Personal Knowledge Management with Semantic Wikis

Max Völkel[1] and Eyal Oren[2]

[1] Institute AIFB, Universität Karlsruhe, Germany
voelkel@aifb.uni-karlsruhe.de
[2] DERI Galway, Ireland
eyal.oren@deri.org

Abstract. Managing knowledge is crucial in our economy. We derive requirements on personal knowledge management (finding, reminding, collaboration, knowledge re-use and cognitively adequate interfaces) from cognitive psychological research and analyse the limitations of current solutions. We introduce a RESTful, wiki-based, open architecture for semantic personal knowledge management that fulfills the analysed requirements to a high extent and gives the user a uniform way to work with knowledge on all layers (syntax, structure, formal semantics). We discuss architectural considerations and describe two implementations.

1 Introduction

Managing and enabling knowledge is key to success in our economy and society, [26, p. 6]. Knowledge is fundamentally created by individuals [11, p. 59]. Supporting individuals in their *personal* knowledge management is therefore crucial. Current tools for personal knowledge management have limitations: analog approaches are not automated and cannot be searched, digital approaches are restrictive, do not support ad hoc structures, and do not support associative browsing.

Our contribution is the application of Semantic Web and wiki technologies to personal knowledge management, giving individuals personal benefit. In this approach we do not use wiki technology as a community platform but as a personal authoring environment. We call the result an SPKM tool, a *semantically enhanced personal knowledge management* tool, shown in Fig. 1. Each individual

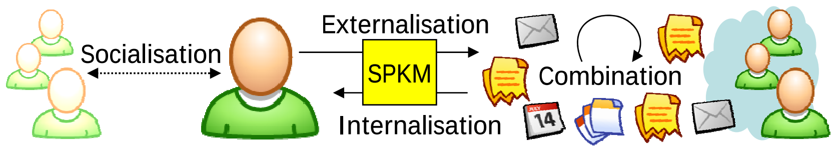

Fig. 1. Semantic Personal Knowledge Management supporting externalisation (authoring) and internalisation (learning) of personal knowledge.

uses his own SPKM tool as a personal knowledge repository; he benefits personally from this system by having better retrieval and reminding of his knowledge. His personal wiki is connected to other applications and other wikis; this network allows individuals to combine their knowledge through sharing and exchanging.

If designed correctly, SPKM tools are easy-to-use and *cognitively adequate* for modelling and refactoring knowledge; enable *information sharing* and reuse within the personal knowledge space, with other knowledge workers, and with existing information systems; and enable *structured access* to personal and collaborative knowledge through queries, categorisation, and associative browsing.

This paper is organised as follows. We analyse the situation in Sec. 2: we describe knowledge management, identify requirements for personal knowledge management, and discuss existing technical solutions. In Sec. 3 we describe our conceptual solution that addresses the analysed requirements of personal knowledge management. In Sec. 4 we introduce our architecture that enables this solution and discuss various architectural issues, and describe existing implementations of our architecture. We evaluate our architecture against other approaches in Sec. 5, and we conclude in Sec. 6.

2 Analysis

In this section, we analyse knowledge and knowledge workers, and derive requirements to support these knowledge workers. We shortly discuss limitations of current personal knowledge management solutions in terms of the requirements.

2.1 Personal Knowledge

Knowledge is "justified true belief"[11, p. 21]: it is a personal belief justified by information. Organisational knowledge management, crucially important in our knowledge society [21], consists of amplifying the individual knowledge and crystallising it as part of the organisation.

Individuals are continuously personally committed to knowledge creation [15,10]. This personal commitment relies on the intentions and autonomy of individuals [10]. Intention defines the understanding and actions of an individual, autonomy gives him the self-motivation and freedom to absorb and create knowledge. Personal autonomy is crucial for knowledge creation.

There are two types of knowledge, tacit knowledge and explicit knowledge [16]. Explicit knowledge is transmittable: articulated in a formal language and codified into information. Tacit knowledge has a personal quality and is hard to formalise; it is embedded in individual experience and consists of insights, intuition, and skills.

Knowledge is created through conversions between tacit and explicit knowledge [11, p. 62–73], as shown earlier in Fig. 1: (a) different sources of explicit knowledge can be *combined* to form new knowledge; (b) tacit knowledge can be *externalised* into explicit knowledge, through metaphors and codification; (c) explicit knowledge is *internalised* into tacit knowledge by acting, doing, and learning; and (d) tacit knowledge can be transferred through socialisation – without language, but through observation, imitation, and practise. As socialisation is often not an option in online environments, *externalisation and internalisation of knowledge become the bottleneck of the knowledge society.*

2.2 Knowledge Layers

We can identify five layers in computerised knowledge processing: *raw*, *addressable*, *syntax layer*, *structure layer* and *semantic layer*.

- on the *raw layer* resources (e.g. images) are just binary data. They cannot be interpreted, addressed, or searched.
- on the *resource layer* resources (e.g. files) have a unique identifier. They can be addressed, linked to, and possibly retrieved (by resolving their identifier).
- on the *syntax layer* resources (e.g. emails, text files) can be interpreted (e. g. encoding, charset) and their syntax can be understood. Their (textual) content can be indexed and keyword queries are possible.
- on the *structure layer* resources (e.g. HTML pages, word processor files, or databases) have a structured format. The structure can be interpreted and exploited to create a better user interface (e.g. bold text, drag-and-drop, tree views). Structured queries (e.g. SQL) are possible for known parts of the data model.
- on the *semantic layer* the structures of resources are described explicitly (using formal ontologies). Data structures are self-describing: they are annotated with statements on how to interpret and use them.

These layers can be traversed; the mappings from syntax to structures (parser), from structures to syntax (user interfaces), and from semantics to structures (data models), are well explored. The transition from the structural, implicit, semantics to the semantic, explicit, layer is however less explored.

For example, John might write a document, and mark some section in red; these markings mean "statements do be discussed with Alice". If he exchanges the document he needs to communicate these *ad-hoc semantics* (discuss with Alice) implied by structures (red) to the receiver. Currently, he would do so separately, by email or phone. Such a separate explanation limits automated knowledge exchange. *Conceptual Data Structures* (CDS) make the intended semantics of such structures explicit, allowing the receiver to see the semantics behind John's markup.

In a similar way, user-perceived *document semantics* (e. g. the nesting of items) can be exploited[3]. This allows documents to be mapped to RDF data structures and then back to different formats like outlines or slide shows. Tables are a particular example of structures with semantics, which are quite difficult to formalise [14].

[3] see CDS ontology at `http://www.xam.de/2005/12/cds`

2.3 Requirements for personal knowledge management

We can make the different activities in knowledge management more concrete by examining the requirements to support knowledge workers. Knowledge workers produce new information by combining an existing body of knowledge [7]. "Ideas are formed in the minds of individuals and are developed in social interactions" [10]. This process is not linear but, as shown in Fig. 2, a continuous interplay between capturing, organising, formalising, and retrieving knowledge.

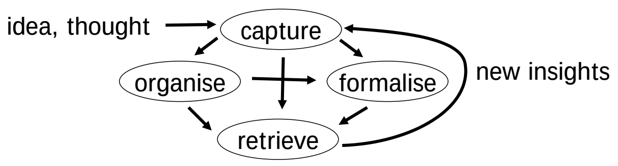

Fig. 2. Managing Ideas

Supporting knowledge workers therefore means to support individual internalisation (learning) and externalisation (writing), and to support their interaction, their socialisation, their sharing of knowledge. Since in online collaboration true socialisation is not an option, knowledge has to be shared explicitly through externalisation and internalisation. From the knowledge conversion types we derive requirements for personal knowledge management (supporting knowledge workers), shown in Fig. 3. We discuss each requirement and identify sub-requirements.

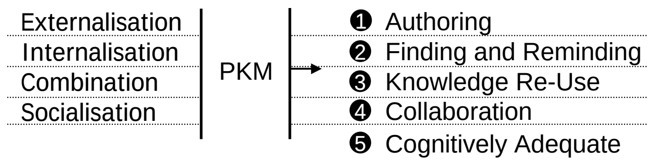

Fig. 3. Main requirements for PKM

Authoring follows from the need to externalise knowledge. To minimise effort, authoring should be simple and cognitively adequate. Authoring should be possible on all three knowledge layers (syntax, structure, and semantics), and should be integrated across these levels. We thus need a **uniform access** of all knowledge layers (syntax, structure, semantics). As depicted in Fig. 5, semantic wikis encompass all knowledge types, offering a uniform way to author and query unstructured text, structured text, and formal semantic data.

Second, we need **soft transitions** between different knowledge layers, rewarding every little effort in structuring and formalising knowledge with better retrieval performance.

Different forms of knowledge authoring can be positioned on a continuum in invested effort and returned benefit (see Fig. 4). For example, knowledge written as free text requires little effort but provides also little benefit, the information is unstructured and cannot be retrieved and reused efficiently; tagging texts with keywords requires slightly more effort and provides slightly improved retrieval; and formal ontology languages require significant authoring effort (authors are restricted in their possibilities and have to follow specific rules) but also provides significant benefits: automated support for knowledge retrieval, reuse, and reasoning. Note that current systems do not offer soft transitions between different knowledge types, e. g. there is no benefit from a half-done ontology.

Third, since the marks made by the knowledge worker (such as words, pictograms, but also red underlining or italics) contain much information but are often only understandable by himself, we need to capture and reproduce all the marks accurately [7]. This is part of authoring on the syntactical level. Summarising, we have the following requirements:

1. Syntactical authoring
2. Structural authoring
3. Semantic authoring
4. Integrated authoring

Finding and reminding is important for internalisation: one needs to find knowledge in order to learn about it, and one needs to be reminded (notified) of forgotten knowledge [27]. Finding involves two phases: recall-directed search and recognition-based scanning [8]. For search the properties (metadata) of information are important, for scanning the spatial layout and the physical marks [7] are important. Querying is thus important, both using full-text keywords and using metadata, and the results should be displayed for scanning.

Since people are passive in finding information [23], a tool should not rely on search (pull). It is thus important to automatically present related information (push) and to allow browsing of the information.

To present related information, knowledge needs to be categorised. Since filing and categorising is cognitively difficult [27], we need to support both manual and automatic clustering and classification. However, since it is not clear how people categorise information [2, p. 295–303], automatic classification should be adaptive and employ various techniques in parallel. Related items should be displayed for scanning.

5. Query
6. Associative browsing
7. Clustering
8. Notification

Knowledge reuse enables the combination of existing knowledge into new knowledge. Reuse includes composing knowledge items out of smaller items; this composition can be manual (as in copy-and-paste) or synchronised (as views in databases). Reuse of knowledge also includes applying existing background knowledge to your knowledge base (rules), and reusing (standardised) terminology to reach a common ground and enable understanding.

9. Composition
10. Rule execution (inferencing)
11. Terminology reuse

Collaboration enables combining and sharing knowledge. For collaboration a communication infrastructure (to be able to transport knowledge) is necessary. The infrastructure should guarantee privacy and security of private data. To enable collaboration with other (knowledge) tools (emails, word processor, etc.) interoperability is necessary between these applications. In ongoing collaborations it is important to track and manage context of information [27].

12. Communication
13. Privacy
14. Interoperability

15 Context management

Cognitive adequacy is a general requirement to balance the personal effort and perceived personal benefit: individuals should have a tool that is adequate for their personal mental model. Cognitive adequacy can be achieved by adaptive interfaces that adjust to their user. It is important to grant the user authoring freedom and not impose constraints on the knowledge organisation [18].

16. Adaptive interfaces
17. Authoring freedom

2.4 Limitations of current solutions

Current solutions for personal knowledge management are: physical piles of papers and notes, hierarchies in emails and files, and personal information management (PIM) tools such as Microsoft Outlook.

Physical piles of paper (possibly organised in physical folders) are very common and suitable for authoring: they capture and reproduce the physical marks, they allow recognition-based recall, and they maintain the spatial layout of the knowledge [8]. However, they do not support finding and reminding, knowledge reuse, or collaboration.

Hierarchical filing (of emails and files) is common in the personal computer. The information can be browsed and searched (although usually limited to full-text keyword search). However, hierarchical filing and retrieving is suboptimal [27]. Furthermore no support is offered for authoring, knowledge reuse, reminding, and collaboration (mostly context management and interoperability are lacking).

PIM tools manage email, calendar, and tasks. They support finding and reminding very well, but do not offer any support for authoring, knowledge reuse, and collaboration (again context management and interoperability are lacking).

3 Solution

Our solution is an SPKM system consisting of a semantic wiki enhanced in multiple ways to support all our requirements. In this section we give a high-level overview of such an SPKM system, in the next section we introduce the architecture and implementations of our system.

Classical wikis [9] are integrated, easy-to-use hypertext environments [12] with three defining characteristics "easy contribution", "easy editing", and "easy linking". They allow simultaneous authoring in different knowledge layers (from free-text to structure), each layer introduces an increased authoring effort but also an increased benefit. The importance of flexible authoring methods has been recognised in knowledge engineering [3].

As shown in Fig. 4, semantic wikis enhance classical wikis with the ability to author and use formal semantics: in the same wiki style of free text editing, semantic statements can be added. They can describe the page or parts thereof semantically. A semantic wiki thus offers a uniform way to work with all knowledge layers, as shown in Fig. 5: they allow users to structure and annotate their data but they do not force them to do so.

Using enhanced wiki syntax (plain-text with few markup commands and few semantic annotation commands) has several benefits: (a) most users are used to text typing and avoid familiarising with yet another user interface; (b) existing skills for text manipulation (e. g. copy and paste of text blocks) are leveraged to edit a document structure; (c) users refine interactively the input until the result matches the intended structure; (d) wikis allow soft transitions between knowledge layers including free-text, therefore no knowledge of any syntax is required to start authoring; (e) wiki syntax has little layout options, forcing the user to focus on the structure and the content; (f) text is in general a faster method of entering semi-structured information than graphical approaches.

The SPKM system builds upon semantic wikis, and enhances them to support all our requirements. Using enhanced wiki syntax we allow **authoring** on all knowledge layers. A powerful search (both keyword search and structural

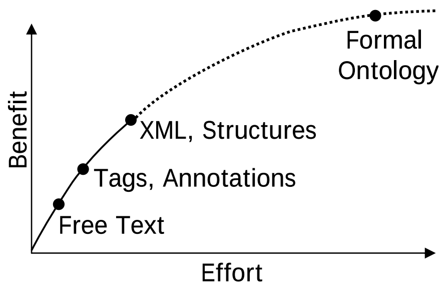

Fig. 4. Semantic wikis handle a continuous spectrum of knowledge types.

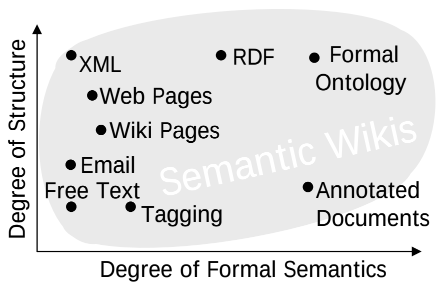

Fig. 5. Semantic Wikis for Uniform Authoring, Re-Use and Retrieval

queries) allows **finding** information. Part of the user interface shows items related to the current information, allows **associative browsing**. The system automatically **analyses** and **clusters** the information to find related items. One can set reminder dates to pages (or define sophisticated reminder rules) and be notified of these events.

All information is stored in a content repository that can apply **rules** with background knowledge, and one is supported in **reusing** existing terminology. SPKM systems can be connected to each other forming a network of knowledge; this network allows **sharing** knowledge with others. The interface of the SPKM system can be **personalised** to each user and **adapt** to his preferences.

4 Architecture

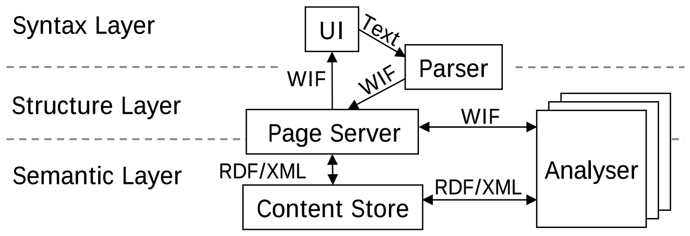

Fig. 6. Overview of the System Architecture

Our proposed system architecture is depicted in Fig. 6. The architecture adheres to the REST architectural style [4, Ch. 5] for reasons of extensibility, entry barrier level, distribution, and scalability [4, Ch. 4].

We identify five main components. The components are connected through HTTP and exchange data in plain text, Wiki Interchange Format (WIF), and RDF. The user interface displays pages for viewing; it requests pages from the page server, which in turn retrieves them from the content store. When the user edits or creates wiki pages (using a text-based wiki syntax), the parser converts the pages into WIF, and the page server stores the page using the content store (which takes care of versioning etc.).

We now explain the components, the data formats, and related issues, and present concrete implementations of the architecture.

4.1 Components

User interface displays pages to the users. The user interface receives one or more pages (e.g. the page content, a navigational menu, a set of related items, and a calendar) from the page server, combines them, and renders them on the screen.

Some important considerations for the user interface are:

– The user interface should enable navigation to *related items* (information related to the current view): given that people use navigation and orienteering rather than direct search [23], we must allow users to passively discover related information. We discuss several methods to discover related items in Sec. 4.5.
– The user interface should be *adaptive* to the user. Adaptivity can be explicit (users state their preferences, such as hiding some sections of the page and amplifying others) or implicit (tracking all user actions to detect user preferences). The user interface could also show a session navigation history, and other personalised navigation.
– The user interface should support *structural editing*: splitting and joining pages into more granular entities, changing structural levels of content.

Parser transforms wiki syntax (that may differ in each wiki) into the neutral WIF that captures the structure of the wiki text. The parser interprets knowledge on the syntax layer and transforms it into the structure layer.

Each deployment of an SPKM system can choose a different wiki syntax by plugging in a different parser component. We introduce the notion of an *open parser*, a (publicly) accessible REST service that receives wiki text and returns WIF. We have implemented several parsers that transform a specific augmented wiki syntax into WIF. With a number of open parsers readily available, personalising the wiki amounts to simply specifying the location of the preferred parser.

In an SPKM context users will often use ad-hoc, personal, syntactical conventions, e.g. "`@@`" to indicate a todo item (c. f. Sec. 2.2). These ad-hoc conventions can be parsed with an adaptive parser (learning from examples [5]). Such an adaptive parser would be architecturally identical to ordinary parsers.

Page Server mediates between the content store (RDF store) and the other components. It can store and retrieve wiki pages (in WIF syntax). It encapsulates the business logic of a wiki, and hides the RDF internals of the content store.

Content Store is the persistence component. The content store should store both RDF data and binary data, and support versioning and access control. The content store[4] could reuse existing RDF stores, binary databases, RDF versioning techniques [25], and RDF access control [22]. All functionality should be uniformly accessible over HTTP, as shown in Fig. 7.

Analysers are non-standard components that analyse and integrate data on either the structural layer or the semantic layer. Users are often unaware of important information related to the item they are just dealing with. Related items are either items that are similar to the current item, or belong to the same category as the current item.

[4] see `http://www.stoRDF.org` for an effort to create such a content store.

Possible ways to compute similarity or classification are vector space models (using cosine similarity), reasoning, data mining, clustering (e.g. on creation date), link topology analysis, or statistical analysis (e.g. Bayesian classification). All these methods bring the fuzziness of the real world back into the hard-edged RDF world.

Interoperability with other systems is a crucial element of the architecture. Other systems can interact directly on the content store (exposing their data as RDF) or via the PageServer (exposing their data as wiki pages). For non-semantic applications, direct interoperability on the content store requires "lifting" onto the semantic level, for which Semantic Desktop adaptor frameworks [20] can be used.

4.2 Connectors

The single connector in the architecture is HTTP[5]. Optimisations can be employed if the components run in the same execution environment (which is reasonable in a personal desktop system).

4.3 Data Formats

Three types of data are exchanged: plain text in wiki syntax, wiki pages in WIF syntax, and RDF data in RDF/XML. This section describes these data formats.

Pages are written in **plain text** (content type: `text/plain`) by the user and transformed into WIF by the parser; the syntax of the wiki text depends on the parser.

RDF is used in communication with the content store, by the page server (that translates this into WIF for the other components) and by some analyser (that need direct access to the data for querying). RDF/XML (content type: `application/rdf+xml`) is a standardised serialisation format for RDF.

WIF describes wiki pages but abstracts from specific wiki syntaxes; it can therefore be used as interchange format between wiki engines. WIF describes both the structural level and the semantical level of pages, but allows ordinary (non-semantic) wikis to ignore the semantic level.

For the **structural data** we use XHTML, a standardised structural document format. Ordinary wikis already export HTML and could therefore straightforwardly export XHTML. The only wiki data that is not natively encoded in XHTML is the nature of hyperlinks: in XHTML all links are equal, but wikis distinguish internal and external links. These link types can however (in a microformats[6] inspired approach) be encoded in the `class` attribute of the `link` element.

[5] `http://www.w3.org/Protocols/rfc2616/rfc2616.html`.
[6] `http://www.microformats.org`.

For the **semantical data** we use RDF/A[7], a proposal for embedding RDF data in XHTML documents. The resulting XHTML documents can also be processed by ordinary (non-semantic) wikis by ignoring the RDF statements.

WIF is transported as `application/xhtml+xml`.

4.4 SPKM Core Ontology

For storage, querying and exchange on the RDF layer, we define a core SPKM ontology, as depicted in Fig. 8. This ontology[8] is kept simple while capturing all relevant aspects of wiki pages: title, content, hyperlinks, authors, and versions. The ontology is linked to WikiOnt [6] and SWIFT[9], and can be extended by applications to include e.g. more fine-grained descriptions of the content.

To distinguish between the wiki page itself, and the concept that it is talking about, we need two different URIRefs (because we want to make statements about both). We postfix the URL of the page with a "#" to denote the concept. The RDF data has to state the relationship between the page URL and the concept URIRef explicitly, using e.g. `rdfs:seeAlso`. Note that the concept URIRef is syntactically not a valid URL. It is not a location, it cannot be dereferenced and no document can be placed there; it only denotes a concept[10].

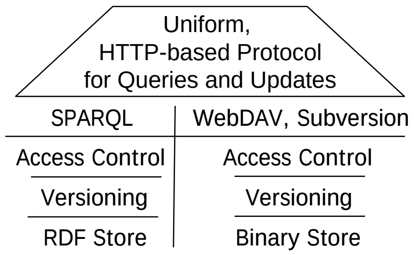

Fig. 7. A Uniform Storage Component

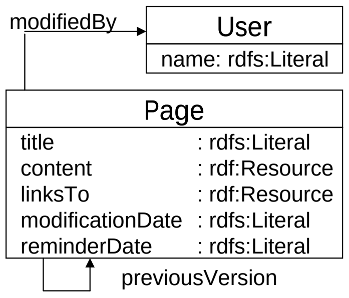

Fig. 8. SPKM Core Ontology

4.5 Faceted Triple Browsing

The user interface displays related items (related to the current page). One approach for viewing and navigating related items is faceted browsing [28], in which the information space is partitioned in conceptual dimensions that constrain the currently visible elements of the information space. For example, a collection of

[7] `http://www.w3.org/2001/sw/BestPractices/HTML/`.
[8] `http://www.xam.de/2005/12/spkmcore`
[9] `http://www.xam.de/2005/08/swift`.
[10] `http://xamde.blogspot.com/2005/11/uri-crisis-solved.html`.

art works can consist of facets such as type of work, time periods, artist names, geographical locations, etc. To implement faceted browsing one normally needs a schema that defines the facets. But in browsing arbitrary RDF we cannot rely on having a schema definition of the data.

We therefore introduce *faceted triple browsing*, a technique for partitioning, displaying, and navigating arbitrary schemaless RDF data. In the wiki we show the current page, and show the list of related pages in a sidebar. If that list is too long, facets and their result size are displayed instead, expanding on demand. This works recursively for subsets by chaining selected facets as a conjunctive query. We calculate a list of facets in two steps: first we find sets of related statements, take their union, and then render this set with facets.

1. the **related statements** for a given resource r are the following three sets: incoming links $r_{in} = (*, *, r)$, property-links $r_p = (*, r, *)$, and outgoing links $r_{out} = (r, *, *)$. The union S of these three sets contains every statement that mentions r.
2. the **facets for the set** S are calculated by grouping similar statements, as follows
   (a) for every statement $a = (s, p, o) \in S$,
      i. we perform six queries, leading to six sets related to a: $(s, p, *)$, $(s, *, o)$, $(*, p, o)$, $(s, *, *)$, $(*, p, *)$, $(*, *, o)$.[11]
      ii. each query with more than one result collects all statements in S that are related to each other (and to a), they thus form a facet, based on the query pattern. Therefore for each query with more than one result we create a facet consisting of the query results, label it with the non-wildcard parts of the query, and add it to the set F of facets for S.
   (b) we remove all duplicate facets from F (duplicates occur since step 2a is repeated for each statement in S).

4.6 Implementations

We currently have to independent implementations of our architecture, SemperWiki [13] and SemWiki [24].

SemperWiki is a desktop application, implemented in Ruby using the Gtk windowing toolkit. SemperWiki is designed as a personal wiki that focuses on usability and desktop integration. Since all components are hosted on one machine the components communicate using a native protocol. SemperWiki offers faceted browsing of related data, supports arbitrary queries and database-like views, and integrates existing web ontologies into the personal wiki.

SemWiki is a browser-based application, that can be hosted locally (as a personal wiki) or on a server (as a public wiki). The components communicate using HTTP and can be fully distributed. Pages are built out of resources, snippets

[11] The statement a itself and the pattern $(*, *, *)$ do not lead to interesting results.

of wiki text, or queries. Rendering is done by applying XSLT stylesheet transformations to the structural data (WIF), and resulting XHTML is displayed by a standard web browser. SemWiki allows for simple browsing of arbitrary RDF and has an intuitive query mechanism.

5 Evaluation

It is not easy to evaluate an architecture; we could evaluate specific tools for personal knowledge management with our implementations, but that would not give the complete picture.

We therefore compare our architecture with other frameworks, namely the semantic desktop Gnowsis [19] and Haystack [17], classical PIM tools such as Microsoft Outlook, and wiki such as MediaWiki and TWiki. We compare these frameworks qualitatively based on our requirements in Sec. 2.3, the results are shown in Fig. 9. Due to space considerations we can not compare them in more detail.

	Syntactical	Structural	Semantic	Integrated	Query	Associative browsing	Clustering	Notification	Composition	Rule execution	Terminology re-use	Communication	Privacy	Interoperability	Context management	Adaptive interfaces	Authoring freedom
	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.	16.	17.
PIM tools	++	+	--	--	++	--	--	+	--	+	--	+	++	+	+	+	--
Semantic Desktop	1)				++	++	+	+	-	--	--	+	+	++	+	+	
Classical Wiki	++	++	--	+	-	++	--	--	+	--	+	-	+	--	--	+	++
SPKM	++	++	++	++	++	++	++	+	++	+	+	+	+	++	+	++	++

Fig. 9. Comparing SPKM to Related Approaches

PIM tools suffer from their fixed schemas for data types (req. 15) and the fact that one cannot compose or browse items (req. 10, 6). They have good communication with PDAs and exchange servers (req. 12-14), but limited communication with other applications.

Semantic Desktop systems interlink data objects across applications (req. 12, 14), and allow users to browse desktop items in an associative, cognitively adequate way (req. 6). However, these systems do not offer authoring of semi-structured data.

Classical wiki systems do exactly that: they allow for simultaneous authoring of free text and semi-structured data (req. 1, 2, 4). Wiki systems have limited

[1] Since the authoring capabilities of a Semantic Desktop depend on the integrated applications, we do not consider it a property of the Semantic Desktop system itself.

reuse of existing knowledge outside the wiki, and not offer reminding functionality, and do not provide interoperability with existing applications.

An SPKM tool can be seen as the missing piece in Semantic Desktop systems: it fills the authoring gap and offers universal access and simultaneous authoring of personal, structured, unstructured and semantic knowledge.

6 Conclusion

In this paper we have presented our SPKM architecture and implementations for semantic personal knowledge management. The requirements for personal knowledge management that we have derived are: finding and reminding, collaboration, knowledge re-use and cognitively adequate interfaces. These requirements show the limitations of current solutions.

Our solution is based on enhanced semantic wikis, for their authoring flexibility and ease-of-use. Our solution addresses these requirements and gives the user a uniform way to work on all knowledge layers. We have presented our open and scalable architecture that allows the integration of existing wikis and other applications, and have introduced our implementations.

Acknowledgements This research was partially supported by the European Commission under contract FP6-507482 (KnowledgeWeb), and is based upon works supported by the Science Foundation Ireland under Grants No. SFI/02/CE1/I131 and SFI/04/BR/CS0694. The expressed content is solely the view of the authors.

References

1. S. Decker, J. Park, D. Quan, and L. Sauermann, (eds.). *The Semantic Desktop – Next Generation Information Management & Collaboration Infrastructure.* Galway, Ireland, 2005. Cited in 13 and 18.
2. M. W. Eysenck and M. T. Keane. *Cognitive Psychology: A Student's Handbook.* Psychology Press (UK), August 2000. Cited in 2.3.
3. D. Fensel *et al.* Integrating semiformal and formal methods in knowledge-based systems development. In *Proc. of the Japanese Knowledge Acquisition Workshop (JKAW-94)*, pp. 73–89. Hitachi, Japan, 1994. Cited in 3.
4. R. T. Fielding. *Architectural styles and the design of network-based software architectures.* Ph.D. thesis, University of California, Irvine, 2000. Cited in 4.
5. S. Gerke. *Interaktives Erkennen von Textstrukturen durch Maschinelles Lernen.* Studienarbeit, Institute AIFB, University of Karlsruhe, 2005. Cited in 4.1.
6. A. Harth, *et al.* Wikiont: An ontology for describing and exchanging wikipedia articles. In *Proc. of Wikimania 2005 – The First Int. Wikimedia Conf.* Jul. 2005. Cited in 4.4.
7. 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. Cited in 2.3, 2.3, and 2.3.
8. M. Lansdale. The psychology of personal information management. *Applied Ergonomics*, 19(1):55–66, 1988. Cited in 2.3 and 2.4.

9. B. Leuf and W. Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet.* Addison-Wesley, 2001. Cited in 3.
10. I. Nonaka. A dynamic theory of organizational knowledge creation. *Organization Science*, 5(1):14–37, Feb. 1994. Cited in 2.1 and 2.3.
11. I. Nonaka and H. Takeuchi. *The Knowledge-Creating Company.* Oxford University Press, New York, 1995. Cited in 1 and 2.1.
12. H. Obendorf. The indirect authoring paradigm - bringing hypertext into the web. *J. Digit. Inf.*, 5(1), 2004. Cited in 3.
13. E. Oren. SemperWiki: a semantic personal wiki. In Decker *et al.* [1]. Cited in 4.6.
14. A. Pivk, P. Cimiano, and Y. Sure. From tables to frames. *Journal of Web Semantics*, 3(2):132–146, Oct. 2005. Cited in 2.2.
15. M. Polanyi. *Personal Knowledge: Towards a Post-Critical Philosophy.* Routledge & Kegan Paul Ltd, London, 1958. Cited in 2.1.
16. M. Polanyi. *Tacit Dimension.* Routledge & Kegan Paul Ltd, London, 1966. Cited in 2.1.
17. D. Quan, D. Huynh, and D. R. Karger. Haystack: A platform for authoring end user semantic web applications. In *Int. Semantic Web Conf.*, pp. 738–753. 2003. Cited in 5.
18. J. Rohmer. Lessons for the future of Semantic Desktops learnt from 10 years of experience with the IDEALIANCE semantic networks manager. In Decker *et al.* [1]. Cited in 2.3.
19. L. Sauermann. The gnowsis semantic desktop for information integration. In *Wissensmanagement*, pp. 39–42. 2005. Cited in 5.
20. L. Sauermann and S. Schwarz. Gnowsis adapter framework: Treating structured data sources as virtual RDF graphs. In *Int. Semantic Web Conf.*, pp. 1016–1028. 2005. Cited in 4.1.
21. P. M. Senge. *The Fifth Discipline.* Currency, January 1994. Cited in 2.1.
22. M. Sibler. *Semantische Zugriffskontrolle - Rechtemanagement mit Ontologien und Regeln.* Master's thesis, AIFB Karlsruhe, Jul. 2005. Cited in 4.1.
23. J. Teevan, C. Alvarado, M. S. Ackerman, and D. R. Karger. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *CHI '04: Proc. of the SIGCHI conf. on Human factors in computing systems*, pp. 415–422. ACM Press, 2004. Cited in 2.3 and 4.1.
24. M. Völkel. SemWiki – a RESTful distributed wiki architecture. Proc. of the WikiSym 2005 (demo session), 2005. Cited in 4.6.
25. M. Völkel, *et al.* SemVersion - versioning RDF and ontologies. KnowledgeWeb Deliverable D2.3.3.v1, Institute AIFB, University of Karlsruhe, Jun. 2005. Cited in 4.1.
26. E. Wenger, R. Mcdermott, and W. M. Snyder. *Cultivating Communities of Practice.* Harvard Business School Press, March 2002. Cited in 1.
27. 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. Cited in 2.3, 2.3, and 2.4.
28. 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. Cited in 4.5.