Kaukolu:
Hub of the Semantic Corporate Intranet

Malte Kiesel

DFKI GmbH, Kaiserslautern, Germany,
malte.kiesel@dfki.de

Abstract. Due to their low entry barrier, easy deployment, and simple yet powerful features, wikis have gained popularity for agile knowledge management in communities of almost all sizes. Semantic wikis strive to give entered information more structure in order to allow automatic processing of the wiki's contents. This facilitates enhanced navigation and search in the wiki itself as well as simple reuse of information in external applications or for generating different views on the same information. This makes semantic wikis especially interesting for corporate intranet deployment, implementing the *Semantic Intranet*. In this paper, we will have a look at Kaukolu, an open source semantic wiki prototype, being deployed in a corporate intranet. External applications use information authored in Kaukolu, effectively forming a cluster of applications interacting and sharing data.

1 Introduction

Wikis become more and more important for managing content of corporate intranets, serving as a platform for information exchange and as knowledge repositories. Typically, part of the information found in corporate intranets are simply plain text (e.g., texts describing projects, templates for mails, brainstorming, tips and best practices, ...), but a major part of the content consists of structured data such as lists relating people to projects, product feature lists, publication lists, or simply collections of annotated web links. While simply being able to manage all of this different content by dropping it—as text—into the wiki is handy, a direct consequence of this is that everything in the wiki *is* essentially text and therefore cannot be imported into other applications such as spreadsheet applications, databases, or a content management system used for the external web site. So, a lot of data duplication needs to be done, ultimativaly resulting in unnecessary workload, outdated content, and inconsistencies in the data presented at different places for differing audiences.

Semantic wikis try to implement a way to establish and maintain structure of the wiki's content using semantic web technologies in order to facilitate information reuse or, in general, to facilitate accessibility of the information to automated means. Also, knowledge of the inner structure of the information contained in wiki pages can be used to enhance browsing and search in the wiki.

In section 2, we give a short overview over the basic wiki ideas and explain the shortcomings of wikis concerning structured data. In section 3, an overview over (semantic) wiki implementations is given, some semantic wiki features are explained, and several problems with existing semantic wikis are mentioned. *Kaukolu*, our implementation of a semantic wiki, is introduced in section 4, along with a walkthrough in section 5. In section 6, a number of possible enhancements of Kaukolu are presented. We end with a conclusion in section 7.

2 What is a Wiki?

Wikis allow a group of people to collaboratively author information using a tool that is easy to use. The main features a wiki provides are kept simple, but flexible, in order to allow for using the basic features for a number of different purposes. For example, the very basic wiki idea of editing a text page allows both editing a document and doing a discussion. Backlinks, another standard wiki feature, can be used for navigation, tagging, or grouping sets of pages. Basic content format of wikis is text so import/export functionality is limited to text formats.

Current applications of wikis range from open encyclopedias such as Wikipedia to collaborative information spaces for both open communities such as open source software projects (e.g., http://wiki.mozilla.org/—even software project management software such as Trac[1] feature wikis for documentation and information exchange) and closed communities such as company intranets.

Structured Data Falls Through the Cracks

A major drawback of wikis is that they are intended only to edit and display plain text[2]—content that represents structured data (e.g., tables or sets of wiki pages using the same structure) can only be exported as text or HTML. These formats preserve content and looks to some degree, but the information structure (i.e., explicit knowledge of what values populate what properties of what entities) gets lost[3]. This is unfortunate: People who maintain, for example, their publication list in the wiki, have to manually re-enter the same information in other places such as the company extranet. However, duplication of information is tedious work, often leading to inconsistencies and large amounts of outdated information. Also, lack of data structure prevents us from running queries or compiling statistics against the data.

Importing information into a standard wiki suffers from the inability to process structured data, too. For example, while it is possible to import spreadsheet data by attaching the spreadsheet file to a wiki page, this data cannot be accessed

[1] http://www.edgewall.com/trac/

[2] Text can get formatted, but this is for the looks only.

[3] Some wikis support structured data to some degree using templates or similar features. However, typically these are proprietary approaches that provide no interoperability with other applications.

or edited in the wiki. The spreadsheet's data structure cannot be exploited and reused. Another way to import spreadsheet data would be to export the data as HTML in the spreadsheet application and import the HTML into the wiki. However, importing large amounts of HTML prevents the users from contributing to the content. Even if users dare to edit the HTML code, keeping the original spreadsheet document in sync with the changes can only be done manually.

This means that existing structures effectively cannot be maintained inside the wiki. Since structured data gets "flattened" on import, users may end up with an unstructured data repository that is difficult to manage and difficult to keep in sync with the corresponding information outside of the wiki.

How to solve this issue?– A naive solution for importing spreadsheet data, for example, would be to import *comma–separated values*[4] (CSV) into the wiki. This would at least bring one benefit of wikis, namely collaborative editing, together with structured data. However, readability and flexibility of CSV is very low. Anybody trying to introduce a new property of an item (anybody who would try to add a new column) would have to reformat all data that has been added so far. Also, anybody depending on the old CSV structure would have to get notified.

The Semantic Wiki Idea

Semantic wikis try to overcome the problem stated above by combining semantic web standards such as RDF/S or OWL with the wiki paradigm. One idea is to *annotate* structure in the wiki by providing *metadata* for existing features such as links and pages. On the other hand, one can strive to completely *represent* the wiki content using instances of the respective ontology language.

3 An Overview over Several Wikis

In [9], an overview of semantic wikis and personal wikis is given, resulting in the description of *SemperWiki*, a semantic desktop wiki.

In most traditional wikis, the idea of metadata typically only appears in a very technical way. For example, in *JSPWiki*[5], metadata is added directly into the wiki text using special tags, and mostly serves the purpose of implementing access control. In *SnipSnap*[6], labels may get attached to wiki pages, serving mainly as a categorization scheme.

The semantic wiki *Platypus*[7] adds RDF(S) and OWL metadata to wiki pages. Metadata has to be entered separately from wiki text and relates a wiki page to another resource; thus, metadata can be transformed into a list of *related pages* that can be shown along with the actual wiki page.

[4] http://en.wikipedia.org/wiki/Comma-separated_values
[5] http://www.jspwiki.org/
[6] http://snipsnap.org/
[7] http://platypuswiki.sourceforge.net/

The *Semantic MediaWiki*[8] [7] is an extension of *MediaWiki*[9], the software used by Wikipedia. Again, metadata associated to a wiki page may point to other resources, but here, literals are allowed, too. Also, metadata is entered directly into the wiki text, and does not have to adhere to a schema. A nice feature of this implementation is its support for multiple datatypes such as coordinates and temperatures, along with conversion between different unit scales.

Rhizome[10] [14] builds on a framework that adapts techniques such as XSLT and XUpdate to RDF. In essence, RDF is used throughout the framework for almost everything, and RxSLT (an XSLT variant adapted for RDF) is used for transforming queries' results to HTML or other output formats. Page metadata has to be entered separately from the page. While the approach is very interesting from a technical point of view, the current implementation requires a lot practice with the underlying techniques.

IkeWiki[11] [13] is a rather new wiki supporting OWL ontologies. It supports inferencing when typing links and relies on JavaScript–based features for supporting the user which helps quite a lot when adding semantic information.

OpenRecord[12] is a kind of database/spreadsheet wiki. It focuses on enabling the user to enter structured data using tables. It heavily uses JavaScript, providing almost the feeling of a standalone application. However, currently it is in alpha stage only.

Problems Found in Existing Semantic Wikis

Existing (semantic) wikis lack in some areas:

Interoperability: While one of the main points of semantic web standards is interoperability, there seems to be no semantic wiki that allows *import* of RDF data. Some wikis allow usage of ontologies (in OWL or RDFS language), but integration into the wiki concepts seems to be amendable. For example, ontologies loaded typically do not show up in the wiki since they are loaded into a separate repository. Thus, ontologies are deemed to remain *static* and cannot be edited by users of the wiki.

Annotation complexity: In existing semantic wikis, RDF is mainly used for *annotations*: RDF supplies semantic information that describes existing human–readable features. Since the basic blocks of wikis are pages and links between them, mapping a wiki to RDF can be done by using pages as representatives of RDF resources, with links between wiki pages denoting relations between RDF resources. This approach is typically implemented by enabling the user to attach RDF triples to wiki pages[13], but setting the subject of each triple of the page

[8] http://semediawiki.sourceforge.net/
[9] http://mediawiki.sourceforge.net/
[10] http://rx4rdf.liminalzone.org/Rhizome
[11] http://ikewiki.salzburgresearch.at/
[12] http://openrecord.org/
[13] Often in terms of selecting types for links to other pages

to the page's URI[14]. It follows that when using RDFS, wiki pages must be both of the type *wiki:page* and of the type the *resource* the wiki page is supposed to describe. This has two drawbacks: First, from a knowledge engineer's point of view, existence of an entity that is both a text (a wiki page) and, for example, a person, is not desirable. Second, while the approach can be handy for generating RDF data of "shallow" ontologies with few classes and many relations[15], we think that it reaches its limits as soon as more elaborate ontologies and structures are used. For example, in Figure 1 the RDF structure of a *foaf:person* is depicted. In semantic wikis that identify an RDF resource with a wiki page, one wiki page of the type *foaf:person* can be used to model this data. However, if we try to model the data shown in Figure 2 (a bibtex entry represented in a format similar to the format used by the *bibtex2rdf*[16] converter), we would need four wiki pages (one wiki page per RDF resource) for just one bibtex entry.

There are other cases that make the problem even more obvious. For example, imagine a large table that lists 100 products along with a short description and price. In order to express this in a semantic wiki that identifies a page with a resource, one gets forced to create 100 wiki pages, one for each row of the table, both cluttering title index and recent changes pages.

In general, we believe that imposing a structure on wiki contents due to technical reasons is against the wiki way. Users should be free to use whatever page structure they want. Structured data, as is RDF, is only *another view* on the wiki content.

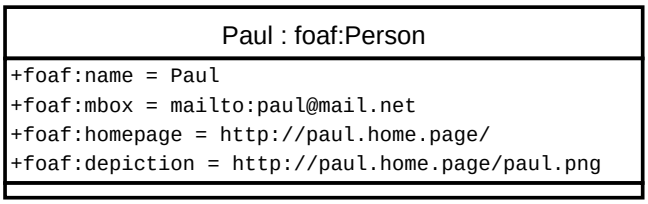

Fig. 1. A foaf:person.

Smooth migration: While some existing semantic wikis allow addition of semantic features to existing content (for example, by typing previously untyped links in the wiki), no wiki seems to provide features to assist the user when extracting further semantic features from (imported) plain text.

[14] The point is that in this approach triples are bound to pages because of their subject URI. It does not really matter whether this URI is the URL the wiki page can be browsed at, a separate "wiki page concept URI", or an arbitrary URI.

[15] For some applications such as Gnowsis [12], this approach is followed by our wiki implementation, too [6].

[16] http://www.l3s.de/~siberski/bibtex2rdf/

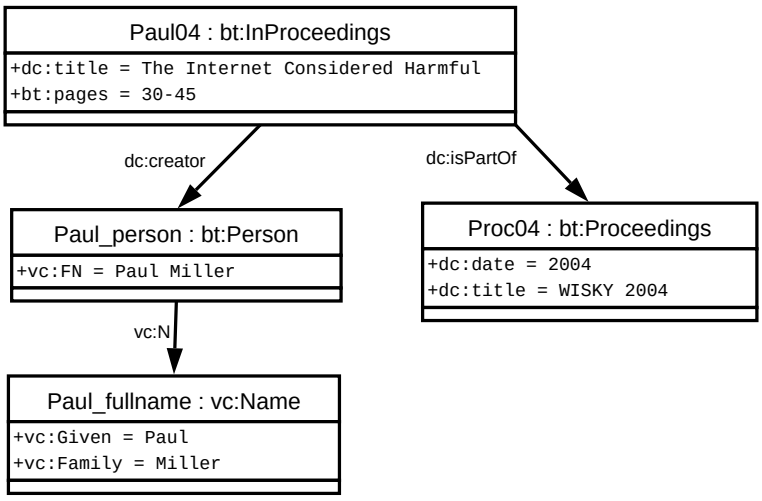

Fig. 2. A bibtex RDF entry.

Queries: The only means of querying semantic information is either very simple queries built with a user interface (such as "Show a list of all publications to me") or complex queries entered manually in a query language such as SPARQL [10].

4 Why Kaukolu is Different

Kaukolu[17], our implementation of a semantic wiki, builds on JSPWiki[18] and Sesame 2[19]. It differs in several aspects from the existing semantic wikis.

- no restrictions are imposed on RDF triples attached to a page (triple's subjects are *not* fixed)
- arbitrary RDF(S) files can be imported
- aliases can be defined for resources and predicates
- autocompletion supports the user when formalizing content

We currently use Kaukolu internally in our department. An evaluation in another company will take place this year.

No Restrictions on RDF Triples

Kaukolu allows to formulate arbitrary RDF on any wiki page using a slightly extended wiki syntax. Subjects of RDF triples are not required to represent the URI of the page the triple is located at. This solves the issues explained in section 3 and allows for more complex RDF data.

[17] Available at http://kaukoluwiki.opendfki.de/ including sources
[18] http://www.jspwiki.org/
[19] http://www.openrdf.org/

RDF(S) Import and Export

Being able to associate arbitrary RDF with a wiki page not only works when formulating RDF but also allows to import RDF. In fact, since RDF Schema is also represented in RDF, one can even import RDFS ontologies to Kaukolu using this method. Imported RDFS ontologies can be used in various ways within Kaukolu, we will explain this later. A direct benefit of RDFS ontologies being stored on wiki pages is that this way users are able to edit and extend the ontologies used by the wiki in a straightforward way, using all features a wiki provides (versioning, collaborative authoring, viewing diffs, ...). However, one has to say that currently changing RDFS using this approach is quite difficult as one has to directly work on RDFS without any tool support.

Aliases Replacing namespace:localname URIs

In contrast to most existing semantic wikis, users of Kaukolu are not required to use localnames, labels, or namespaces of RDFS properties in order to express RDF triples using these predicates. For example, typically the user has to write something like *(this) dc:author "Author Name"* if he wants to express that the current wiki page has a Dublin Core *author* property. In Kaukolu, we allow an intermediate step: every RDF instance or RDFS property may be associated to arbitrary strings (*aliases*) that can be used instead of the URI/label of the respective property or instance. In Figure 3, aliases are defined using the *hasSubjectURI/hasPredicateURI* keyword. This not only relieves the user from having to remember namespaces or localnames but also facilitates internationalization by usage of ontology metainformation [3].

Autocompletion for Both Semantic and Non–Semantic Content

Of course, even with wiki syntax and aliases for properties and instances, entering RDF triples is a tedious task. Without further support, the user would need to keep the documentation of the ontologies always at hand, typing mistakes would introduce severe errors, and the user would have to remember the URIs of all RDF instances created in the wiki. In Kaukolu, there is ontology–based autocompletion support, which proposes aliases based on RDFS range and domains. For example, when typing *Paul knows*, with *Paul* being a *foaf:person*, and *knows* being associated to *foaf:knows*, the system automatically proposes a list of *foaf:persons* defined in the wiki to complete the RDF triple, as only *foaf:persons* are allowed as range of *foaf:knows*, even without any prefix typed. If a prefix has been typed already, it is used to narrow down the list of suggestions. Autocompletion works for predicates, too. In case no alias is found in the typed text, Kaukolu assumes that the user does not intend to write triples, and simply proposes names of wiki pages as autocompletion suggestions, based on the prefix already typed. So if you type "InfoOn", and there are "InfoOnPaul" and "InfoOnSarah" pages in the wiki, those both page names are suggested.

All Standard Wiki Features are Implemented

Most other semantic wikis have been rewritten from scratch and therefore miss several standard wiki features such as file attachments, access control, plugin support, or support for multiple backends. Kaukolu is based on JSPWiki[20], an established wiki that is quite feature–complete.

5 Kaukolu in Practice

In the following, we will demonstrate some of Kaukolu's features. Paul Miller will import an ontology describing bibtex entries into Kaukolu, and add a new publication item to a wiki page (ontology–driven autocompletion will help here). Then, Paul will export the bibtex RDF generated into an external application *RDFHomepage* [5] which generates an HTML page containing a publication list.

Ontology Import: In Figure 3, we see a wiki page holding the bibtex RDFS ontology used for our publication list. Any RDFS ontologies can get imported. On import, they will be converted to RDF wiki syntax which is similar to N3 [1]. Ontologies can be created using ontology editors such as Protégé-2000 [8]. Note that one can now collaboratively edit the ontology within Kaukolu. Export to RDFS is also possible using the "View related RDF" button to the lower right. Updating the ontology can be done either directly in the wiki or by re–importing the ontology.

Fig. 3. The wiki page holding the bibtex RDFS ontology.

Add a Publication Entry: In Figure 4, we see a user adding a publication entry to his wiki page. Since *hasType* (corresponding to *rdf:type*) implies that

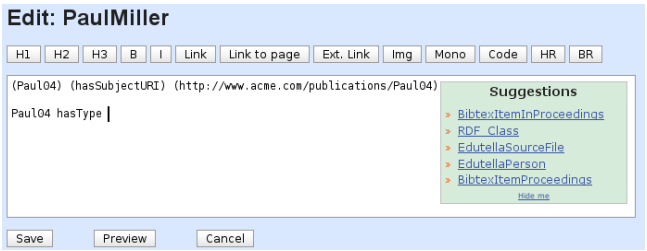

Fig. 4. Ontology–based autocompletion in action.

the triple's object will be of type *rdfs:Class*, only instances of *rdfs:Class* are displayed in the autocompletion suggestion box.

The complete page describing Paul's publication item is shown in Figure 5. Note that one can use standard wiki markup along with the RDF extensions (a bullet list is used).

Fig. 5. The complete bibtex item.

Export Publication List to External Application: In Figure 6, we see an HTML page generated by *RDFHomepage* using the RDF formulated on the user's wiki page. Note that the page generated is intended for external audiences and cannot be edited. The RDF created in Kaukolu, a Java–based application, is processed by RDFHomepage, a PHP–based application.

[20] http://www.jspwiki.org/

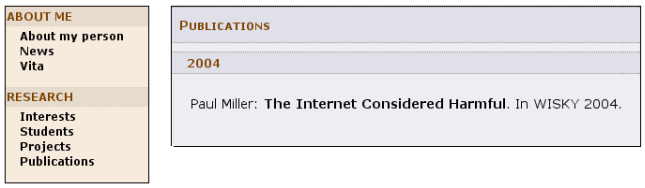

Fig. 6. The publications page as generated by RDFHomepage.

Further Applications

Kaukolu can be used for different ontologies, too. For example, RDFHomepage not only generates a publication list, but also uses information contained in an *organizational repository* (formulated in RDF/S) for generating a list of projects the respective person using RDFHomepage participates in. Kaukolu allows to collaboratively maintain the organizational repository which up to now has been maintained centrally in our department.

6 Future Work

Limitations of Kaukolu

Kaukolu currently does not provide means to generate a number of RDF constructs. Most notably, it cannot deal with RDF containers such as bags, sequences, and alternative values yet. While this is a drawback when authoring or importing RDF, expressivity is not touched by leaving these features away since these constructs can be substituted by RDF lists and multiple–valued properties in most cases. Alternatively, Kaukolu's wiki syntax for creating RDF triples could be extended to incorporate handling of containers.

Another limitation of Kaukolu is its inability of generating blank nodes. On import, any blank nodes may get assigned a random and unique URI. However, Kaukolu currently never exports blank nodes.

Planned Features

We have identified several ways in which Kaukolu may be improved.

Use of RDF metadata within Kaukolu: While currently the main reason for generating RDF is its usage in external applications, Kaukolu can use RDF data for navigation: If RDF data is attached to a page, it can be shown in a navigation sidebar. We are aware that this is only a very basic feature. Further possibilities for using annotations within the wiki would be to use them for search or feature a reverse translation of RDF to wiki markup or HTML directly for display which would allow to create customized views on formalized wiki

content ("Show a list of all persons working in project X here" or "Show a list of all properties of software X here"). Previous work on this topic includes Fresnel [2] and Haystack [11].

Proposal of RDF metadata from natural texts: We plan to use the SProUT natural language processing module [4] in Kaukolu. This will allow to generate RDF from natural text and partly eliminate the dependence on formulating triples directly. Also, this will be a great feature for switching from a standard wiki to Kaukolu since existing texts can get mined for RDF data then automatically. While we do not expect the extracted data to be perfect, we believe that it will serve as a start. SProUT has been used in the SmartWeb project[21] for extracting RDF instances from natural texts in the sports domain.

Creation of RDFS instances: Creating instances of RDFS classes by entering the corresponding RDF triples is quite time–consuming. There should be a more comfortable way to create new instances. A lightweight way would be to create default triples according to the ontology and to let the user fill in object values.

Better embedding of RDF triples: Currently, any RDF expressed in Kaukolu must be part of a wiki page's text and therefore gets displayed when the page gets rendered. Since Kaukolu allows aliases for subjects, predicates, and properties to get embedded in natural text (as in "*PaulMiller* came to *know SarahMiller* in 2005"), there is no absolute need of separating the parts of the text that represent RDF from the remaining text. However, in practice embedding triples often leads to awkward sentences. A more flexible way of embedding RDF–generating content into wiki pages seems desirable. In the future, we will implement a feature that allows to separate RDF–generating statements from normal text. Features to generate these statements as well as features to keep them in sync with normal text will be added.

7 Conclusion

In this paper, we gave an overview over the ideas of (semantic) wikis and implementations that are available. Our semantic wiki prototype Kaukolu addresses some of the shortcomings of existing semantic wikis and is intended for intranet usage. Its main features are its ability to import and export RDF and its ontology–supported autocompletion feature which relieves users from having to know the ontologies used letter by letter. We believe that export *and* import of structured data are essential features for a semantic web application. A demonstration of theses features was given, using a publication list as an example for structured content which gets formulated in Kaukolu and used in a separate application, illustrating Kaukolu's export functionality. Finally, we discuss some issues in Kaukolu, along with ideas how to address them.

[21] http://www.smartweb-project.de/

8 Acknowledgments

This work has been supported in part by the NEPOMUK project, which is funded by the IST Programme of the European Union under grant FP6-027705.

References

1. Berners-Lee, T. Getting into RDF & Semantic Web using N3, 2000.
2. Bizer, C., Lee, R., and Pietriga, E. Fresnel - Display Vocabulary for RDF, 2005.
3. Buitelaar, P., Sintek, M., and Kiesel, M. Integrated Representation of Domain Knowledge and Multilingual, Multimedia Content Features for Cross-Lingual, Cross-Media Semantic Web Applications. In *Proceedings of the ISWC 2005 Workshop on Knowledge Markup and Semantic Annotation* (2005).
4. Drozdzynski, W., Krieger, H.-U., Piskorski, J., Schäfer, U., and Xu, F. Shallow processing with unification and typed feature structures — foundations and applications. *Künstliche Intelligenz 1* (2004), 17–23.
5. Grimnes, G., Schwarz, S., and Sauermann, L. RDFHomepage or Finally, a Use For Your FOAF File. In *Proceedings of Semantic Web Scripting Workshop at ESWC06* (2006). http://rdfhomepage.opendfki.de/.
6. Kiesel, M., and Sauermann, L. Towards Semantic Desktop Wikis. *UPGRADE special issue on "The Semantic Web"* (2005).
7. Krötzsch, M., Vrandecic, D., and Völkel, M. Wikipedia and the Semantic Web — The Missing Links. In *Proceedings of Wikimania 2005* (JUL 2005), Wikimedia Foundation. http://www.aifb.uni-karlsruhe.de/WBS/mak/pub/wikimania.pdf.
8. Noy, N. F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R. W., and Musen, M. A. Creating Semantic Web contents with protege-2000. *IEEE Intelligent Systems 16*, 2 (2001), 60–71.
9. Oren, E. SemperWiki: A Semantic Personal Wiki. In *Proceedings of the 1st Semantic Desktop Workshop at the ISWC2005* (2005).
10. Prud'Hommeaux, E., and Seaborne, A. SPARQL query language for RDF. World Wide Web Consortium, Working Draft WD-rdf-sparql-query-20060220, Feb. 2006.
11. Quan, D., Huynh, D., and Karger, D. R. Haystack: A platform for authoring end user semantic web applications. In *International Semantic Web Conference* (2003), pp. 738–753.
12. Sauermann, L. Gnowsis semantic desktop iswc2004 demo. In *Proceedings of the International Semantic Web Conference 2004* (2004).
13. Schaffert, S., Gruber, A., and Westenthaler, R. A Semantic Wiki for Collaborative Knowledge Formation. In *Semantics* (2005).
14. Souzis, A. Rhizome Position Paper, 2004. http://rx4rdf.liminalzone.org/FOAFPaper.