Efficient Content Creation on the Semantic Web Using Metadata Schemas with Domain Ontology Services
(System Description)

Onni Valkeapää, Olli Alm, and Eero Hyvönen

Helsinki University of Technology (TKK), Laboratory of Media Technology
University of Helsinki, Department of Computer Science
Semantic Computing Research Group (SeCo)
P.O. Box 5500, FI–02015 TKK, Finland
{onni.valkeapaa,olli.alm,eero.hyvonen}@tkk.fi
http://www.seco.tkk.fi/

Metadata creation is one of the major challenges in developing the Semantic Web. This paper discusses how to make provision of metadata easier and cost-effective by an annotation editor combined with shared ontology services. We have developed an annotation system supporting distributed collaboration in creating annotations, and hiding the complexity of the annotation schema and the domain ontologies from the annotators. Our system 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.

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 [1]. 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 [14,18]. 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 [19].

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 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

E. Franconi, M. Kifer, and W. May (Eds.): ESWC 2007, LNCS 4519, pp. 819–828, 2007.

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[1] [19], an ontology service framework Onki[2] [9] and an information extraction tool Poka[3] 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 automatically 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 (FinnONTO)[4] [4].

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.
- **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 Saha Annotation System

2.1 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

[1] http://www.seco.tkk.fi/applications/saha/
[2] http://www.seco.tkk.fi/applications/onki/
[3] http://www.seco.tkk.fi/applications/poka/
[4] http://www.seco.tkk.fi/projects/finnonto/

(reference/domain ontologies) [15]. An annotation schema has an important role in expressing *how* the ontological concepts used in annotations are related to the web 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 [8] to form its user interface according to an annotation schema loaded on 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 same cases, URI). We make the distinction between the annotation of a document (e.g. a web page) and the description of some other resource (e.g. a person) that is somehow related to the document being annotated. In addition to containing classes used to annotate documents (*annotation classes*), an annotation schema used with Saha can also contain *reference classes* for describing resources other than documents. In other words, an annotation schema forms a basis for the local knowledge base (KB) that contains descriptions of different kinds of resources that may or may not exist on the web. Instances of the reference classes are used as values of properties in annotations.

Each annotation schema loaded to Saha forms an *annotation project*, which can have multiple users as annotators. In practice, an annotation project is Jena's[5] ontology model stored in a database. A model is comprised of the annotation schema and the instances of the schema's classes. It can be serialized to RDF/XML in order to use the annotations in external applications.

2.2 Architecture and User Interface

The main difference between Saha and ontology editors such as Protégé [12] 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, but rather focuses on using them as a basis for the annotations.

Saha is a web application implemented using the Apache Cocoon[6] and Jena frameworks. It uses extensively techniques such as JavaScript and Ajax[7]. The basic architecture of Saha is depicted in figure 1. 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.

The user interface of Saha, depicted in figure 2, provides an annotator with a view of the classes and properties of an annotation schema. The annotator can choose a

[5] http://www.hpl.hp.com/semweb/tools.htm
[6] http://cocoon.apache.org/
[7] http://en.wikipedia.org/wiki/Ajax_%28programming%29

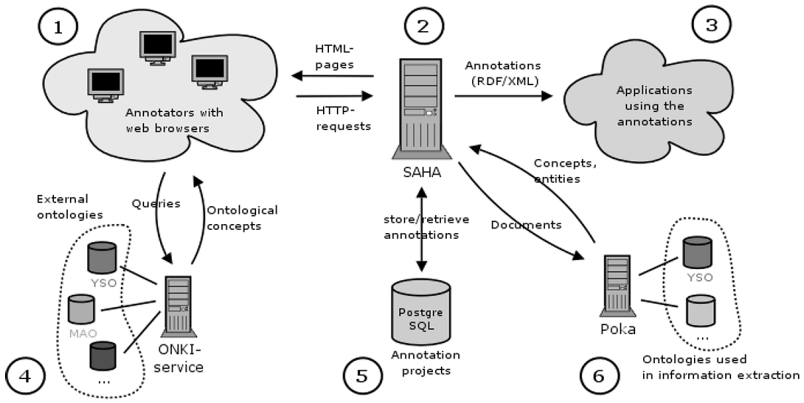

Fig. 1. Architecture of Saha

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 2, an annotation belonging to class "Document" 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.

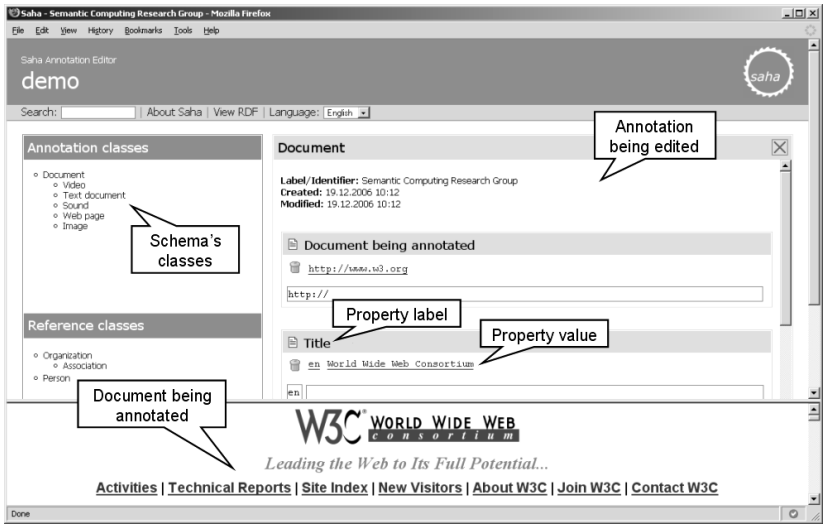

Fig. 2. The user interface of Saha

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 [5]. Here, the user types

in a search word and selects a proper instance from the list populated by the system. 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.

2.3 Setting Up an Annotation Project

Saha's annotation cycle starts by defining settings for an annotation schema. These settings 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 an order of the properties of a class in which they are shown to the annotator. Human readable labels, by turn, are needed when annotations or instances are represented in the user-interface. These labels can be, in many cases, formed automatically using property-values supplied by the annotator for the annotation/KB-instance. In Saha, properties can be filled manually or using integrated ontology services, which include the ontology server system Onki and the information extraction tool Poka to be presented in section 3. 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 people's names could be coupled with the property *dc:creator*.

Settings for an annotation project are defined in a schema-specific RDF-file, which we call *meta-schema*. Although the use of a meta-schema is not compulsory, it is highly practical in most cases. At the moment, meta-schemas are done by hand, but we are developing an easy-to-use editor for the task.

3 Utilizing Ontology Services

3.1 Onki Ontology Services

One of the key features of Saha is its ability to connect to the Onki ontology service [9]. The Onki system has an important role in sharing ontological resources between different organizations and actors. In annotation, Onki enables the use of concepts of external domain ontologies as values of an annotation schema's properties. These ontologies are made available to the annotators through the Onki ontology server, which offers two interfaces to ontological information: searching and browsing. The first one is similar to the instance KB search described above. 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 is to use a browser view of the Onki system. It is practical when the annotator does not get agreeable

results using the 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 (see figure 3). 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 to different web applications on the client side using Ajax.

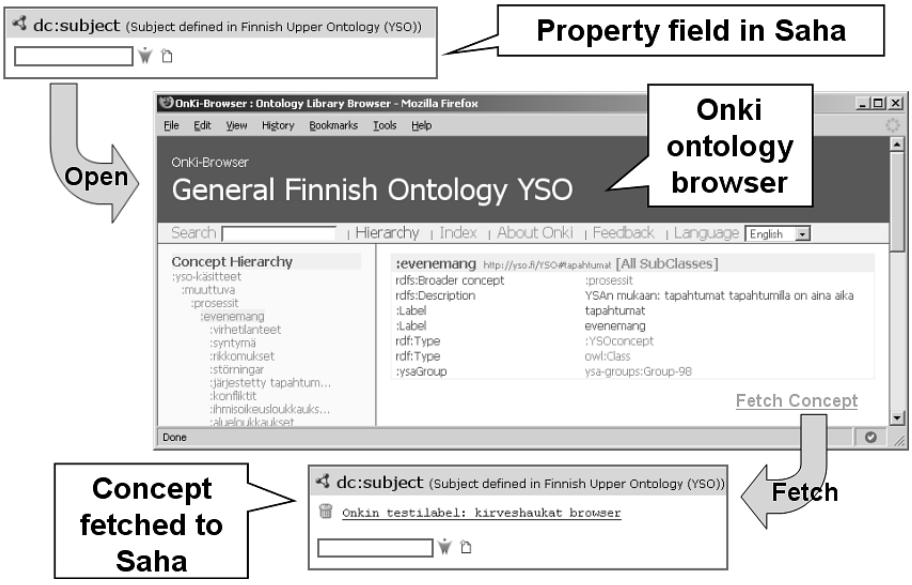

Fig. 3. Using the Onki ontology browser

3.2 Automatic Recognition of Concepts and Entities

Saha uses the ontology-based information extraction tool Poka to suggest concepts based on the documents being annotated. Poka can process a document to 1) recognize concepts of external ontologies and to 2) extract named entities using non-ontological tools.

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 to be used as values of those properties. Because Saha supports arbitrary annotation schemas, extraction tools must be adaptable in order to support different extraction tasks. In the 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.

Extraction of Non-ontological Entities. In Poka, two extraction components for non-ontological entities have been implemented: 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. Simple coreference resolution within a document is implemented by mapping the individual name occurrences to the corresponding unambiguous full name if there exist one. Single first names and surnames without corresponding full names are discarded. Search for potential names is started from the uppercase words of the document. Predefined list of first names is utilized for recognizing potential full names. With morphosyntactic clues some hits can be discarded. For example, first names in Finnish rarely have certain morphological affixation such as "-ssa" (similar to the English preposition "in") or "-lla" (preposition "on") when they occur before the surname in the sentence. Poka makes use of the morphosyntactic analyzer and parser FDG[8] [17]. 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, say *myAnnotation:Person*. If there exists an instance with the same name, the user can tell whether the newfound name refers to an existing instance or to a new one.

The regular expression extractor acts in a similar way as the name extractor. The difference is that the thing to be extracted is defined by the expression, not the component itself. Expressions can be utilized to find literal values or potential new instances from the document.

Extraction of Ontological Concepts. For ontological extraction, an ontology has to be integrated to the extraction system. By ontological extraction we mean 1) deduction of string representations of concepts *from* the ontology and 2) finding the occurences of the representations.

In Poka, the integration 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 properties representing concept names, e.g. the values of the literal properties *rdfs:label* or *skos:prefLabel*, are chosen as targets for the recognition.

For the string matching in the extraction process, the string representations of ontological resources are indexed in the prefix trie. Since two or more concepts may share the same label, a trie entity can refer to multiple URIs. In some languages (e.g. Finnish), it is useful to lemmatize the concept representations for efficient extraction. This is because syntactical forms of words may vary greatly in languages with heavy morphological affixation [11]. Lemmatization of *both* the text and the concept names helps to achieve better recall in the extraction process.

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.

[8] http://www.connexor.com, Machinese Syntax

4 Discussion

4.1 Contributions and Applications

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 the 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.

Saha is currently a working prototype. It is in trial use for the distributed content creation of the semantic health promotion portal TerveSuomi.fi [3,16]. Much of the content and metadata for the portal will be provided by health experts working at various health organizations in Finland. Saha has also been tested, among others, in metadata creation for the Opintie portal, a follow-up version of the educational semantic portal Orava [10], using Learning Object Metadata (LOM).

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, are difficult to comprehend. These difficulties, however, can be facilitated by proper design of annotation schemas.

4.2 Related Work

A number of semantic annotation systems and tools exist today [14,18]. These systems are primarily used to create and maintain semantic metadata descriptions of web pages.

Annotea [6] 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 Semantic Markup Tool [8] has a user interface that is generated according to an annotation schema in a similar way as is done in Saha. It uses Information Extraction techniques to find different kinds of entities in documents and proposes them for values of the annotation's properties. The schemas it supports are relatively simple, and it cannot be thus used to describe more complex semantic relations. Moreover, the expressivity and adaptation of templates is not explicitly stated in [8]. The Ont-O-Mat system [2], 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 the annotators unfamiliar with concepts related to ontologies and semantic annotation in general. Another example of the user interface of an annotation tool requiring understanding of the Semantic Web concepts can be found in SMORE [7].

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 on the user's system, before the annotation can begin. 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.

4.3 Future Work

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 the 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 37 companies and public organizations.

References

1. Dill, S., Tomlin, J., Zien, J., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S. and Tomkins, A. (2003) SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation. Proceedings of the 12th International World Wide Web Conference, WWW2003.
2. Handschuh, S. and Staab, S. (2002) Authoring and Annotation of Web Pages in CREAM. Proceedings of the 11th International Conference on World Wide Web, WWW2002.
3. Holi, M., Lindgren, P., Suominen, O., Viljanen, K. and Hyvönen, E. (2006) TerveSuomi.fi – A Semantic Health Portal for Citizens. Proceedings of the 1st Asian Semantic Web Conference, ASWC2006, poster papers.
4. Hyvönen E. (2006) FinnONTO—Building the Basis for a National Semantic Web Infrastructure in Finland. Proceedings of the 12th Finnish AI Conference STeP 2006.
5. Hyvönen, E. and Makelä, E. (2006) Semantic Autocompletion. Proceedings of the 1st Asian Semantic Web Conference, ASWC2006.

6. Kahan, J., Koivunen, M.R., Prud'Hommeaux, E. and Swick R.R. (2001) Annotea: An Open RDF Infrastructure for Shared Web Annotations, Proceedings of the 10th International World Wide Web Conference, WWW2001.
7. Kalyanpur, A., Hendler, J., Parsia, B. and Golbeck, J. (2005) SMORE – Semantic Markup, Ontology, and RDF Editor. Available at: http://www.mindswap.org/papers/SMORE.pdf
8. Kettler, B., Starz, J., Miller, W. and Haglich, P. (2005) A Template–based Markup Tool for Semantic Web Content. Proceedings of the 4th International Semantic Web Conference, ISWC2005.
9. Komulainen, V., Valo, A. and Hyvönen, E. (2005) A Tool for Collaborative Ontology Development for the Semantic Web. Proceedings of the International Conference on Dublin Core and Metadata Applications, DC 2005.
10. Känsälä, T. and Hyvönen, E. (2006) 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.
11. Löfberg, L., Archer, D., Piao, S., Rayson, P., McEnery, T., Varantola, K. and Juntunen, J.–P. (2003) Porting an English Semantic Tagger to the Finnish Language. In Proceedings of the Corpus Linguistics 2003 conference, pp. 457–464. UCREL, Lancaster University.
12. Noy, N., Sintek, M., Decker, S., Crubézy and M., Fergerson, R. (2001) Creating Semantic Web Contents with Protégé–2000. IEEE Intelligent Systems 2(16):60–71.
13. Popov, B., Kitchukov, I., Angelov, K. and Kiryakov, A. (2006) Co-occurrence and ranking of entities. Available at: http://www.ontotext.com/publications/CORE_otwp.pdf
14. Reeve, L. and Han, H. (2005) Survey of Semantic Annotation Platforms. Proceedings of the 2005 ACM Symposium on Applied Computing.
15. Schreiber, G., Dubbeldam, B., Wielemaker and J.,Wielinga, B. (2001) Ontology–Based Photo Annotation. IEEE Intelligent Systems, 16(3):66–74.
16. Suominen O., Viljanen K. and Hyvönen E. (2007) User-centric Faceted Search for Semantic Portals. Proceeedings of the 4th European Semantic Web Conference ESWC2007, forth-coming.
17. Tapanainen, P. and Järvinen, T. (1997) A Non–projective Dependency Parser. Proceedings of the 5th Conference on Applied Natural Language Processing, pp. 64–71.
18. Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E. and Ciravegna, F. (2006) Semantic Annotation for Knowledge Management: Requirements and a Survey of the State of the Art. Journal of Web Semantics, 4(1):14–28.
19. Valkeapää, O. and Hyvönen, E. (2006) 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.