# OntoWiki – A Tool for Social, Semantic Collaboration

Sören Auer[1,2], Sebastian Dietzold[2], and Thomas Riechert[2]

[1] University of Pennsylvania, Department of Computer and Information Science
Philadelphia, PA 19104, USA,
auer@seas.upenn.edu
[2] Universität Leipzig, Institut für Informatik, Augustusplatz 10-11,
D-04109 Leipzig, Germany,
{lastname}@informatik.uni-leipzig.de

**Abstract** We present OntoWiki, a tool providing support for agile, distributed knowledge engineering scenarios. OntoWiki facilitates the visual presentation of a knowledge base as an information map, with different views on instance data. It enables intuitive authoring of semantic content, with an inline editing mode for editing RDF content, similar to WYSIWYG for text documents. It fosters social collaboration aspects by keeping track of changes, allowing to comment and discuss every single part of a knowledge base, enabling to rate and measure the popularity of content and honoring the activity of users. Ontowiki enhances the browsing and retrieval by offering semantic enhanced search strategies. All these techniques are applied with the ultimate goal of decreasing the entrance barrier for projects and domain experts to collaborate using semantic technologies. In the spirit of the Web 2.0 OntoWiki implements an "architecture of participation" that allows users to add value to the application as they use it. It is available as open-source software and a demonstration platform can be accessed at http://3ba.se.

We present a tool supporting agile Knowledge Engineering in a pure Web environment. It is called OntoWiki since it is close in the spirit to existing Wiki systems. Technologically however, the OntoWiki design is independent and complementary to conventional Wiki technologies. As such, the OntoWiki approach differs from recently emerged strategies to integrate Wiki systems and the Semantic Web (cf. [6,5,8,11,12]). In these works it is proposed to integrate RDF triples into Wiki texts in a special syntax. It is a straightforward combination of existing Wiki systems and the Semantic Web knowledge representation paradigms. However, we see the following obstacles:

- *Usability:* The main advantage of Wiki systems is their unbeatable usability. Adding more and more syntactic possibilities counteracts ease of use for editors.
- *Redundancy:* To allow the answering of real-time queries to the knowledge base statements have to be stored additionally in a triple store. This introduces a redundancy complicating the implementation.

– *Scalability:* Knowledge base changes which involve statements with different subjects will scale very bad since all corresponding Wiki texts have to be parsed and changed.

The OntoWiki strategy presented in this paper, on the contrary, does not try to mix text editing with knowledge engineering, instead it applies the Wiki paradigm of "making it easy to correct mistakes, rather than making it hard to make them"[9] to collaborative knowledge engineering. The main goal of the OntoWiki approach thus is to rapidly simplify the presentation and acquisition of instance data from and for end users. This is achieved by regarding knowledge bases as "information maps". Each node at the information map is represented visually and intuitively for end users in a generic but configurable way and interlinked to related digital resources. Users are further enabled to enhance the knowledge schema incrementally as well as to contribute instance data agreeing on it as easy as possible to provide more detailed descriptions and modelings. Consequently, the following requirements have been determined for OntoWiki:

– *Intuitive display and editing* of instance data should be provided in generic ways, yet enabling means for domains specific views.
– *Semantic views* allow the generation of different views and aggregations of the knowledge base.
– *Versioning and Evolution* provides the opportunity to track, review and selectively roll-back changes.
– *Semantic search* facilitates easy-to-use full-text searches on all literal data, search results can be filtered and sorted (using semantic relations).
– *Community support* enables discussions about small information chunks. Users are encouraged to vote about distinct facts or prospective changes.
– *Online statistics* interactively measures the popularity of content and activity of users.
– *Semantic syndication* supports the distribution of information and their integration into desktop applications.

In the remainder of the paper we propose strategies on how to put these requirements into effect in a real-life system and report about implementation in a prototypical OntoWiki on the basis of Powl [1], a framework for Semantic Web application development. To stress the generic character of OntoWiki, the figures in this paper show screenshots of the OntoWiki prototype with a knowledge base collaboratively developed[3] and containing information about scientific conferences, as well as another publicly available knowledge base[4] containing information about research projects, people and publications at a research institute.

[3] at http://3ba.se
[4] http://www.aifb.uni-karlsruhe.de/viewAIFB_OWL.owl

## 1 Visual Representation of Semantic Content

The compromise of, on the one hand, providing a generic user interface for arbitrary RDF knowledge bases and, on the other hand, aiming at being as intuitive as possible is tackled by regarding knowledge bases as "information maps". Each node at the information map, i.e. RDF resource, is represented as a Web accessible page and interlinked to related digital resources. These Web pages representing nodes in the information map are divided into three parts: a left sidebar, a main content section and a right sidebar. The left sidebar offers the selection of content to display in the main content section. Selection opportunities include the set of available knowledge bases, a class hierarchy browser and a full-text search. Once a selection is made, the main content section will arrange matching content in a *list view* linking to *individual views* for individual instances (cf. 1). The right sidebar offers tools and complementary information specific to the selected content.

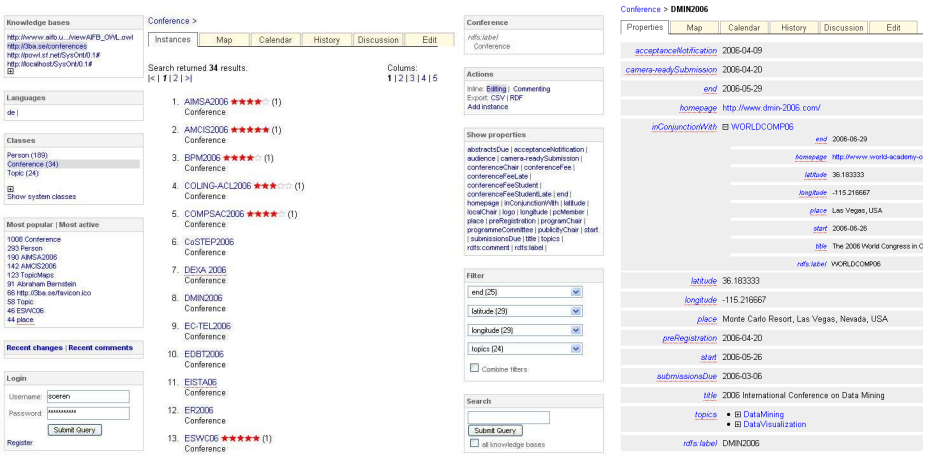

**Figure 1.** List view (left) and view of an individual instance with expanded inline reference view (right).

*List views* allow to view a selection of several instances in a combined view. The selection of instances to display can be either based on class membership or on the result of a selection by facet or full-text search. OntoWiki identifies those properties used in conjunction with the instances of the selection. The display of the corresponding property values for each instance can be switched on, thus resulting in a tabular view. Furthermore, each individual instance displayed is linked to an individual view of that instance.

*Individual views* combine all the properties attached to an particular instance. Property values pointing to other individuals are rendered as HTML links to

the corresponding individual view. Alternatively, to get information about the referenced individual without having to load the complete individual view it is possible to expand a short summary (loaded per AJAX) right where the reference is shown. The right sidebar provides additionally information about similar instances (of the same type) and incoming links (i.e. references from other instances).

**Different Views on Instance Data** The OntoWiki prototype facilitates different views on instance data. Such views can be either domain specific or generic. Domain specific views have to be implemented as plug-ins. Generic views provide visual representations of instance data according to certain property values. The following views are currently implemented:

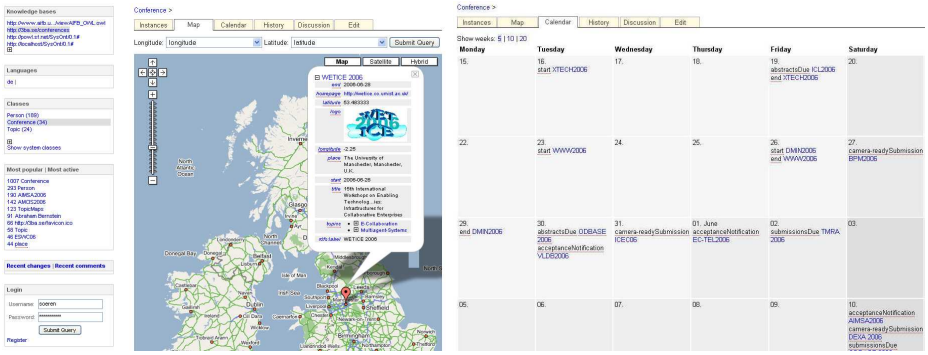

**Figure 2.** Map view (left) and calendar view (right) of instance data.

*Map View.* If the selected data (either a single instance or a list of instances) contains property values representing geographical information (i.e. longitude and latitude coordinates) a map view provides information about the geographical location of the selected data (cf. Figure 2). Technically, this view is realized by integrating the Google Maps API[5]. However, the integration is bi-directional, since objects displayed in the map can be expanded and instance details are dynamically fetched from the knowledge base and displayed directly within the map view. The selection of instances to be displayed can be furthermore the result of a facet-based filtering (cf. Section 4).

*Calendar View.* Instances having property values with the associated datatype `xsd:date` can be displayed in a calendar view (cf. Figure 2). As for the map view the selection of instances displayed in the calendar view can be the result of a facet-based filtering. Each item displayed is linked to the individual view of

[5] http://www.google.com/apis/maps/

the corresponding instance. The sidebar offers a link to export calendar items in iCal format, which enables to import the selected calendar items into a desktop calender application.

## 2 Collaborative Authoring

To enable users to edit information presented by the OntoWiki system as intuitively as possible, the OntoWiki approach supports two complementary edit strategies for the knowledge base:

- *Inline editing*, the smallest possible information chunks (i.e. statements) presented in the OntoWiki user interface are editable for users.
- *View editing*, common combinations of information (such as an instance of a distinct class) are editable in one single step.

Both editing strategies are supported by a mixed client and server side *concept identification and reuse technology and a library of* generic editing widgets. In the remainder of this section the editing strategies and their supporting technologies are presented in more detail.

### 2.1 Inline Editing

For human users it is important that the statements of a knowledge base are presented on the user interface in a way facilitating the efficient reception of this information. To achieve this goal information should be ordered and grouped and as a result of this information appearing redundant should be omitted. If the context clarifies, for example, that some information describes a distinct concept (e.g. since the OntoWiki page for a person was accessed) the concept will be displayed only once on the OntoWiki user interface, even though all the statements describing the concept contain the concepts URI reference as subject. If furthermore a property (e.g. referencing publications) occurs multiple times (e.g. since the person described is author of multiple publications) those statements should be grouped together and the label of the property should be displayed only once (cf. Figure 3).

Even though such a human friendly representation of the statements contained in the knowledge bases conceals the original statement structure the OntoWiki system is aware which information displayed on the user interface originated from what statements. To enable users to rapidly edit or add statements as soon as they notice mistakes or missing information OntoWiki features an inline editing mode. This means that all information originating from statements presented on the OntoWiki user interface is equipped with a small edit button as well as an add button (cf. Figure 3). After clicking one of those buttons a resource editor (cf. Figure 4) is loaded and the corresponding statement can be easily edited or a similar content (i.e. a statement with same subject and predicate) can be added.

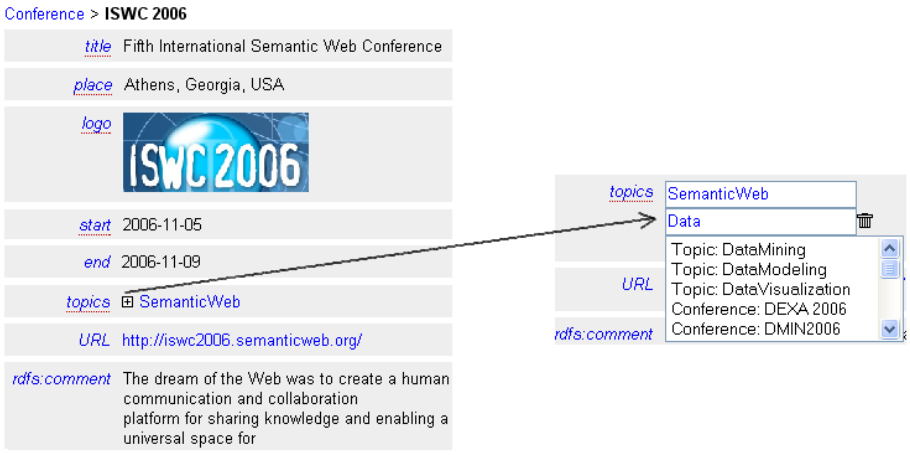


**Figure 3.** OntoWiki instance display with statement edit buttons (left). Statement editor with interactive search for predefined individuals based on AJAX technology (right).

This strategy can be seen analogous to the WYSIWYG (What You See Is What You Get) editing strategy for text editing, since information can be edited in the same environment as it is presented to users.

### 2.2 Concept Identification and Reuse

Knowledge bases become increasingly advantageous, if once defined concepts (e.g. classes, properties, or instances) are as much reused and interlinked as possible. This especially eases the task of rearranging, extracting and aggregating knowledge. To become part of the daily routine for even inexperienced and rare users of the OntoWiki system already defined concepts should be suggested to the user, whenever he is requested to contribute new information. In a Web based environment and for highly scalable knowledge bases conventional Web technologies were the major obstacles for this.

Conventional Web technologies do not support large data sets to be handled at the client (browser) side. But this is usually needed when working with large knowledge bases. To overcome this limitation, reloading of web pages becomes necessary. This approach is time consuming and requires multiple user interactions. Recently, with the deployment of more sophisticated Web browsers, supporting modern JavaScript and XML technologies, mixed server and client side web applications became possible. These were recently named AJAX (Asynchronous JavaScript and XML) and early adopters such as Google-Maps[6] or Flickr[7] make extensive use of them.

[6] http://maps.google.com
[7] http://www.flickr.com

The OntoWiki uses the AJAX technology to interactively propose already defined concepts while the user types in new information to be added to the knowledge base (cf. Figure 3). To realize this interactive search, all URI references and literals are indexed for full-text searches in the statement repository.

### 2.3 Editing Widgets

For convenient editing of differently typed literal data the OntoWiki system provides a library of reusable user interface components for data editing, called widgets. Such widgets are implemented in a server side programming language (e.g. PHP), they generate HTML fragments together with appropriate Cascading Style Sheet definitions and optionally JavaScript code. They may be customized for usage in specific contexts by widget configurations. The following widgets are currently provided by the prototypical OntoWiki implementation:

- *Statements*: allows editing of subject, predicate, and object.
- *Nodes*: edit literals or resources.
- *Resources*: select and search from/for existing resources
- *Literals*: literal data in conjunction with a data type or a language identifier.
- *Widgets for specific literal data types*: e.g. dates, HTML fragments.
- *File widget*: allows uploading of files to the OntoWiki system.

All widgets can be configured. The OntoWiki system allows to define and attach certain sets of configurations to a specific widget. In addition to widget specific configurations, generic widget configuration which should be applicable to all widgets includes HTML attributes such as `class`, height and width of the widget, or arbitrary CSS styles.

A widget selection connects a widget configuration with a context. Contexts are the data type of the literal to be edited, the property of the statement which's object is edited, the property in conjunction with a specific class, the knowledge base the node to be edited belongs to, as well as the editing user and her group.

### 2.4 View Editing

Editable views are combinations of widgets to edit a specific view on the knowledge base in one single step. The OntoWiki system provides the following types of editable views:

- *Metadata*: comments, labels, and annotations (such as versioning and compatibility information) which can be attached to arbitrary resources are combined in a metadata view.
- *Instances*: An instance view combines all properties attached to the instance's class or one of the super-classes. For large knowledge bases this might include a large amount of properties. The OntoWiki system thus allows to restrict the view to such properties which are really used in conjunction with other instances of the same class. On the basis of range definitions

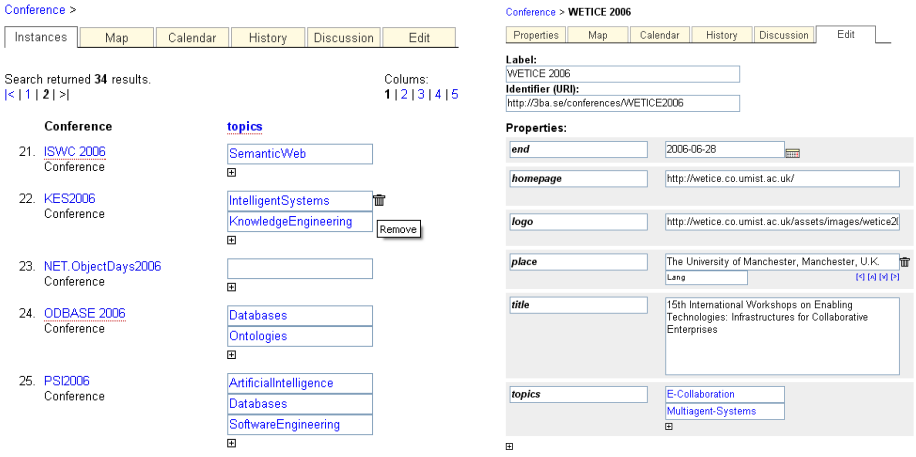


**Figure 4.** Editing of a property's values at many instances at once (left). Dynamically generated form combining different widgets based on an OWL class definition (right).

for the property, OntoWiki selects appropriate editing widgets. Additional properties can be added on-the-fly, the system will ask the user in a next step to specify the property's characteristics (e.g. domain, range, cardinality restrictions).

– *Views*: The earlier described inline-editing technique allows to edit arbitrary views. The columns of list views arranging many instances in a tabular way for example can be easily edited at once, thus allowing to rapidly add "horizontal" knowledge (across several instances) to the knowledge base (cf. Figure 4).

## 3 Enabling Social Collaboration

A major aim of OntoWiki is to foster and employ social interactions for the development of knowledge bases. This eases the structured exchange of meta-information about the knowledge base drastically and promotes collaboration scenarios where face-to-face communication is hard. Making means of social interactions as easy as possible furthermore contributes in creating an "architecture of participation" that allows users to add value to the system as they use it. Social collaboration within OntoWiki is in particular supported by:

*Change tracking.* All changes applied to a knowledge base are tracked. OntoWiki enables the review of changes on different levels of detail (see also [3]) and optionally restricted to a specific context, such as changes on a specific instance, changes on instances of a class, or changes made by a distinct user. In addition to present such change sets on the Web, users can subscribe to get information

about the most recent changes on objects of their interest by email or RSS/Atom feeds.

*Commenting.* All statements presented to the user by the OntoWiki system may be annotated, commented, and their usefulness can be rated. This enables community driven discussions, for example about the validity of certain statements. Technically, this is implemented on the basis of RDF reifications, which allow to make statements about statements. Small icons attached to an object of a statement within the OntoWiki user interface indicate that such reifications exist (cf. Figure 5). Positioning the mouse pointer on such an icon will immediately show up a tool tip with the most recent annotations; clicking on the icon will display them all.


**Figure 5.** Comments attached to statements.

*Rating.* OntoWiki allows to rate instances. Users have to be registered and logged into the system to participate in order to avoid duplicate ratings by the same user. However, a user may change his rating for a certain instance. Special annotation properties allow the creation of rating categories with respect to a certain class. Instances of the class can then be rated according to these categories, thus allowing for example the rating of instances of a class publication according to categories originality, quality and presentation.

*Popularity.* All accesses to the knowledge base are logged thus allowing to arrange views on the content based on popularity. As with ratings or user activity, the popularity of content can be measured with respect to a certain knowledge base or fragment of it (e.g. popularity with respect to class membership). This enables users to add value to the system as they use it.

*Activity/Provenance.* The system keeps record of what was contributed by whom. This includes contributions to the ontology schema, additions of instance data or commenting. This information can be used to honor active users in the context of the overall system, a specific knowledge base or a fragment of it (e.g. instance additions to some class). This way it contributes to instantly gratify users for their efforts and helps building a community related to certain semantic content.

## 4 Semantic Search

To realize the full potential of a semantic browsing experience the semantic structuring and representation of content should be employed to enhance the retrieval of information for human users. OntoWiki implements two complementary strategies to achieve this goal.

### 4.1 Facet-based Browsing

Taxonomic structures give users exactly one way to access the information. Furthermore, the development of appropriate taxonomic structures (whether e.g. class or SKOS keyword hierarchies) requires significant initial efforts. As a pay-as-you-go strategy, facet-based browsing allows to reduce the efforts for a priori knowledge structuring, while still offering efficient means to retrieve information. Facet-based browsing was also implemented by the Longwell Browser[8] for RDF data and it is widely deployed in the shape of tagging systems of the Web 2.0 folksonomies. To enable users to select objects according to certain facets, all property values (facets) of a set of selected instances are analyzed. If for a certain property the instances have only a limited set of values, those values are offered to restrict the instance selection further. Hence, this way of navigation through data will never lead to empty results. The analyzing of property values though can be very resource demanding. To still enable fast response times the OntoWiki system caches the results of of a property value analysis for later reuse and invalidates those cache objects selectively if values of the respective property are updated (see [2, Chapter 5] for details).

### 4.2 Semantically Enhanced Full-text Search

OntoWiki provides a full-text search for one or multiple keywords occurring in literal property values. Since there can be several property values of a single individual containing the search string the results are grouped by instances. They are ordered by frequency of occurrence of the search string. Search results may be filtered to contain only individuals which are instances of a distinct class or which are described by the literal only in conjunction with a distinct property (cf. Figure 6).

A semantic search has significant advantages compared to conventional full-text searches. By detecting classes and properties, contain matching instances, the semantic search delivers important feedback to the user how the search may be successfully refined.

The semantic search is currently implemented as a search in the local RDF store. In conjunction with a crawler, which searches, downloads, and stores arbitrary RDF documents from the web, OntoWiki can be easily transformed in a Semantic Web search engine.

[8] http://simile.mit.edu/longwell/

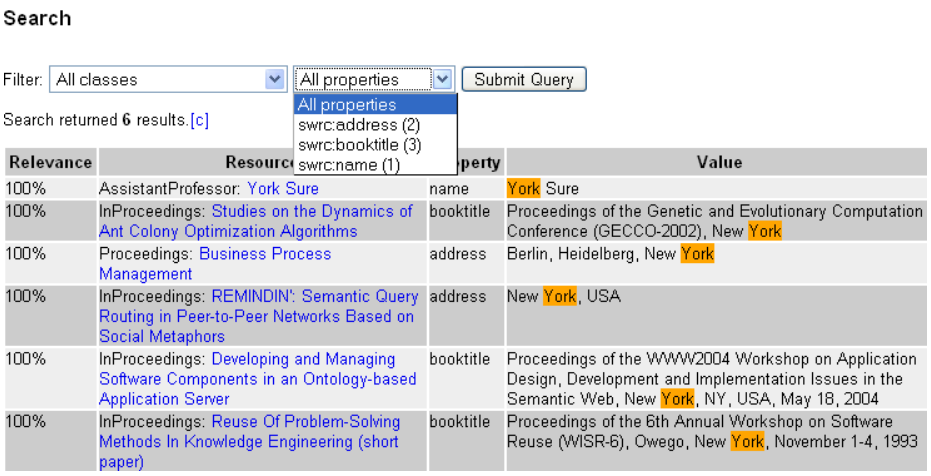


**Figure 6.** User interface for the semantic search in the OntoWiki system. After a search for "York" it suggested to refine his search to instances with one of the properties `swrc:address`, `swrc:booktitle` or `swrc:name`.

## 5 Implementation and Status

OntoWiki is implemented as an alternative user interface to the schema editor integrated in Powl. Powl is a platform for Semantic Web application development realized in a 3-tier architecture consisting of storage tier, object-oriented API and user interfaces (cf. Figure 7). Many of the requirements for OntoWiki were gathered from use cases of Powl.

OntoWiki was implemented in the scripting language PHP, thus allowing to be easily deployed on most Web hosting environments. The application is available as open-source software from SourceForge[9]. A publicly available knowledge repository on the basis of OntoWiki is available at http://3ba.se.

The system is designed to work with knowledge bases of arbitrary size (only limited by disk space). This is achieved by loading only those parts of the knowledge base into main memory which are required to display the information requested by the user on the screen (i.e. to render a Web page containing this information).

Currently, OntoWiki is extended and adopted within a variety of R&D projects. The project SoftWiki[10] for example is developing a prototype based on OntoWiki, which aims to employ OntoWiki's social collaboration functionality for end-user driven Requirements Engineering of massively distributed software development projects. For the project Orchestra [7] OntoWiki's storage, browsing and retrieval functionality is envisioned to be used as a shared repository for

[9] http://powl.sf.net
[10] http://softwiki.de

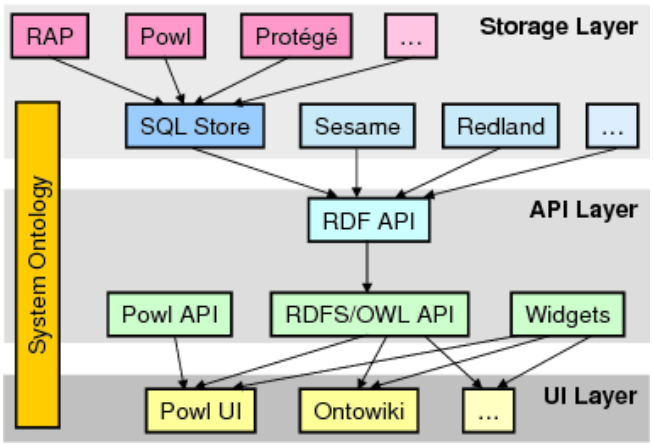


**Figure 7.** Architecture of Powl and OntoWiki.

ontologies and queries in the bio-informatics domain. In the project "Vernetzte Kirche" [4] Powl and parts of OntoWiki were applied to foster a meta-data initiative for social, cultural and religious content.

## 6 Conclusion

In this paper we presented the OntoWiki approach, exemplary exhibiting how tool support for agile, collaborative knowledge engineering scenarios can be provided. Since the OntoWiki system is based technologically on Powl, we stressed in this paper especially aspects facilitating the usage of the OntoWiki system. These include the visual presentation of a knowledge base as an information map, social collaboration aspects as well as a semantic search strategy. Such efforts, which decrease the entrance barrier for domain experts to collaborate using semantic technologies, are in particular crucial to gain a maximized impact on collaborative knowledge engineering. Examples from other domains, such as Community Content Management and Software Development, showed that such efforts can have an enormous impact on distributed collaboration, thus enabling completely new products and services. Conventional Wiki technologies for example radically simplified the editing process and enabled the Wikipedia project[11] to attract a multitude of editors finally succeeding in the creation of the worlds largest encyclopedia. Technologies for distributed collaborative software development as CVS and Subversion[12] or the SourceForge[13] platform made it possible to develop almost any standard software for private or business needs

[11] http://wikipedia.org
[12] http://subversion.tigris.org
[13] http://sf.net

largely in absence of strong, centralized, commercial corporations. The aim of OntoWiki is to contribute giving the Semantic Web a much broader basis.

*Application domain.* The OntoWiki system is technologically independent of and complementary to conventional text Wiki systems. It enables the easy creation of highly structured content by distributed communities. The following points summarize some limitations and weaknesses of OntoWiki and thus characterize the application domain:

- *Environment*: OntoWiki is a Web application and presumes all collaborators working in a Web environment, possibly spatially distributed.
- *Usage Scenario*: OntoWiki focuses on knowledge engineering projects where a single, precise usage scenario is either initially not (yet) known or not (easily) definable.
- *Reasoning*: Application of reasoning services is (initially) not mission critical.
- *Budget*: Only a small amount of financial and personnel resources are available.

*Open issues and potential future work.*

- Implement a privilege system and access control for and on the basis of the RDF data model with support for rights management on higher conceptual levels than that of statements.
- Obtain more case studies, in particular independent comparisons, are needed to provide further evidence to see whether OntoWiki lives up to its promises.
- Examine possibilities to tighter integrate the Description Logic reasoning services into OntoWiki.
- Establish better methods of interaction with existing content and knowledge management systems.

*Further related work.* In addition to the affinity with Wiki systems and Web portals in general the OntoWiki approach can be seen as a representative of a new class of semantic portals (cf. [13]). The SEAL SEmantic portAL [10] for example exploits semantics for providing and accessing information at a portal as well as constructing and maintaining the portal. Due being based on a rather static methodology [14] it focuses less on spontaneous, incremental enhancements of the knowledge base than OntoWiki. Another approach to develop a semantic portal is the website of the Mindswap project[14]. Semantic Web knowledge representation standards are used as primary data source and for interoperability, the editing and publishing process as well as collaboration aspects however seem to be either not tackled or publicised.

## Acknowledgments

This research was supported in part by the following grants: BMBF (SE2006 #01ISF02B), NSF (CAREER #IIS-0477972 and SEIII #IIS-0513778).

[14] http://www.mindswap.org/first.shtml

## References


1. Sören Auer. Powl: A Web Based Platform for Collaborative Semantic Web Development. In Sören Auer, Chris Bizer, and Libby Miller, editors, *Proceedings of the Workshop Scripting for the Semantic Web*, number 135 in CEUR Workshop Proceedings, Heraklion, Greece, 05 2005.
2. Sören Auer. *Towards Agile Knowledge Engineering: Methodology, Concepts and Applications*. PhD thesis, Universität Leipzig, 2006.
3. Sören Auer and Heinrich Herre. A Versioning and Evolution Framework for RDF Knowledge Bases. In *Proceedings of Ershov Memorial Conference*, 2006.
4. Sören Auer and Bart Pieterse. "Vernetzte Kirche": Building a Semantic Web. In *Proceedings of ISWC Workshop Semantic Web Case Studies and Best Practices for eBusiness (SWCASE05)*, 2005.
5. David Aumüller. Semantic Authoring and Retrieval within a Wiki (WikSAR). In Demo Session at the Second European Semantic Web Conference (ESWC2005), May 2005. Available at http://wiksar.sf.net, 2005.
6. David Aumüller. SHAWN: Structure Helps a Wiki Navigate. In *Proceedings of the BTW-Workshop "WebDB Meets IR"*, 2005.
7. Zachary G. Ives, Nitin Khandelwal, Aneesh Kapur, and Murat Cakir. ORCHESTRA: Rapid, collaborative sharing of dynamic data. In *CIDR*, pages 107–118, 2005.
8. Markus Krötzsch, Denny Vrandecic, and Max Völkel. Wikipedia and the Semantic Web - The Missing Links. In Jakob Voss and Andrew Lih, editors, *Proceedings of Wikimania 2005, Frankfurt, Germany*, 2005.
9. Bo Leuf and Ward Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley Professional, 2001.
10. Alexander Maedche, Steffen Staab, Nenad Stojanovic, Rudi Studer, and York Sure. SEmantic portAL: The SEAL approach. In Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web*, pages 317–359. MIT Press, 2003.
11. Eyal Oren. SemperWiki: A Semantic Personal Wiki. In Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann, editors, *Proc. of Semantic Desktop Workshop at the ISWC, Galway, Ireland, November 6*, volume 175, November 2005.
12. Adam Souzis. Building a Semantic Wiki. *IEEE Intelligent Systems*, 20(5):87–91, 2005.
13. Steffen Staab, Jürgen Angele, Stefan Decker, Michael Erdmann, Andreas Hotho, Alexander Maedche, Rudi Studer, and York Sure. Semantic Community Web Portals. In *Proc. of the 9th World Wide Web Conference (WWW-9)*, Amsterdam, Netherlands, 2000.
14. Steffen Staab, Rudi Studer, Hans-Peter Schnurr, and York Sure. Knowledge processes and ontologies. *IEEE Intelligent Systems*, 16(1):26–34, 2001.