Collaborative Web-Publishing with a Semantic Wiki

Rico Landefeld, Harald Sack
Friedrich-Schiller-Universität Jena,
D-07743 Jena
rico.landefeld@takwa.de, sack@minet.uni-jena.de

Abstract: Semantic wikis have been introduced for collaborative authoring of ontologies as well as for annotating wiki content with semantic meta data. In this paper, we introduce a different approach for a semantic wiki based on an ontology meta model customized especially for the deployment within a wiki. For optimal usability client-side technologies have been combined with a simple semantic query language. Text fragments of a wiki page can be annotated in an interactive and rather intuitive way to minimize the additional effort that is necessary for adding semantic annotation. Thus, the productivity and efficiency of a semantic wiki system will open up for non expert users as well.

1 Introduction

The very first browser of the World Wide Web (WWW) provided a function that soon sank into oblivion again: web pages could not only be read, but also written and thus be changed directly. Some years ago, wiki systems [LC01] picked up that idea again by providing the possibility for each visitor to change the content of wiki pages. Wiki systems are lean content management systems that administrate HTML documents. The user of a wiki system is able to generate or change wiki documents only by using the facilities of a simple web browser. In this way, wiki documents are developed and maintained collaboratively by the community of all users. Wiki systems don't give formal guidelines for generating or structuring their content. This lack of formal rules might have been responsible for their fast growth of popularity as can be seen, e.g., in the free online-encyclopedia Wikipedia[1]. On the other hand, if a wiki system is growing as rapidly as Wikipedia does, lack of formal rules necessitates frequent restructuring to keep the content well arranged and usable.

Typical wiki systems only provide a limited number of functions for structuring the content. As a rule users create special pages with overviews or class systems for structuring the wiki content. But the maintenance of this manually created categorization system becomes rather expensive. Moreover, it stimulates misusage, as e.g., you may find many categories in Wikipedia that have been created to subsume entities that share merely one special feature [VKV+06]. Similar problems have been reported for intranet wikis [BG06]. In general, most of the mentioned problems in wikis can be reduced to the fact that their content is encoded in HTML (Hypertext Markup Language) or some simplified version of it.

[1] http://www.wikipedia.org

HTML only formalizes formatting and (limited) structuring of text without the possibility of formalizing any semantics that is required for automated aggregation and reuse of data.

Semantic Wikis try to combine wiki systems with semantic technology as a building block of the currently emerging semantic web [BLHL01]. They connect textual content with a knowledge model by formalizing the information of a wiki page with a formal knowledge representation language. In this way, the content of wiki pages becomes machine readable and even machine understandable. Semantic wikis show one possible way to overcome the aforementioned problems related to traditional wikis in general while at the same time enabling collaborative generation and maintenance of formal knowledge representations (ontologies). But, the arbitrary wiki user is not an expert knowledge engineer. Therefore, usability and "ease of us" become a rather important factor for designing the user interface of a semantic wiki.

Current projects have chosen different ways to deploy formal knowledge representations within a wiki. From our point of view the ratio of cost and effect is most important. The cost refers to the cognitive and factual work that the user has to invest to generate and maintain semantic annotations. On the other side, the effect subsumes all the advantages that the user might get from a system that deploys this semantic annotation. Cognitive and factual work is mostly determined by the design of the user interface and the underlying ontology meta model of the wiki. At the same time the semantic expressiveness of the annotations determines the efficiency of the achieved functionality. Therefore, the ontology meta model of a semantic wiki represents a compromise between complexity and expressiveness. In addition, the integration of semantic annotations into wiki systems demands new concepts of user interaction that help to limit the necessary effort. Existing semantic wiki systems have several deficiencies: either, their underlying ontology meta model is mapping elements of the knowledge representation language directly to wiki pages, or they are using a simplified ontology meta model that results in rather limited semantic functionality. Most projects are based on traditional wiki systems and therefore inherit also their user interaction facilities.

We propose a semantic wiki concept that combines the following three concepts: a simplified ontology meta model especially customized to be used within a wiki system, a WYSIWYG-Editor (What you see is what you get) as a user interface for both text- and ontology editing, and preferably a most simple semantic query language. A prototype of our semantic wiki *Maariwa*[2] has been successfully implemented. The paper is organized as follows: Section 2 covers related work and in particular discusses ontology meta models and user interaction concepts. In Section 3 we introduce the semantic wiki project Maariwa, while Section 4 resumes our results and discusses future work.

2 Related work

Besides the extension of traditional wikis with semantic annotation, we also have to consider approaches for collaborative authoring of ontologies based on wiki technology that

[2] http://ipc755.inf-nf.uni-jena.de:8081/Maariwa

date back to a time before the semantic web initiative even started (cf. [FFR97, SRKK97, ACFLGP01]). We therefore distinguish two different semantic wiki approaches depending on their focus either on textual (wiki) content or (formal) knowledge representation. The *Wikitology* paradigm [DRR+05b] refers to wiki systems acting as a user interface for collaborative authoring of ontologies. There, a wiki page represents a concept and hyperlinks between wiki pages represent relationships between concepts. Thus, the wiki system acts merely as tool to manipulate the ontology. In difference, so called *ontology-based wiki systems* are semantic wiki systems that focus on textual content, while using knowledge representations to augment navigation, searchability, and reusability of information.

Another differentiating factor is determined by the adaption of the ontology meta model for the use within the wiki and the coverage of the underlying knowledge representation languages (KRL). The ontology meta model of a semantic wiki defines a mapping between the elements of the KRL and the application model. Moreover, it determines the semantic expressiveness of annotation and serves as a basis for querying information. Semantic annotation can be maintained together with or separate from the textual wiki content. Next, we will introduce and discuss relevant semantic wiki implementations and their underlying ontology meta model.

Platypus Wiki [CCT04] is one of the earliest semantic wiki implementations. It maps a wiki system to an RDF (resource description framework) graph. Wiki pages represent RDF resources and hyperlinks represent RDF properties. Semantic annotation is maintained together with the textual wiki content within a separate text field as RDF(S) (RDF schema) [BG04] or OWL (web ontology language) [MvH03] in XML serialization format.

Rhizome [Sou04] supports semantic annotations by using a special Wiki Markup Language (WikiML). The entire wiki content including text, structure, and meta data internally is encoded in RDF. In contrast to traditional wiki systems, Rhizome supports a fine-grained security model. Besides manipulation of meta data Rhizome does not offer any functionality that utilizes this semantic annotation.

Rise [DRR+05a] is customized for requirement analysis in software engineering. It is based on an ontology that represents different document types and their relationships. Templates determine structure and relationships of wiki pages that represent instances of a document type. The Rise ontology can be extended by adding new templates. Semantic annotations can be edited with an extended WikiML and are used for consistency check and navigation.

Semantic MediaWiki (SMW) [VKV+06, KVV05] is an extension of the well known MediaWiki [3]. The online-encyclopedia Wikipedia is the most prominent example of a MediaWiki application. SMW aims to improve structuring and searchability of the Wikipedia content by deploying semantic technologies. Therefore, SMW tries to follow Wikipedia's user interface to attract a broad user community. SMW extends the WikiML with attributes, types, and relationships. To represent classes, SMW utilizes existing Wikipedia categories. By assigning a wiki page to a given category it becomes an instance of the class being represented by this category. In addition, SMW provides a set of units of measurement and customizable data types. Attributes, types, and relationships are represented

[3] http://www.mediawiki.org/

by own wiki pages. Semantic search is implemented in SMW with a proprietary query language (WikiQL) that closely reflects the annotation syntax.

Makna Wiki [DPST06] uses RDF triples (subject, predicate, object) for annotating wiki pages. RDF triples can be added to a text page by using a customized WikiML or within a separate form. RDF triples' subjects or objects refer to textual wiki pages that represent a concept each. Makna Wiki only supports maintenance and manipulation of instances, but no class definitions. It extends JSPWiki[4] and its WikiML with typed links and literals. Makna Wiki's semantic annotation is utilised for navigation based on RDF triples and for limited semantic search (e.g., for searching instances of classes with distinct properties).

IkeWiki [SGW05] tries to bring together application experts and knowledge engineers. Therefore, the user interface offers seperate views for textual content and semantic annotation. The annotation editor supports the assignment of classes to wiki pages and typed links for representing relationships. Furthermore, classes, properties, and resources can be freely created and manipulated. The ontology meta model closely reflects the underlying KRL (RDF(S) and OWL). The user interface for editing meta data supports automatic completion of terms.

SweetWiki [BG06] combines social tagging [GH06] and semantic technologies into what they call *semantic tagging*. Wiki pages are annotated with user tags that form not only a collective index (a so called *folksonomy* [Van05]), but a formal ontology. This is achieved by regarding each user tag as a concept of an ontology. Relationships between concepts are not determined by users, but by designated experts. The user does not interact with the ontology directly, but is merely able to create and to assign user tags. For the user, there is no distinction between instances and classes, because tags can represent both. Sweet Wiki's user interface provides a WYSIWYG editor for manipulating the wiki content.

Besides the above mentioned projects there exist several alternative semantic wiki implementations that can also be arranged within the framework given by our examples ranging from wikitologies to ontology-based wikis. As a rule, early implementations such as Platypus strictly separate textual content from knowledge representations, while later projects (besides IkeWiki, which separates annotations from content) integrate knowledge representations and textual content by utilizing customized WikiML. Above all, IkeWiki and MaknaWiki provide dynamic authoring support. The ontology meta model of PlatypusWiki, MaknaWikia and IkeWiki merely provide a direct mapping of the underlying KRL without any covering. Only MaknaWiki offers (limited) semantic search facilities. If elements of RDF(S) or OWL are utilized directly, RDF query languages such as SparQL [PS07] can be applied. IkeWiki enables data export via SparQL without providing a search interface. MaknaWiki and Platypus use elements of RDF(S) and OWL without making any use their semantic expressiveness.

Contrariwise, SMW deploys a simplified ontology meta model based on OWL-DL with an easy to use query language (compared to SparQL) The SMW user interface keeps the connection between classes and their attributes covered, but offers no further editing assistance to the user (besides the provision of templates). Sweet Wiki's ontology meta model does not distinguish between classes, instances, datatype properties, or object properties –

[4] http://jspwiki.org/

everything is mapped on tags. Therefore, information can only be provided via tag search or via direct SparQL queries with the consequence that information being distributed over several wiki pages can not be queried.

We propose the semantic wiki *Maariwa* that utilizes an ontology meta model covering the underlying OWL-Lite language that enables information reuse as well as semantic queries over distributed information. In the following chapter we introduce Maariwa's underlying concepts and give a brief sketch on its implementation.

3 The Maariwa concept – architecture and implementation

Maariwa is a semantic wiki project of the FSU Jena with the objective of implementing an augmented wiki system that enables simultaneous creation and manipulation of textual content and ontologies. Maariwa's semantic annotation is utilized to put augmented navigation and semantic search into practice for reuse and aggeregation of the wiki content. In Maariwa ontologies are used to structure the textual wiki content for providing access paths to the knowledge being represented in the wiki. Maariwa's access paths can be addressed via *MarQL*, a simple semantic query language, to enable semantic search. We therefore refer to Maariwa's underlying concept as *ontology-based web publishing*. Maariwa is geared towards user communities without expert knowledge in knowledge representation. Furthermore, Maariwa facilitates access by utilizing a WYSIWYG-editor for text and ontology manipulation.

3.1 Maariwa's Ontology Meta Modell

The Ontology meta model of Maariwa is designed to enable simple and efficient searchability, as.e.g. answering questions like "What physicists were born in the 19th century" or"Which cities in Germany have more than 100.000 inhabitants?". To answer these questions, the ontology meta model has to provide the semantic means to express these queries, while on the other hand it must be simple enough that the arbitrary user without expert knowledge can comprehend it. Tab. 1 shows a subset of OWL-Lite elements that are utilized for Maariwa's ontology meta model. We refer to datatype properties and object properties as attributes and relationships. Furthermore, both attributes and relationships are not defined as global entities but only local within their classes. This enables attributes and relationships with the same name in different classes without worrying the non expert user with naming conflicts. Attributes are determined by a datatype with an optional measuring unit. Only numbers, strings, and dates are considered for datatypes.

By integrating the ontology meta model into the wiki system, wiki pages can be annotated with concepts of ontologies to formalise their content. Classes, individuals, and sets of individuals can be described by wiki pages. A page that describes a class is associated with attributes, relationships, and superclasses. Pages that describe individuals are associated with one class at least. To denote a set of individuals, a page has to be associated with a

OWL-Lite element	Maariwa element
class	class
datatype property	attribute
object property	relationship
class instance	wiki page representing a class being istantiated as an instance-page
subClassOf	superclass /subclass
individual	wiki page describing an individual, being an instance of one or more classes
datatype property instance	attribute value of an instance-page
object property instance	relationship value of an instance-page

Table 1: Mapping of OWL-Lite elements to Maariwa's ontology meta model.

MarQL expression (see section 3.4). Typed links between pages may refer to relationships. Classes may use instance-pages as simple tags by associating neither a class hierarchy nor relationships or attributes.

3.2 Integration of semantic annotation and textual content

The Maariwa editor is designed to minimize the user's additional effort for adding and maintaining semantic annotations. Repeated input of text or ontology concepts is avoided most times simply by copying existing concepts from the page context. New concepts are created in dialogue with the user. Thus, Maariwa enables the development of textual and ontological resources in parallel. Maariwa provides two alternatives for creating semantic annotations: concepts of an ontology can be created from textual content of a wiki page, while on the other hand concepts might be defined first providing a textual description in the wiki page afterwards. Schema and instance data can be manipulated in parallel without interrupting the editing process of the wiki page (see Fig. 1). Maariwa's WYSIWIG-editor is implemented as so called *Rich Internet Application* [Loo06] that provides desktop functionality for web applications and adopts the role of the traditional WikiML-based user interface. In Maariwa, semantic annotations are directly displayed within the wiki page (see. Fig. 2). Different colors denote the semantics of typed hyperlinks. Text that contains attribute values as well as links that represent relationships is highlighted. In addition, tooltips (i.e., pop-up information windows) display the semantic of an annotated text fragment if is is touched with the mouse pointer. Navigation within class hierarchies and class relations is enabled with a superimposed class browser. The semantic annotation of each wiki page can be separately accessed and exported in RDF/XML encoding via an own URL. Also export and import of ontologies as a whole is supported.

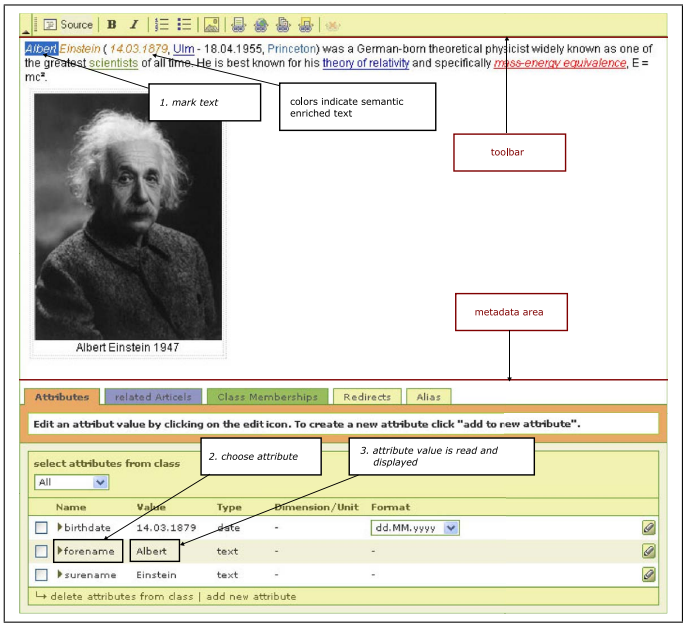

Figure 1: Editing of a Maariwa wiki page with textual content (above) and ontologies (below).

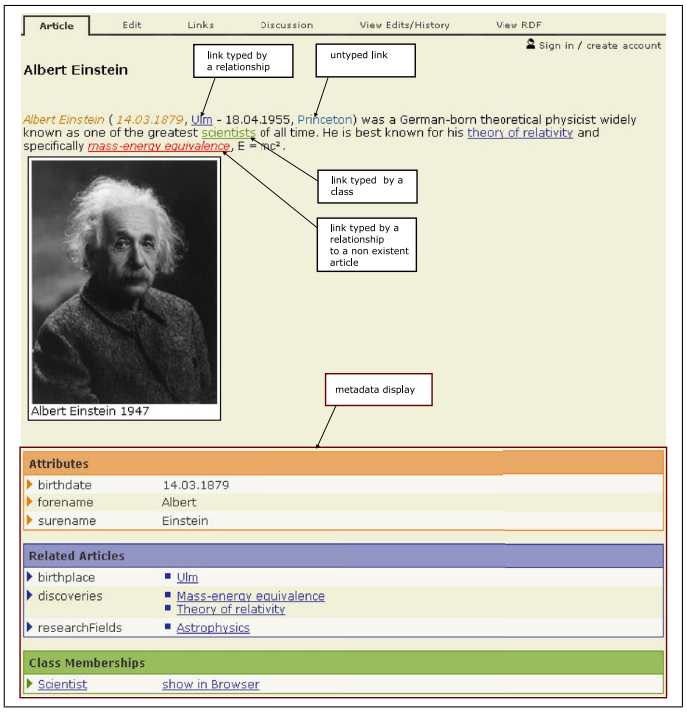

Figure 2: A wiki page with semantic annotation in Maariwa.

3.3 The MarQL semantic query language

The syntax of SparQL reflects the characteristics of the RDF data model. RDF data are represented as triples and the RDF document can be interpreted as a graph. SparQL traverses the RDF graph and as result delivers the nodes that satisfy the constraint given in the SparQL query. Because RDFS and QWL are based on the RDF syntax SparQL can also be used to query RDFS- and OWL-files, but without exploiting their semantic expressiveness.

MarQL is a customized semantic query language for the Maariwa ontology meta model. MarQL syntax does not refer to RDF triples but directly addresses ontology elements such as classes, attributes and relationships. The underlying RDF encoding of the data remains hidden. In comparison to SparQL, the syntax of MarQL is much more compact but less flexible. MarQL only implements a fixed set of query patterns. A MarQL query results in a set of wiki pages that refer to individuals, which satisfy the constraints of the MarQL query. The structure of a MarQL query can be shown with an example: the expression *Scientist.institution.location.country = Germany* refers to wiki pages about scientists that work at an institute being located in Germany. *Scientist* refers to a class with a relationship *institution*. Relationships can be applied recursively and are denoted as a path expression. In this way, *institution* and *location* are connected. This means, that there must exist a class, which is target class of a relationship with *institution*, while in addition having a relationship with *location* . In the same way *location* and *country* are connected, while *country* can either be an attribute or a relationship and therefore *Germany* might denote an individual or an attribute value. MarQL provides logical operators as well as string operators and operators for comparison. E.g., the query *City.population* $\geq$ *100.000* results in a set of all wiki pages that describe cities with more than 100.000 inhabitants, or *Scientist.birthdate* $<$ *1.1.1900 AND Scientist.birthdate* $\geq$ *1.1.1800* results in a set of wiki pages with scientists that are born in the 19th century. The latter example results in a list of instances of a class and can also be referred to as a simple *tag*.

3.4 Maariwa architecture and implementation

Maariwa does not extend any traditional wiki system but is a proprietary development based on Java. The application core of Maariwa implements a service level that realizes a wiki system as a set of loosely coupled services (see Fig. 3). Besides data management and revision control of the wiki pages and the related ontologies, keyword search and semantic search are also implemented as services. MarQL queries are translated into SparQL queries. Manipulation of the ontologies in the editor is organized in different dialog levels. Within a dialog updates of one or more objects can be performed. These updates can either be revoked or they will also be adopted in the subjacent levels. Therefore, the service level offers cascading manipulation levels that implement this functionality with the help of local copies and snapshots. Backend data processing is achieved with a relational database management system. The service level stores ontology elements in various RDF triple stores and all other objects in separate database tables. The service level is used by the components of the web front-end that constitute Maariwa's user interface. Web server

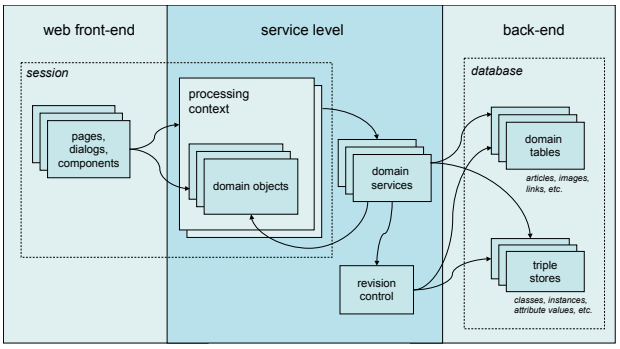

Figure 3: Architecture of the Maariwa system.

and browser client communicate asynchronously via AJAX [Gar05] to achieve better usability. The WYSIWYG-editor represents the text of the wiki page as XHTML fragment and is based on Javascript.

As a rule, wiki systems deploy a revision control system (RCS) to prevent abuse of unprotected write access. Maariwa adapts this RCS for ontologies, too. The RCS maintains two levels: schema level and instance level. The schema level comprises all changes on classes, relationships and attributes, while the instance level covers changes of individuals including their attribute values and relationship values. Both levels are tightly coupled, because each schema update might have effects on all derived individuals. Therefore, a schema version additionally includes all updates on the instance level that occurred since the last update of the schema. Each version of a wiki page besides the update of the page content also contains updates of the associated concepts.

4 Conclusion

In this paper, we have introduced the semantic wiki approach Maariwa based on an ontology meta model customized especially for the deployment within a wiki. For optimized usability recent client-side technologies have been combined with a simple semantic query language. The user can annotate text fragments of a wiki page in an interactive and rather intuitive way to minimize the additional effort that is necessary for adding semantic annotation. Thus, the productivity and efficiency of a semantic wiki system will open up for non expert users as well. The ontology meta model enables the formulation of access paths to wiki pages as well as the reuse of already implemented relationships in the underlying knowledge representation. The simple query language MarQL uses Maariwa's annotations and the contained access paths for implementing a semantic search facility. Ontology meta model and query language are synchronized to support annotations on different levels of expressiveness. Thus, enabling simple tags as well as complex ontologies with attributes and relationships.

Currently, Maariwa is extended to include meta data also directly within the textual wiki content, as e.g., tables with self-adjusting data aggregations. Also natural language processing technology is considered to be utilized in the WYSIWYG-editor for automated suggestions as well as for translating natural language queries into MarQL. For deploying a semantic wiki, the user considers always the ratio of cost and effect that is caused by the additional effort of providing semantic annotation. In doing so, it is not important whether the creation of semantically annotated textual content or the creation of mere ontologies is focussed, but how both can be integrated within an system that provides a reasonable and efficient interface for user access.

References

[ACFLGP01] J. C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez. WebODE: a Scalable Workbench for Ontological Engineering. In *1st Int. Conf. on Knowledge Capture (KCAP01).Victoria. Canada*, 2001.

[BG04] D. Brickley and R. V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. W3c recommendation, RDF Core Working Group, W3C, 2004.

[BG06] M. Buffa and F. Gandon. SweetWiki: semantic web enabled technologies in Wiki. In *Proc. of the 2006 international symposium on Wikis*, pages 69–78, 2006.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[CCT04] S. E. Campanini, P. Castagna, and R. Tazzoli. Platypus Wiki: a Semantic Wiki Wiki Web. In *Semantic Web Applications and Perspectives, Proc. of 1st Italian Semantic Web Workshop*, 2004.

[DPST06] K. Dello, E. Paslaru, B. Simperl, and R. Tolksdorf. Creating and using Semantic Web information with Makna. In *Proc. of the 1st Workshop on Semantic Wikis – From Wiki To Semantics*. ESWC2006, 2006.

[DRR+05a] B. Decker, E. Ras, J. Rech, B. Klein, and C. Höcht. Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis. In *Workshop on Semantic Web Enabled Software Engineering (SWESE), at ISWC 2005, Galway, Ireland*, 2005.

[DRR+05b] B. Decker, J. Rech, E. Ras, B. Klein, and C. Hoecht. Self-organized Reuse of Software Engineering Knowledge supported by Semantic Wikis. In *Workshop on Semantic Web Enabled Software Engineering (SWESE), at ISWC 2005*, 2005.

[FFR97] A. Farquhar, R. Fikes, and J. Rice. The Ontolingua server: A tool for collaborative ontology construction. *Int. Journal of Human-Computer Studies*, 46(6):707–727, 1997.

[Gar05] J. J. Garrett. Ajax: A New Approach to Web Applications, 2005. `http://www.adaptivepath.com/publications/essays/archives/000385.php`.

[GH06] S. Golder and B. A. Huberman. The Structure of Collaborative Tagging Systems. *Journal of Information Sciences*, 32(2):198–208, April 2006.

[KVV05] M. Krötzsch, D. Vrandecic, and M. Völkel. Wikipedia and the semantic web - The missing Links. In *Proc. of the 1st Int. Wikimedia Conf., Wikimania*, 2005.

[LC01] B. Leuf and W. Cunningham. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley, 2001.

[Loo06] C. Loosley. Rich Internet Applications: Design, Measurement, and Management Challenges. Technical report, Keynote Systems, 2006.

[MvH03] D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language: Overview. Working draft, World Wide Web Consortium, March 2003.

[PS07] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF (Working Draft). Technical report, W3C, March 2007.

[SGW05] S. Schaffert, A. Gruber, and R. Westenthaler. A Semantic Wiki for Collaborative Knowledge Formation. In *Semantics 2005, Vienna, Austria*, 2005.

[Sou04] A. Souzis. Rhizome Position Paper. In *Proc. of the 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, 2004.

[SRKK97] B. Swartout, P. Ramesh, and T. Russ K. Knight. Toward Distributed Use of Large-Scale Ontologies. In *Symp. on Ontological Engineering of AAAI. (Stanford, California, March)*, 1997.

[Van05] T. Vander Wal. Folksonomy Explanations, 2005. `http://www.vanderwal.net/random/entrysel.php?blog=1622`.

[VKV+06] M. Völkel, M. Krötzsch, D. Vrandecic, H. Haller, and R. Studer. Semantic Wikipedia. In *Proc. of WWW 2006, Edinburgh, Scotland*, 2006.