

# Semantic Wikis for Personal Knowledge Management\*

Eyal Oren<sup>1</sup>, Max Völkel<sup>2</sup>, and Stefan Decker<sup>1</sup>

<sup>1</sup> DERI Galway, Ireland

`first.last@deri.org`

<sup>2</sup> Institute AIFB, Universität Karlsruhe, Germany

`voelkel@aifb.uni-karlsruhe.de`

**Abstract.** Wikis are successful tools for collaborative information collection. Wikis are becoming popular knowledge management tools, but do not fully support the requirements for such tools, namely structured search and knowledge reuse. Adding semantic annotations to Wikis helps to address these limitations by offering advanced information access (navigation and querying) and allowing knowledge reuse (through embedded queries and semantic information exchange). We present an architecture for Semantic Wikis and present our prototype SemperWiki.

## 1 Introduction

Wikis are highly successful tools for collaborative information collection, noted for example in the quality of Wikipedia compared to traditional encyclopedia [3]. Wikis are lately becoming very popular tools for personal and organisational knowledge management as well<sup>3</sup>. But Wikis were not originally designed for personal knowledge management: they lack functionality for structured search and knowledge reuse. Semantic Web technologies can augment Wikis in exactly those weak spots of knowledge structuring and reuse. This paper introduces a Semantic Wiki architecture and implementation that supports these requirements.

### 1.1 Requirements for personal knowledge management

Knowledge is fundamentally created by individuals [7, p. 59]. Supporting individuals in their personal knowledge management is therefore crucial. Considering the knowledge creation spiral [7, p. 62–73], knowledge workers require support in:

1. **authoring:** codifying knowledge into information to share with others
2. **finding and reminding:** finding and reminding existing knowledge [14]

---

\* Submitted to *DEXA 2006*; please do not distribute.

<sup>3</sup> in DERI and AIFB we use several Wikis to manage our knowledge in projects, clusters, industry collaborations, and standard groups. See e.g. <http://twiki.org/cgi-bin/view/Main/TWikiStories> for anecdotal evidence.

3. **knowledge reuse:** combining an existing body of knowledge [5]
4. **collaboration:** developing ideas through social interactions
5. **cognitive adequacy:** convincing workers to use knowledge tools requires balancing effort with perceived personal benefit.

## 1.2 Limitations of current solutions

*Traditional tools* such as todo lists or paper piles are very common [4]. They are suitable for authoring but do not support finding, reminding, knowledge reuse, or collaboration. *Hierarchical filing* (of emails and files) allows browsing and (full-text) searching, but does not support authoring, knowledge reuse, reminding, and collaboration. *Personal information management* tools (e.g. MS Outlook) manage email, calendar, and tasks. They support finding and reminding, but do not support authoring, knowledge reuse, and collaboration.

## 1.3 Wikis for knowledge management

Wikis are collaborative hypertext environments, focused on open access, ease-of-use, and modification [6]; popular for their simplicity and easy access [2]. Wiki syntax is simple and allows creation of links and textual markup of lists and headings. Wikis commonly support binary data attachments, versioning and change management, change notification, full-text search, and access control.

Wikis are lately becoming popular tools for personal and organisational knowledge management. Knowledge workers are using them individually, organisations deploy them internally, and many project organisations collaborate through restricted-access Wikis. But considering the requirements of personal knowledge management, Wikis do support authoring and collaboration, but do not enable knowledge reuse, and have only limited support for finding and reminding:

**Structured access** of a Wiki is not possible, which arises when one is browsing or searching for information. For example, one cannot currently **query** Wiki systems, because the information is not structured but rather is textual. For example, users looking for “*How old is John Grisham?*”, “*Who wrote the Pelican Brief?*”, or “*Which European authors have won the Nobel price for literature?*” cannot ask these questions directly. Instead, they have to navigate to the page that contains this information and read it themselves. For more complicated queries that require some background knowledge users need to manually combine the knowledge from several sources.

Another example of structured access to information can be found in page **navigation**: Wikis allow users to easily make links from one page to other pages, and these links can then be used to navigate to related pages. But these explicit links are actually the only means of navigation<sup>4</sup>. If no explicit connection is made between two related pages, e.g. between two authors

---

<sup>4</sup> except for back-references that appear on a page and show pages that reference it.

that have the same publishing company, then no navigation will be possible between those pages.

**Knowledge reuse** of Wiki content is not possible; this limitation becomes apparent when aggregations or translations of content need to be provided:

Reusing information through **reference and aggregation** is common in the real world. Consider for example that books are generally written by an author and published by the author's publisher. The books authored by John Grisham (on his page) should therefore also automatically appear as books published by Random House (on their page). But creating such a view is currently not possible in a Wiki, and instead the information has to be copied and maintained manually.

In current Wikis it is either assumed that people will speak a common language (usually English) or that **translations** to other languages will be provided. But manually translating pages is a maintenance burden, since the Wiki system does not recognise the structured information inside the page text. For example, a page about John Grisham contains structured information such as his birth date, the books he authored, and his publisher. Updates to this information have to be migrated manually to the translated versions of this page.

Both limitations result from a lack of structure in the Wiki content: almost all information is written in natural language, and has little machine-understandable semantics. For example, a page about the author John Grisham could contain a link to the page about his novel "The Pelican Brief". The English text would say that John Grisham wrote the Pelican Brief, but that information is not machine-understandable, and can therefore not be used for querying, navigating, translating, or aggregating any information.

## 2 Semantic Wikis

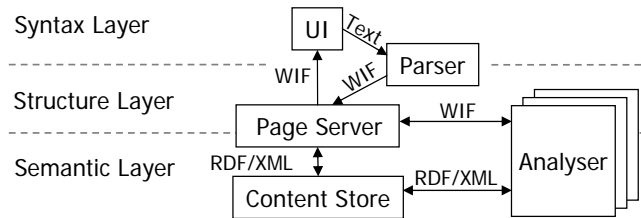
A Semantic Wiki allows users to make formal descriptions of resources by annotating the pages that represent those resources. Where a regular Wiki enables users to describe resources in natural language, a Semantic Wiki enables users to additionally describe resources in a formal language. The authoring effort is relatively low: the semantic annotations are very similar to the layout or structural directives that are already in widespread use in ordinary Wikis.

Using the formal annotations of resources, Semantic Wikis offer additional features over regular Wikis. Users can query the annotations directly ("show me all authors") or create views from such queries. Also users can navigate the Wiki using the annotated relations ("go to other books by John Grisham"), and users can introduce background knowledge to the system ("all poets are authors; show me all authors").

In designing a Semantic Wiki system several architectural decisions need to be taken. In this section, we explain the basic architecture and outline the design choices and their consequences.

## 2.1 Architecture Overview

A Semantic Wiki consists (at least) of the following components: a user interface, a parser, a page server, a data analyser, and a data store, as shown in figure 1. First we introduce each component, then we discuss the information access, the annotation language, and the ontological representation of the Wiki.



**Fig. 1.** Architecture of a Semantic Wiki

**overview:** The *page server* encapsulates the business logic (e. g. what belongs to a page) and exposed its data in a syntax neutral format (WIF<sup>5</sup>). The *user interface* lets the user browse and query the wiki pages. If a page is edited, the WIF is converted to wiki syntax and the changed wiki syntax is converted by the *parser* back to WIF. The *content store* stores all data as RDF, allowing for the full power of RDF query languages. The *analyser* can interact with the page server and the content store. It's task is to augment pages and RDF with automatically found relations. Different kind of analysers fit into the architecture, e. g. using reasoning or statistics.

**user interface:** responsible for all user interaction. If the Wiki is web-based (the classical model), then the user interface is a web server. A desktop application can also act as the user interface component. In this case, collaboration is achieved by using a shared content store.

The user interface allows users to type text and annotations in a freely intermixed fashion. The user interface shows terms from shared *ontologies*, enabling users to browse for an appropriate term<sup>6</sup>.

**page server:** includes ordinary wiki functionality mentioned in section 1.3, such as version management, binary attachments, and access control.

**parser:** converts the text written by the user into objects: it parses the text for semantic annotations, layout directives, and links. This is transmitted in the wiki-syntax neutral format WIF.

**content store:** is responsible for storing and retrieving the semantic annotations, and for exchanging data with other information systems (such as other semantic wikis). An an off-the shelve RDF triple store can be used.

<sup>5</sup> [http://www.wikisym.org/wiki/index.php/WSR\\_3](http://www.wikisym.org/wiki/index.php/WSR_3)

<sup>6</sup> Descriptions can be shared and understood if written in a common terminology, and browsing ontologies helps finding an appropriate common term.

**data analyser:** responsible for computing a set of related resources from a given page. In a regular Wiki, this means just to find all back-references, i.e. pages that link to the current one. In a semantic environment, the relations between resources are much richer. The data analyser uses the annotations about the current page, found by the parser, and searches for relevant relations in the content store (such as “other books by current author”, or “other persons that have the same parents as the current one”).

## 2.2 Annotation language

For the user of a Semantic Wiki, the most visible change compared to conventional Wikis is the modified annotation language. For Semantic Wikis the annotation language is not only responsible for change in text style and for creating links, but also for the semantic annotation of Wiki pages and for writing embedded queries in a page.

**Annotation primitives** As in conventional Wikis, internal links are written in CamelCase or by enclosing them in brackets; external links are written as full absolute URIs, or are abbreviated using namespace abbreviations.

syntax	meaning
<code>rdf:type foaf:Person</code>	page has <code>rdf:type foaf:Person</code>
<code>dc:topic [http://google.com]</code>	page has <code>dc:topic http://google.com</code>
<code>dc:topic TodoItem</code>	page has <code>dc:topic http://wikinamespace/TodoItem</code>
<code>dc:topic ‘todo’</code>	page has <code>dc:topic “todo”</code>
<code>?s dc:topic ?o</code>	embedded query for all pages and their topics
<code>?s dc:topic TodoItem</code>	embedded query for all todo items

**Table 1.** Annotation syntax

The additional syntax for semantic annotations is shown in table 1: annotations are written on a separate line, and consist of a predicate followed by an object. Predicates can only be resources (identifiable things), objects can be either resources or literals. An example page is displayed in figure 2. It describes John Grisham, an author published by Random House.

<b>JohnGrisham</b>
John Grisham is an author and retired lawyer.
<code>rdf:type foaf:Person</code>
<code>dc:publisher RandomHouse</code>

**Fig. 2.** Example page

**Subject of annotations** Wiki pages often refer to real-world resources. Annotations can refer to a Wiki page but also to the resource described on that page. For example, the triple “JohnGrisham created-on 1955-02-08” can refer to the creation date (birth date) of the person or to the creation date of the Wiki page about that person.

The question “what do URIs exactly identify” (of which the annotation subject is a subclass) is an intricate open issue on the Semantic Web<sup>7</sup>: a URI can for example identify an object, a concept, or a web-document.

Our approach is to explicitly distinguish the “page” and the “real world” resource that it describes. Since we expect more annotations of the real-world resource (e.g. John Grisham) than annotations of the page itself, we attribute annotations by default to the real-world resource, and allow annotations about the page (such as its creation date, version, or author) to be made by prepending annotations with an exclamation mark.

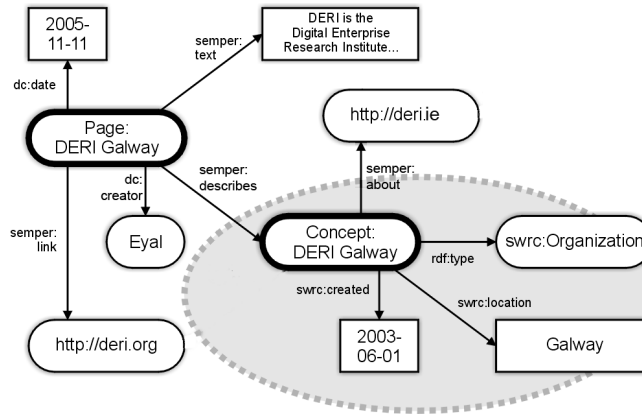
<b>DERI Galway</b>
DERI Galway is one location of the Digital Enterprise Research Institute, researching Semantic Web technology; our main page is at <a href="http://deri.org">http://deri.org</a> .
rdf:type swrc:Organization swrc:location "Galway" swrc:created "2003-06-01" semper:about urn://deri.ie !dc:creator EyalOren

**Fig. 3.** Annotating real-world resources

For example, figure 3 shows a page that describes the research institute DERI. The annotations state that DERI is a research institute, founded in June 2003, and located in Galway. The last annotation, prepended with an exclamation mark, refers to the page instead of the resource; it states that Eyal Oren is the creator of that page. The page uses the “semper:about” predicate to relate the concept that the page is discussing (DERI Galway) to the existing identifier (<urn://deri.ie>).

Figure 4 shows the RDF graph that is generated from the page in figure 3. The page “DERI Galway” has a property **text**, it has a **creator** and **creation date** that are populated by the Wiki system, and it contains a navigational hyperlink to <http://deri.org>. The concept “DERI Galway” is the actual subject of the annotations: it is a research organisation, with a certain creation date and location.

<sup>7</sup> see <http://www.w3.org/DesignIssues/HTTP-URI.html>.



**Fig. 4.** Corresponding RDF graph about DERI Galway

**Embedded queries** Users can embed queries on any Wiki page. These embedded queries are executed when a page is visited, and their results are included in the displayed page<sup>8</sup>. They could for example show aggregations (such as all the books written by John Grisham); embedding queries in page allows knowledge reuse by persistently combining pieces from different sources.

As shown in table 1, embedded queries are written using triple patterns, sequences of subject, predicate, object, that can contain variables (names that start with a question mark). A triple pattern is interpreted as a query: triples matching the pattern are returned. Patterns can be combined to form joins. Figure 5 shows the earlier example page about John Grisham, including an embedded query at the bottom of the page. The query returns all books written by JohnGrisham; it creates a view on the data that is displayed below the page text.

### 2.3 Information access

Information can be accessed by structured navigation and querying facilities.

**Navigation** Navigation in ordinary Wikis is limited to explicit links entered by users; it is not possible to navigate the information based on structural relations. A Semantic Wiki provides the metadata necessary to navigate the information in a structured way. For example, knowing that John Grisham is an author, we can show all other authors in the system, and offer navigation to them.

<sup>8</sup> Views resulting from embedded queries could be read-only or editable. Editable views cause some maintenance issues (should the change be recorded in the version history of the result page or of the page affected by the edit) similar to the view-update problem in databases.

<b>JohnGrisham</b>
John Grisham is an author and retired lawyer.
<pre> rdf:type foaf:Person dc:publisher RandomHouse </pre>
<pre> this query shows all his books: ?book dc:creator JohnGrisham </pre>
<pre> TheFirm dc:creator JohnGrisham TheJury dc:creator JohnGrisham ThePelicanBrief dc:creator JohnGrisham </pre>

**Fig. 5.** Page showing embedded query

Our approach for structural navigation is based on faceted meta-data browsing [15]. In faceted browsing, the information space is partitioned using orthogonal conceptual dimensions (facets) of the data, which can be used to constrain the relevant elements in the information space. For example, a collection of art works can consist of facets such as type of work, time periods, artist names, geographical locations, etc.

Common faceted browsing approaches construct the facets manually for a specific data collection. In a Semantic Wiki users are free to add arbitrary meta-data, prohibiting manual facet generation. We are developing a technique for automatically generating and clustering facets based on [10]; see [9] for details.

**Querying** We distinguish three kinds of querying functionality: *keyword search*, *queries*, and *views*:

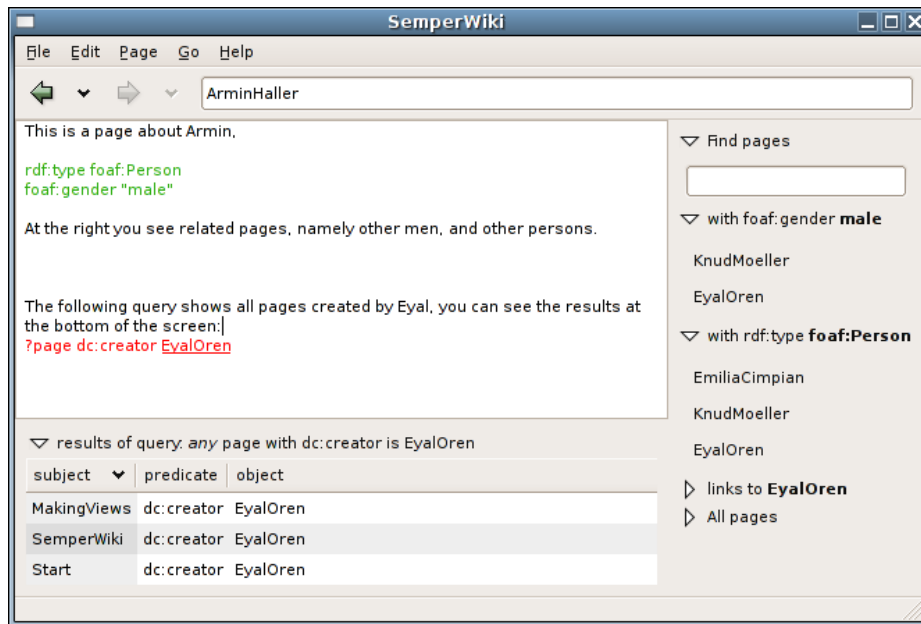
1. A *keyword-based full-text* search is useful for simple information retrieval, and supported by all conventional Wiki systems.
2. *Structured queries* use the annotations to allow more advanced information retrieval. The user can query the Wiki for pages (or resources) that satisfy certain properties. To retrieve for example all authors one can query for “?x type author”. Triple patterns can be combined to form database-like joins: “?x type author and ?x has-publisher ?y” retrieves all authors and their publishing companies.
3. As discussed earlier, users can create persistent searches by *embedding queries* in pages. A query included on a page is executed each time the page is visited and continuously shows up-to-date query results.

### 3 Implementation

SemperWiki<sup>9</sup> is our prototype implementation of a Semantic Wiki. It was first developed as a desktop application [8], and is currently being ported to the web, while still conforming to the same architecture.

<sup>9</sup> <http://semperwiki.org/>

We give only a brief overview of the implementation, see [8] for details. Figure 6 shows a screenshot from the desktop version, displaying a page about Armin Haller. The page freely intermixes natural language and simple semantic annotations stating that he is a male person. On the right hand side related items are shown based on the semantic annotations. Users are offered more intelligent navigation based on the metadata, in addition to the explicit links between pages. On the bottom of the page we see an embedded query, that shows a continuously up-to-date view of all pages created by Eyal Oren.



**Fig. 6.** Navigating and Information reuse

SemperWiki addresses the noted limitations of ordinary Wikis. Concerning **structured access**, users can find related information through associative browsing: the Wiki analyses the semantic relations in the data and provides navigational links to related information. Users can search for information using *structured queries*, in addition to simple full-text search.

Concerning **information reuse**, the semantic annotations allow better translation and maintenance; the annotations are language independent<sup>10</sup> and can be understood and reused without barriers. Users can also write *embedded queries*, creating saved searches (database views). These views can be revisited and reused, and provide a consistent picture of structured information. Furthermore

<sup>10</sup> annotations are language independent if ontologies contain label translations, but that is easier than translating general knowledge.

all information is represented in RDF using standard Semantic Web terminologies which allows information exchange.

## 4 Related Work

Souzis [11] describes an architecture for Semantic Wikis but focuses on annotating and representing page structure while we are concerned with page content, and discusses specific implementation decisions rather than generic architecture choices.

Other approaches improve Wikis by augmenting them with semantic annotations. Platypus [12] is a Wiki with semantic annotations, but adding and using annotations requires significantly more effort than normal text. Both WikSAR [1] and Semantic Wikipedia [13] offer easy-to-use annotations, but neither allow reuse of existing Semantic Web terminologies, and both only allow simple annotations of the current page (thereby exclude blank nodes).

## 5 Conclusion

Wikis are successful for information collection, but do not fully satisfy the requirements of personal knowledge management. We have shown how Semantic Wikis augment ordinary Wikis: using the metadata annotations they offer better information access (through structured navigation such as faceted browsing and structured queries) and knowledge reuse (through embedded queries and information exchange). We have implemented our architecture in a first prototype; we plan to validate its usability in a future user study.

## References

1. D. Aumueller and S. Aurer. Towards a semantic wiki experience - desktop integration and interactivity in WikSAR. In S. Decker *et al.*, (eds.) *Proc. of the 1st Workshop on the Semantic Desktop*. 2005.
2. A. L. Burrow. Negotiating access within wiki: a system to construct and maintain a taxonomy of access rules. In *HyperText '04*, pp. 77–86. 2004.
3. J. Giles. Internet encyclopaedias go head to head. *Nature*, 438:900–901, 2005.
4. S. R. Jones and P. J. Thomas. Empirical assessment of individuals' 'personal information management systems'. *Beh. & Inf. Techn.*, 16(3):158–160, 1997.
5. A. Kidd. The marks are on the knowledge worker. In *CHI '94: Proc. of the SIGCHI conf. on Human factors in computing systems*, pp. 186–191. ACM Press, 1994.
6. B. Leuf and W. Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley, 2001.
7. I. Nonaka and H. Takeuchi. *The Knowledge-Creating Company*. Oxford University Press, New York, 1995.
8. E. Oren. SemperWiki: a semantic personal Wiki. In S. Decker *et al.*, (eds.) *Proc. of the 1st Workshop on the Semantic Desktop*. Nov. 2005.
9. E. Oren and R. Delbru. Automated construction of facets for semi-structured navigation. Tech. rep., DERI, 2006. To appear.

10. V. Sinha and D. R. Karger. Magnet: Supporting navigation in semistructured data environments. In *SIGMOD*. 2005.
11. A. Souzis. Building a semantic wiki. *IEEE Intelligent Systems*, pp. 87–91, Sep. 2005.
12. R. Tazzoli, P. Castagna, and S. E. Campanini. Towards a semantic wiki wiki web. In *Proc. of the ISWC*. 2004.
13. M. Völkel, M. Krötzsch, D. Vrandečić, and H. Haller. Semantic wikipedia. In *WWW2006*. May 2006.
14. S. Whittaker and C. Sidner. Email overload: exploring personal information management of email. In *CHI '96: Proc. of the SIGCHI conf. on Human factors in computing systems*, pp. 276–283. ACM Press, 1996.
15. K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *CHI '03: Proc. of the SIGCHI conf. on Human factors in computing systems*, pp. 401–408. ACM Press, 2003.