semantic autocompletion, or wants to see the resources within the context of the class hierarchy. The Onki ontology browser can be opened in a new window by clicking a property field in Saha. After that, the annotator is able to browse the class hierarchy, and when a suitable concept is found, fetch it to the input form of Saha by clicking on the button "Fetch concept" on the Onki browser page. Both modes of using ontology services provided by Onki can be conveniently integrated with different web applications on the client side using Ajax.

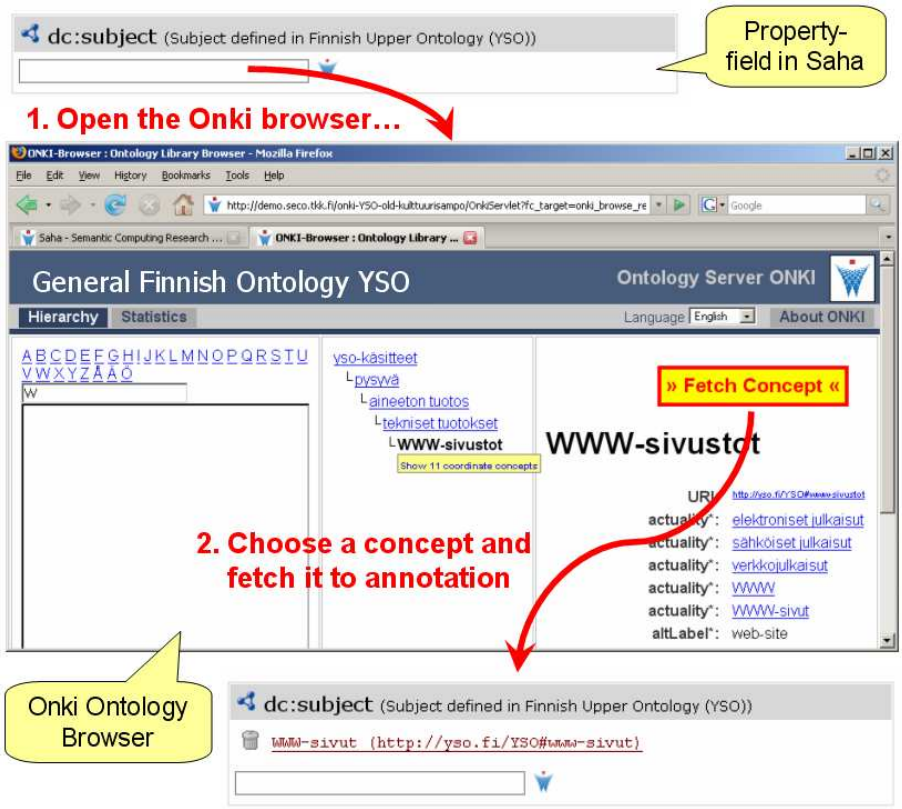

Figure 6: Fetching concepts from Onki ontology service using browser-interface

4 Poka Information Extraction Tool

Saha uses the ontology-based information extraction tool Poka for suggesting concepts from the documents. Poka recognizes 1) concepts of reference ontologies and 2) non-ontological[10] named entities.

In schema-based annotation, things to be extracted are defined by the properties of the annotation schema's classes. Accordingly, the function of an extraction component is to provide suitable concepts or entities as property values. To support arbitrary annotation schemas, extraction tools must be adaptable to different extraction tasks. In Poka environment, we have solved the problem of adaptivity in two ways. First, we have implemented generic non-ontological extraction components such as person name identifier and regular expression extractor. Second, user-defined external ontologies can be integrated with the system and used in concept recognition.

4.1 Document Pre-processing

For the extraction task, the document is pre-processed to Poka's internal document format. For the language-dependent content, a language dependent syntactic parser can be coupled in the system. Currently, Poka uses the Finnish morphosyntactic analyzer and parser FDG [Tapanainen and Järvinen 97] mainly to lemmatize words. The lemmatization of text is especially useful because the syntactical forms of words may vary greatly in languages with heavy morphological affixation (e.g. Finnish) [Löfberg et al. 03]. For other languages, the morphosyntactic analyser can be replaced with another language-specific stemmer or lemmatizer by building a parser that implements Poka's parser interface. If the lexical forms of words do not vary much, it is also possible to use language-independent "tokenisation–only" approach.

For the Finnish content, part-of-speech tagging of FDG is also utilized. Based on the part-of-speech information, Poka tags the tokens to substantives, adjectives, numerals, verbs and uppercase type. With the typing of tokens, Poka's extraction process can be focused on a certain type of tokens to speed up the extraction phase. In some cases, the focusing also helps to discard false hits in concept matching. For example, if we know that the named entities are written in uppercase format in the document collection, the search of places from the place ontology can be started from the uppercase words. Respectively, search of the verb-ontology's resources can be focused to verb-tokens.

Poka can extract concepts from various document types. In Saha integration, it currently supports concept extraction from HTML, PDF, and text documents. The support for other document formats (e.g. MS Word and PostScript) can be achieved with the integration of text decoder tools.

4.2 Extraction of Non-ontological Entities

Poka utilizes two extraction components for non-ontological entity extraction: person name extractor for Finnish language and regular expression extractor. The main idea in the rule-based name recognition tool is to first search for full names within the text at hand. After that, occurrences of the first and last names are mapped to full names.

[10] This term will be used to denote entities not explicitly present on the ontology at hand.

Simple coreference resolution within a document is implemented by mapping the individual name occurrences to the corresponding unambiguous full name if one exists. Individual first names and surnames without corresponding full names are discarded. Search for potential names is started from the uppercase words of the document utilizing a predefined list of first names. With morphosyntactic clues some hits can be discarded. For example, first names in Finnish rarely have certain morphological affixations such as "-ssa" (similar to the English preposition "in") or "-lla" (preposition "on") when they occur before the surname in the sentence. The FDG-parser's surface-syntactic analysis is also used for revealing proper names.

The names that are automatically recognized are suggested as potential new instances in Saha. The type of a new instance is a reference class of the annotation schema used in Saha, e.g. `foaf:person`. If there exists an instance with the same name, the annotator can tell whether the newfound name refers to an existing instance or to a new one. If a new instance is created, the user fills additional person information according to the schema definition.

The regular expression extractor is utilized to suggest values for literal properties. The extraction pattern is defined in the annotation project's settings. A pattern is defined in Java pattern notation[11]. When the document URL is set, the pattern is matched against words of the document. For example, a date pattern of the form DD-MM-YYYY, retrieves all the occurrences of the pattern to the literal field. The suitable values are then selected by the user. Current implementation does not support multi-word patterns.

4.3 Extraction of Ontological Concepts

By ontological extraction we mean 1) deduction of relevant string representations of concepts from the ontology and 2) finding the occurrences of the representations. In Poka, the extraction of ontological concepts starts by defining a set of concepts in an ontology that are to be extracted from the documents. The ontology can be used in its entirety, or it can be only partly used by selecting e.g. instances or some sub-part of the ontology's hierarchy tree. After this, the human readable property values representing concept names, e.g. the values of the literal property `rdfs:label`, are chosen as the target for the recognition.

To ease the integration of user-defined ontologies as vocabularies, we have created a browser-based user interface, DynaPoka, for this task (see figure 7). DynaPoka offers a way to examine and view the literal resources of an ontology. First, an ontology is uploaded to the server and the ontology's subsumption structure (`rdfs:sublassOf`), literal properties and language tags are shown. The end-user can view literal values by choosing properties and languages. Selection of a class in the tree-view shows the labels of the resources that are subclasses of the selected class or instances of a subclass. DynaPoka offers a user interface to test how the selected string values suit the extraction task. The user interface has an inline frame for visualising the extraction of web pages. After the URL input, each occurrence of the resource's string is bolded in the HTML document.

DynaPoka acts as a tool for integrating a user-defined ontology for Saha. The selected resources can be serialized as Poka's internal term file format. The term files

[11] http://java.sun.com/javase/6/docs/api/java/util/regex/Pattern.html

define the resource labels to be extracted using Saha. DynaPoka eases the adaptation and reuse of ontological resources as extraction vocabularies. It offers a solution to the portability problem which can be seen as a major drawback in the reuse of information extraction systems [Grishman 97].

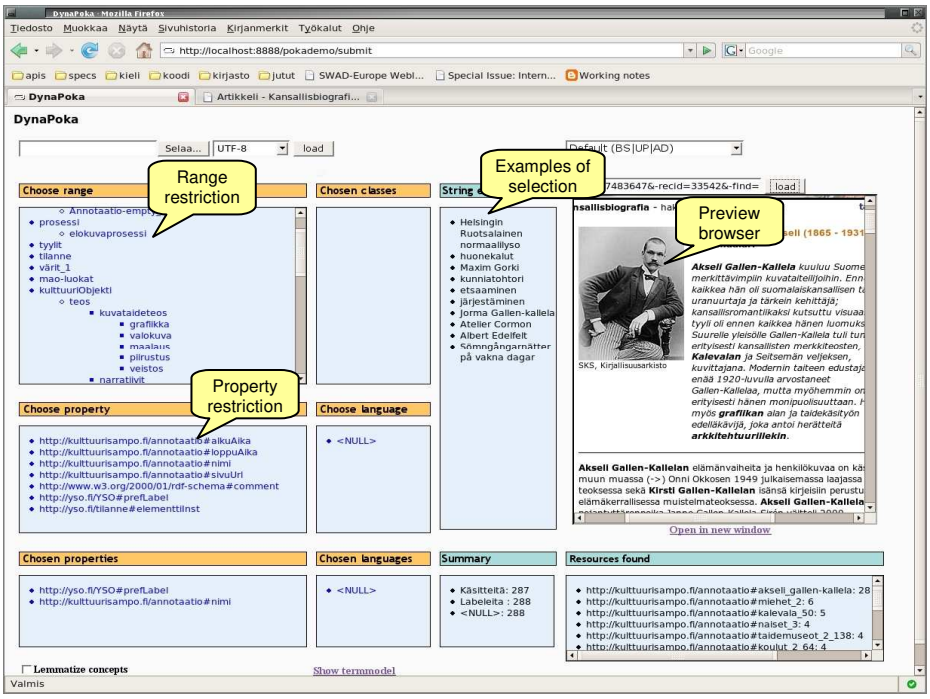

Figure 7: DynaPoka user interface

For efficient string matching in the extraction process, the string representations of ontological resources are indexed in a prefix tree. Since two or more concepts may share the same label, a word in the trie is allowed to refer to multiple URIs. For language-dependent concept search, the labels of ontological concepts are also lemmatized to achieve better recall.

Currently the adaptation of new extraction ontologies is done by system experts. Our future work involves developing a user interface for integrating ontological resources for extraction.

In our project, we have mainly harnessed two ontologies for extraction. For place recognition, the place ontology of the MuseumFinland portal [Hyvönen et al. 05b] extended in the CultureSampo portal [Hyvönen et al. 07a] is exploited. For topic indexing, the General Finnish Upper Ontology (YSO) [Hyvönen et al. 05a, 07b] is in use. It contains over 20,000 Finnish indexing concepts organized into ten major

subsumption hierarchies. The concepts of the ontology are based on the indexing terms of the General Finnish Thesaurus YSA[12].

To improve the recognition of important indexing terms, it is possible to weight the concepts of a document in different ways. For example, in topic indexing, concepts that form semantic *cliques*, i.e. semantically related terms, gain more weight as suggested in [Vehviläinen et al. 06]. This means that suggested YSO-concepts for a topic property are ordered by not only the term frequency, but by taking into account the semantic connectedness or neighbourhood context of the concepts as well.

4.4 Resource Identification

A major problem in automatic concept extraction is resource identification. The problem of identification can be divided into two disambiguation problems: identification between different resources sharing a same name and identification between a resource defined in a model (ontology) and a resource that is not in the model. A system supporting the first case has to be able to deal with resources that share a similar name. For the second problem, the system has to be able to instantiate a new resource even if the model contains already one or more resources with a similar name. A system without the support for identification usually operates on a lexical level discarding the uniform identifiers.

Poka supports both identification cases. In Saha, the resource identification is solved by human intervention. For example, if Poka has found a string token from the document matching to the labels of two different concepts in the place ontology, the user has to choose one of them, or create a new place concept.

One of the main difficulties in the Saha user interface is how to represent disambiguation for the annotator. For example, we may have an object property with a class *person* as its range. Existing persons for the property may simultaneously be derived from 1) an actor ontology connected to Poka's concept extraction tool, 2) Saha's local instance KB or 3) the actor ontology in the Onki server. Moreover, an annotator may want to instantiate a new person. All in all, the identification task seems to induce complexity in the ontology-based modeling and annotation.

Systems utilizing automatic annotation like Magpie [Dzbor et al. 03] and Melita [Ciravegna et al. 02] cannot identify resources sharing the same label. In fully automatic annotation systems KIM [Popov et al. 06] and Semtag [Dill et al. 03], the architectures support instance identification in a restricted pre-defined ontology model.

In Saha annotation framework, the annotation process is semi-automatic and based on the user's input. This approach enhances the manual annotation process, but may be insufficient for vast annotation projects. A shift towards automatic annotation could be provided by first annotating automatically all possible resources and then using Saha for qualifying the results. The quality assurance approach could be provided with a user interface highlighting the possible conflicts in the annotations, such as disambiguated resources and new resources created with related names.

[12] http://vesa.lib.helsinki.fi/

5 Related Work

A number of semantic annotation systems and tools exist today [Reeve and Han 05, Uren et al. 06]. These systems are primarily used to create and maintain semantic metadata descriptions of web pages.

5.1 Manual and Semi-Automatic Annotation Tools

Annotea [Kahan et al. 01] supports collaborative, RDF-based markup of web pages and distribution of annotations using annotation servers. Annotations created with Annotea can be regarded as semi-formal, since the system does not support the use of ontological concepts in annotations. Instead, annotations are textual notes which are associated with certain sections of the documents they describe.

The Ont-O-Mat system [Handschuh and Staab 02], in turn, can be used to describe diverse semantic structures as well as to edit ontologies. It also has a support for automated annotation. The user interface of the Ont-O-Mat is not, however, very well-suited for annotators unfamiliar with concepts related to ontologies and semantic annotation in general.

Semantic Markup Tool (SMT) [Kettler et al. 05] is a schema-based, semi-automatic annotation system developed by the ISX Corporation. The system supports information extraction from the HTML-pages. Extraction is based on commercial, non-ontological tools [Kettler et al. 05] which retrieve person names, place names and date strings from the documents. Schemas are defined by a system expert and it is not clearly stated how easily the system can be adapted to new ones. In SMT, schemas are defined in XML and the output can be serialized into OWL. The article [Kettler et al. 05] does not explicitly state the way how extracted literal values are treated. For example, if the extracted values act as names (e.g. `rdfs:label`) of the new instances (e.g. persons), is it possible to define other names for them? If not, identification of resources sharing the same name is difficult if the only distinctive feature is the URI of each resource. Another fundamental issue concerns the possibility to re-reference populated instances. To achieve rich semantic markup, an instance populated from a document must be reached from another. For example, two documents may have the same author.

Most of the current annotation systems, like the ones mentioned here, are applications that run locally on the annotator's computer. Because of this, the systems may not necessarily be platform independent and must always be installed locally by the user before annotating. In Saha, these problems are addressed by implementing the system as a web application. By doing so, the system can be installed and maintained centrally and the requirements for the annotator's computational environment are minimal. The way Saha is designed and implemented also strongly supports the collaboration in annotation, making the sharing of annotations and new individuals (free indexing concepts) easy.

5.2 Methods for Automatic Extraction

In the field of automatic ontology-based annotation, annotation methods can be divided into two main groups. The first group consists of applications that rely on non-ontological (e.g. rule-based) extraction tools which are used to find new instances

from the text. An extraction component is usually connected to a class (or classes) of an ontology. Applications using non-ontological extraction methods are, for example, Gate's OntoGazetteer [Kenter and Maynard 05], KIM [Popov et al. 03] and SMT [Kettler et al. 05]. An adaptive extraction tool Amilcare [Ciravegna and Wilks 03] can be also characterised as a non-ontological extraction tool: strings tagged from a document define the extraction pattern for the concept.

The second group consists of tools where the information to be extracted is primarily derived from an ontology. Applications in this category involve DynaPoka and the concept highlighter tool Magpie [Dzbor et al. 03] and automatic ontology-based indexing and retrieval tool Semtag [Dill et al. 03].

Magpie is a web browser plugin that highlights string representations of ontological resources on a web page. In hands-on testing, Magpie's extraction methods show some apparent weaknesses. If the ontology contains two or more corresponding strings, only the first one is shown. In addition to this, there seems to be some problems with overlapping strings as well. For example, if the ontology has string representations "semantic", "web" and "semantic web", the document's string "semantic web" does not match with the last one.

In KIM, extraction of ontological resources is based on a set of non-ontological extraction tools. An extraction tool is harnessed to extract certain types of concepts. For example, KIM's person name recognition tool extracts person names and compares a found string to the string representations of persons in the ontology. If a string matches with the existing one, an occurrence of an existing person is found. Otherwise a new person is instantiated. An elegant feature of the KIM system is to treat pre-populated ontological instances as trusted; conversely, new instances found from the documents are uncertain, untrusted entities [Popov et al. 03].

Semtag extracts, disambiguates and indexes automatically named entity resources of the ontology. Albeit resembling the KIM platform [Popov et al. 03], the extraction method is completely different. Semtag extracts strings based on the string representations of the ontology.

From our point of view, the field of automatic annotation lacks systems that can easily adapt existing ontologies to be used as vocabularies in extraction. In Magpie, word lists are derived from the ontology by the system experts. Semtag and Kim utilize state-of-art extraction methods, but the systems are difficult to adapt for different tasks.

6 Evaluation

Saha is a working prototype. It is in trial use for the distributed content creation of semantic web portals being developed in the FinnONTO project. These include, among others, the semantic health promotion portal TerveSuomi.fi [Holi et al. 06, Suominen et al. 07, Hyvönen et al. 07c] and CultureSampo, a semantic portal for Finnish culture [Hyvönen et al. 07a].

Full usability testing of Saha has not yet been conducted. Initial feedback from end-users indicates that some intricate ontological structures, such as deep relation paths between resources, may be difficult to comprehend. These difficulties, however, can be facilitated by proper design of annotation schemas. Following cases explain how Saha is being applied in metadata creation for CultureSampo.

6.1 Annotating Historical Buildings in Espoo

A dataset of Espoo City Museum containing information on the historical real estates in the city of Espoo is currently being converted for the use in CultureSampo. The dataset contains descriptions of buildings, such as the year of construction, architect, coordinates, keywords, etc. The initial data is in plain XML-format and the conversion started by transforming it to RDF/OWL using an automatic tool implemented for the task. This conversion involved, among others, mapping literal keywords to corresponding ontological concepts of the Finnish Upper Ontology YSO using the information extraction tool Poka. After the conversion, the data was loaded to Saha for checking the validity of the annotations and to make necessary corrections and additions to them. This task was handed over to domain experts working in the museum. For each historical building, free textual descriptions which were not included in the original XML-dataset were added. In addition to this, some ontological indexing concepts were added using the ontology browser Onki. The total number of real estates annotated was 80.

Saha offered a convenient way for the museum's staff to take part in the content creation process where the data is ontologically described. Despite not being experienced in data processing in general and in semantic web technologies in particular, they were successful in using Saha and achieved goals set for enrichment of the data. Here, the crucial point was to provide them with tools that did not require high degree of technical skill and were both easy to learn and to use. The case also serves as an example of how Saha can contribute to the content creation process that cannot be fully automated.

After the annotation stage at the museum, the data will be loaded to CultureSampo portal, where historical buildings are presented in context with other cultural artefacts featured in the portal.

6.2 Annotating Poems of Kalevala

A set of runes of Kalevala—the national epic of Finland—was annotated [Hyvönen et al. 07d] for the use in the CultureSampo portal. From 50 runes in Kalevala, four were selected for the case study, each about 1500 words long. The runes were divided into scenes, and each scene was annotated by a set of events taking place in it. Resources used in annotations are references to the General Finnish Upper Ontology YSO and to two Kalevala-specific ontologies: places and actors. The four runes were annotated with a total of 132 scenes and 383 events. 58 different Kalevala actors and 23 Kalevala places were used in addition to 189 annotation concepts taken from YSO. The annotations tell what kind of events and scenes take place at different lines of the runes. Based on such descriptions, semantic recommendation links with explanations to other parts of the text and other kinds of cultural artefacts can be created.

The annotations were initially created by hand by a folklorist using MS Excel. Later, the same data was manually converted to RDF/OWL using Saha. This work was done by a person with some knowledge on the Semantic Web technologies, but not involved in developing Saha. The results of the case study show that Saha can be used to create annotations with rather complex semantic structures, which may contain long relation paths between different resources.

7 Discussion and Future Work

Ontology-based semantic annotations are needed when building the Semantic Web. Although various annotation systems and methods have been developed, the question of how to easily and cost-effectively produce quality metadata still remains largely unanswered. We tackled the problem by first identifying the major requirements for an annotation system. As a practical solution, an annotation system was designed and implemented which supports distributed creation of metadata and which can utilize ontology services as well as automatic information extraction. It is designed to be easily used by non-experts in the field of the Semantic Web.

Our future plans include using Saha to provide metadata for additional semantic portals as well as further develop the automation of the annotation. Currently, the coupling of the annotation schema's properties and information extraction components provided by Poka are not fully utilizing the ontological characteristics. In other words, instead of using restrictions and constraints such as `rdfs:range` to define which of the schema's properties an automatically recognized resource matches to, we are currently using a meta-schema to do the mapping. However, our plans include using the property restrictions to do the matching in the future. We are also aiming to map the automatically extracted entities to ontologies in order to support property restriction with them as well. For example, date regular expressions would be mapped to a corresponding class of the reference ontology, say `myOnto:Date`. This way, the proper values for an object property are defined by the range (ontological restriction), not by the component itself.

Acknowledgements

This research is a part of the National Ontology Project in Finland (FinnONTO) 2003-2007, funded mainly by the Finnish Funding Agency for Technology and Innovation (Tekes) and a consortium of 36 companies and public organizations.

References

[Ciravegna et al. 02] Ciravegna, F., Dingli, A., Petrelli, D., Wilks, Y.: User-system cooperation in document annotation based on information extraction. Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW02. (2002)

[Ciravegna and Wilks 03] Ciravegna, F., Wilks, Y.: Designing adaptive information extraction for the Semantic Web in Amilcare. Annotation for the Semantic Web. IOS Press, Amsterdam. (2003)

[Dill et al. 03] Dill, S., Tomlin, J., Zien, J., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation. Proceedings of the 12th International World Wide Web Conference, WWW2003. (2003)

[Dzbor et al. 03] Dzbor, M., Domingue, J., ja Motta, E.: Magpie: towards a Semantic Web browser. Proceedings of the 2nd International Semantic Web Conference. (2003)

[Grishman 97] Grishman, R.: Information extraction: techniques and challenges. Information Extraction International Summer School SCIE-97. (1997)

[Handschuh and Staab 02] Handschuh, S., Staab, S.: Authoring and Annotation of Web Pages in CREAM. Proceedings of the 11th International Conference on World Wide Web, WWW2002. (2002)

[Holi et al. 06] Holi, M., Lindgren, P., Suominen, O., Viljanen, K., Hyvönen, E.: TerveSuomi.fi – A Semantic Health Portal for Citizens. Proceedings of the 1st Asian Semantic Web Conference, ASWC2006, poster papers. (2006)

[Hyvönen et al. 05a] Hyvönen, E., Valo, A., Komulainen, V., Seppälä, K., Kauppinen, Ruotsalo, T., Salminen, M., Ylisalmi, A.: Finnish national ontologies for the semantic web—towards a content and service infrastructure. In Proceedings of International Conference on Dublin Core and Metadata Applications, DC 2005. (2005)

[Hyvönen et al. 05b] Hyvönen, E., Mäkelä, E., Salminen, M., Valo, A., Viljanen, K., Saarela, S., Junnila, M., Kettula S.: MuseumFinland—Finnish Museums on the Semantic Web. Journal of Web Semantics, 3(2). (2005)

[Hyvönen and Mäkelä 06] Hyvönen, E., Makelä, E.: Semantic Autocompletion. Proceedings of the 1st Asian Semantic Web Conference (ASWC2006), Springer-Verlag. (2006)

[Hyvönen et al. 07a] Hyvönen, E., Ruotsalo, T., Häggström, T., Salminen, M., Junnila, M., Virkkilä, M., Haaramo, M., Mäkelä, E., Kauppinen, T., Viljanen, K.: CultureSampo—Finnish Culture on the Semantic Web: The Vision and First Results.. In: K. Robering (Ed.), Information Technology for the Virtual Museum, LIT Verlag, Berlin/London. (2007)

[Hyvönen et al. 07b] Hyvönen, E., Viljanen, K., Mäkelä, E., Kauppinen, T., Ruotsalo, T., Valkeapää, O., Seppälä, k., Suominen, O., Alm, O., Lindroos, R., Känsälä, T., Henriksson, R., Frosterus, M., Tuominen, J., Sinkkilä, R., Kurki, J.: Elements of a National Semantic Web Infrastructure—Case Study Finland on the Semantic Web (Invited paper). Proceedings of the First International Semantic Computing Conference (IEEE ICSC 2007), Irvine, California. IEEE Press. (2007)

[Hyvönen et al. 07c] Hyvönen, E., Viljanen, K., Suominen, O.: HealthFinland—Finnish Health Information on the Semantic Web. Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC/ASWC 2007), Springer-Verlag. (2007)

[Hyvönen et al. 07d] Hyvönen, E., Takala, J., Alm, O., Ruotsalo, T., Mäkelä, E.: Semantic Kalevala—Accessing Cultural Content Through Semantically Annotated Stories. Proceedings of the Workshop Cultural Heritage on the Semantic Web, the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC/ASWC 2007). (2007)

[Kahan et al. 01] Kahan, J., Koivunen, M.-R., Prud'Hommeaux, R., Swick, R.: Annotea: An Open RDF Infrastructure for Shared Web Annotations, Proceedings of the 10th International World Wide Web Conference, WWW2001. (2001)

[Kenter and Maynard 05] Kenter, T., Maynard, D.: Using GATE as an annotation tool, University of Sheffield, Natural language processing group, Available at: http://gate.ac.uk/sale/am/annotationmanual.pdf. (2005)

[Kettler et al. 05] Kettler, B., Starz, J., Miller, W., Haglich, P.: A Template–based Markup Tool for Semantic Web Content. Proceedings of the 4th International Semantic Web Conference, ISWC2005. (2005)

[Komulainen et al. 05] Komulainen, V., Valo, A., Hyvönen, E.: A Tool for Collaborative Ontology Development for the Semantic Web. Proceedings of the International Conference on Dublin Core and Metadata Applications, DC 2005. (2005)

[Känsälä and Hyvönen 06] Känsälä, T., Hyvönen, E.: A Semantic View–Based Portal Utilizing Learning Object Metadata. Proceedings of the Workshop on Semantic Web Applications and Tools, the 1st Asian Semantic Web Conference, ASWC2006. (2006)

[Löfberg et al. 03] Löfberg, L., Archer, D., Piao, S., Rayson, P., McEnery, T., Varantola, K., Juntunen, J.-P.: Porting an English Semantic Tagger to the Finnish Language. In Proceedings of the Corpus Linguistics 2003 conference, pp. 457–464. UCREL, Lancaster University. (2003)

[Noy et al. 01] Noy, N., Sintek, M., Decker, M.S., Crubézy, M., Fergerson, R.: Creating Semantic Web Contents with Protégé–2000. IEEE Intelligent Systems 2(16):60–71. (2001)

[Popov et al. 03] Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A., Goranov, M.: Towards Semantic Web Information Extraction. Proceedings of ISWC, Sundial Resort, Florida, USA. (2003)

[Popov et al. 06] Popov, B., Kitchukov, I., Angelov, K., Kiryakov, A.: Co-occurrence and ranking of entities. Available at: http://www.ontotext.com/publications/CORE_otwp.pdf. (2006)

[Reeve and Han 05] Reeve, L., Han, H.: Survey of Semantic Annotation Platforms. Proceedings of the 2005 ACM Symposium on Applied Computing. (2005)

[Schreiber et al. 01] Schreiber, G., Dubbeldam, B., Wielemaker J., Wielinga, B.: Ontology–Based Photo Annotation. IEEE Intelligent Systems, 16(3):66–74. (2001)

[Suominen et al. 07] Suominen, O., Viljanen, K., Hyvönen, E.: User-centric Faceted Search for Semantic Portals. Proceedings of the 4th European Semantic Web Conference ESWC2007, Springer-Verlag. (2007)

[Tapanainen and Järvinen 97] Tapanainen, P., Järvinen, T.: A Non–projective Dependency Parser. Proceedings of the 5th Conference on Applied Natural Language Processing, pp. 64–71. (1997)

[Uren et al. 06] Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art. Journal of Web Semantics, 4(1):14–28. (2006)

[Valkeapää and Hyvönen 06] Valkeapää, O., Hyvönen, E.: A Browser-based Tool for Collaborative Distributed Annotation for the Semantic Web. Proceedings of the Workshop on Semantic Authoring and Annotation, the 5th International Semantic Web Conference, ISWC2006. (2006)

[Valkeapää et al. 07], Valkeapää, O., Alm, O., Hyvönen, E.: Efficient Content Creation on the Semantic Web Using Metadata Schemas with Domain Ontology Services (System Description). Proceedings of the 4th European Semantic Web Conference ESWC2007, Springer-Verlag. (2007)

[Vehviläinen et al. 06] Vehviläinen, A., Hyvönen, E., Alm, O.: A Semi-automatic Semantic Annotation and Authoring Tool for a Library Help Desk Service. Proceedings of the Workshop on Semantic Authoring and Annotation, the 5th International Semantic Web Conference, ISWC2006. (2006)

[Viljanen et al. 07] Viljanen, K., Hyvönen, E., Mäkelä, E., Suominen, O., Tuominen, J.: Mash-up Ontology Services for the Semantic Web. Demo track at the European Semantic Web Conference ESWC2007. (2007)

[W3C, 06] World Wide Web Consortium, Best Practice Recipes for Publishing RDF Vocabularies; Link: http://www.w3.org/TR/swbp-vocab-pub/