Proceedings of the Third Workshop on Semantic Wikis – The Wiki Way of Semantics

5th European Semantic Web Conference
Tenerife, Spain, June 2008

edited by Christoph Lange

June 2, 2008

Preface

Dear Reader,

almost two years have passed since the last Semantic Wiki workshops in 2006. In the meantime, more than 200 researchers and practitioners have subscribed to our mailing list[1]. Many of the ideas developed in 2006 have been evaluated in practical scenarios, and many of the implementations that had been in a prototype stage then have matured and now serve as an infrastructure for follow-up projects. Wikis, including non-semantic ones, have become more ubiquitous as "weapons of mass collaboration"[2] than ever, which calls for tapping these sources of knowledge in a systematic way. New scientific challenges are found in the larger Semantic Web context and then met and investigated in the controlled testbed of a Semantic Wiki, or they are found in semantic wikis themselves. Semantic Wikis contain in an integrated fashion many of the core challenges of the Semantic Web community: authoring, versioning, interlinked data, semantic browsing, semantic annotating, semantic diffs, semantic search and getting overview. In a Semantic Wiki, all of these are part of a coherent whole, of a tool that must still remain lightweight and easy to use.

We wish to thank *all* authors who spent their nights and days contributing to this topic and thereby made this workshop possible. The high number of good submissions made the work for the programm committee members even more difficult – thank you all for your work. Many thanks also to the ESWC organisation team, which set the stage for this workshop as one out of 12. Let us continue to bring the lively wiki spirit to the Semantic Web and enjoy reading the proceedings.

Galway, June 2008
Christoph Lange, Sebastian Schaffert, Hala Skaf-Molli and Max Völkel

[1]swikig@aifb.uni-karlsruhe.de
[2]Don Tapscott and Anthony D. Williams, "Wikinomics"

Contents

Programme

09:00 – 10:30 Session 1

09:00 – 09:15	Opening Ceremony *Christoph Lange, Sebastian Schaffert, Hala Skaf-Molli, and Max Völkel*
09:15 – 10:00	Keynote: The KiWi Project – Knowledge in a Wiki *Peter Dolog, Aalborg University*
10:00 – 10:20	Best Paper: Towards an Interlinked Semantic Wiki Farm *Alexandre Passant and Philippe Laublet* *—discussion—*
10:30 - 11:00	**Coffee Break**

11:00 - 13:00 Session 2: Lightning Panels

	a group of short talks on related topics, ≈ 15 min. per full paper, 5 min. per short paper, followed by a joint discussion
11:00 – 11:40	**Knowledge Management** Ad-Hoc Knowledge Engineering with Semantic Knowledge Wikis *Jochen Reutelshöfer, Joachim Baumeister, and Frank Puppe* Hypertext Knowledge Workbench *Max Völkel* *—discussion—*
11:40 – 12:30	**Applications in Mathematics** Mathematical Semantic Markup in a Wiki: The Roles of Symbols and Notations *Christoph Lange* A Real Semantic Web for Mathematics Deserves a Real Semantics (Position Paper) *Pierre Corbineau, Herman Geuvers, Cezary Kaliszyk, James McKinna, and Freek Wiedijk* Flyspeck in a Semantic Wiki – Collaborating on a Large Scale Formalization of the Kepler Conjecture *Christoph Lange, Sean McLaughlin, and Florian Rabe* *—discussion—*
12:30 – 12:50	**Annotation** Using Attention and Context Information for Annotations in a Semantic Wiki *Malte Kiesel, Sven Schwarz, Ludger van Elst, and Georg Buscher* RDF Authoring in Wikis *Florian Schmedding, Christoph Hanke, and Thomas Hornung* *—discussion—*

12:50 – 14:00 **Lunch**

14:00 – 15:10 Session 3: Lightning Panels

14:00 – 14:20 **Alternative Interfaces**

AceWiki: Collaborative Ontology Management in Controlled Natural Language
Tobias Kuhn

Next-Generation Wikis: What Users Expect; How RDF Helps
Axel Rauschmayer

—discussion—

14:20 – 14:50 **Other Application Areas**

Integrating a Wiki in an Ontology Driven Web Site: Approach, Architecture and Application in the Archaeological Domain
Andrea Bonomi, Alessandro Mosca, Matteo Palmonari, and Giuseppe Vizzari

Extending the Makna Semantic Wiki to support Workflows
Karsten Dello, Lyndon Nixon, and Robert Tolksdorf

—discussion—

14:50 – 16:00 Session 4: Demo/Poster Session

SWOOKI: A Peer-to-peer Semantic Wiki
Charbel Rahhal, Hala Skaf-Molli, and Pascal-Molli

A Generic Corporate Ontology Lifecycle
Markus Luczak-Rösch and Ralf Heese

Descriptive Schema: Semantics-based Query Answering
Sau Dan Lee, Patrick Yee, Thomas Lee, David W. Cheung, and Wenjun Yuan

Property Clustering in Semantic MediaWiki – Define Your Own Classes and Relationships
Gero Scholz

BOWiki: Ontology-based Semantic Wiki with ABox Reasoning
Joshua Bacher, Robert Höhndorf, and Janet Kelso

including demos of systems presented earlier in talks

. . . and other demos/posters

16:00 – 16:30	**Coffee Break (poster/demo session open to visitors)**

16:30 – 18:00 Session 5: Interactive

16:30 – 16:45	Discussion on Standardisation and Interoperability: Problem Statement
16:45 – 17:45	Teamwork
17:45 – 18:00	Concluding Remarks

18:00 – 19:00	**Joint Dinner (continuing teamwork discussions)**

19:00 – 20:00

Nepomuk Nexus: Reception and Presentation

20:45 SemWiki 2008 Social Event

Organisation

- Christoph Lange
- Sebastian Schaffert
- Hala Skaf-Molli
- Max Völkel

Towards an Interlinked Semantic Wiki Farm

Alexandre Passant[1,2], Philippe Laublet[1]

[1] LaLIC, Université Paris-Sorbonne,
28 rue Serpente,
75006 Paris, France
`firstname.lastname@paris4.sorbonne.fr`

[2] Electricité de France Recherche et Développement,
1 avenue du Géneral de Gaulle,
92141 Clamart Cedex, France
`firstname.lastname@edf.fr`

Abstract. This paper details the main concepts and the architecture of UfoWiki, a semantic wiki farm – i.e. a server of wikis – that uses form-based templates to produce ontology-based knowledge. Moreover, the system allows different wikis to share and interlink ontology instance between each other, so that knowledge can be produced by different and distinct communities in a distributed but collaborative way.

Key words: semantic wikis, wiki farm, linked data, ontology population, named graphs, SIOC

1 Introduction

During the last few years, various Web 2.0 services and principles - such as blogging, wikis, social tagging and social networking - gained interest in corporate environments, leveraging tools that people are more and more used to in their personal life to the enterprise [1]. On the other hand, Semantic Web [2] technologies are used in different business information systems to enrich data integration, querying and browsing, thanks to powerful means to represent knowledge like ontologies and standards to model or query data as RDF and SPARQL.

While some consider Web 2.0 and Semantic Web as being opposite concepts with different origins and goals, we believe as others [3] that these two views should - and even must - be combined to offer easy-to-use but powerful services to end-users. Thus, information systems should benefit from usability and social aspects of Web 2.0 and also from data formalisms of the Semantic Web. It will provide to end users means to collaboratively build, maintain and re-use ontology-based data, a task often dedicated to knowledge management experts, especially in organizations.

In this paper we will describe a semantic wiki-farm system, i.e. a wiki server where communities can setup new wiki instances, called UfoWiki – Unifying Forms and Ontologies in a Wiki – that aims to achieve this goal, currently in use at EDF R&D[3]. The paper is organized as follows. First, we will briefly introduce the limits of classical wikis and various implementations of semantic wikis designed to enhance wiki features thanks to semantics. Then we will introduce the features and the architecture of our semantic wiki farm, as well as novelties compared to current semantic wikis systems, especially the way we combine data and meta-data to keep some information about the knowledge created from wiki pages. We will then describe how people can use one wiki to create ontology instances thanks to form-based templates and emphasize on how the system interlinks data from one wiki to another one but also allows to re-use external data. We will then show how created data can be reused to provide advanced features in a single wiki but also for the complete wiki farm.

2 Wikis and Semantic Wikis for Knowledge Management

2.1 Limits of Traditional Wikis

Among the numerous practices and tools that became popular thanks to Web 2.0, wikis offer new and interesting possibilities regarding collaborative knowledge management. Pages versioning, open plus non-hierarchical editing, hyperlinks and back-links provide useful services to gather and build knowledge within communities and business environments or in open environments as the Web.

Nevertheless, traditional wikis suffer from the difficulty for computers to exploit and reuse the knowledge they contain. A reader could learn from a wiki that EDF is a company that produces nuclear energy in France but a software agent will not be able to easily answer queries like *"Is EDF located in France ?"* or *"List all companies known in that wiki"* without natural language processing algorithms. Indeed, wikis deal with documents and not with machine-understandable representations of real-world concepts and objects, as a reader does when browsing or editing a page. So, a wiki will model that *"There are some hyper-links between a page titled EDF, a page titled France and a page titled nuclear energy"*, but will not be able to deduce anything about the nature of those different objects and their relationships, since pages do not carry enough semantics about the knowledge they contain (Fig. 1).

2.2 The Semantic Web and Ontologies for Better Wikis

To bridge this gap between documents and machine-readable knowledge about real world objects, data must be described in a way software agents interpret and understand uniformly in order to reuse it efficiently. Ontologies [4] and the Semantic Web are effective ways to do so, since they provide common data structures, vocabularies and languages for modeling and querying domains of interest and related individuals. During the last few years and since the first SemWiki workshop [6] various semantic wikis prototypes have been built, combining wiki

[3] Electricit de France, aka EDF, is the leading energy company in France, its R&D department involves about 2000 researchers

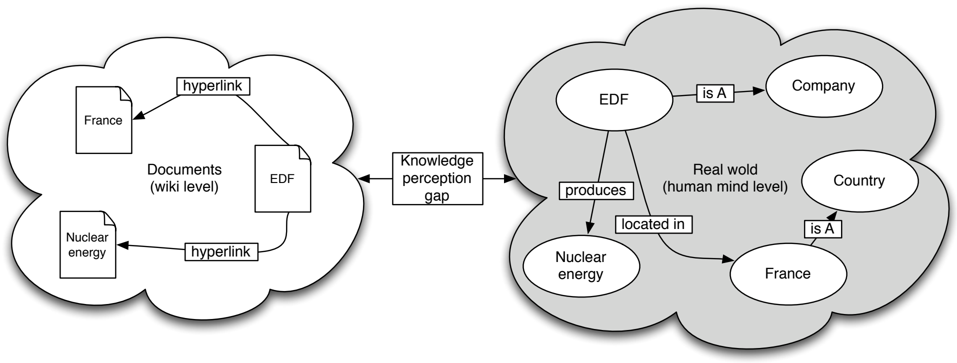


Fig. 1. The gap between documents and real-world knowledge

features and Semantic Web technologies. While tools use different ways to produce this machine-readable data thanks to efforts of their community of users, they all share the common goal of providing value-added services from advanced pages browsing to query answering or even reasoning upon the created dataset.

Systems such as Semantic MediaWiki [7] or SemperWiki [8] require to use a special wiki syntax or to directly embed RDF in order to add semantic annotations to wiki pages. While this is an open approach in the spirit of wiki principles, this can lead to semantic heterogeneity problems since any user can use its own vocabulary to add annotations in a document, making them difficult to re-use. A system like IkeWiki [9] combines plain-text feature of wikis and a dedicated triples-based form interface to help users annotating content by re-using existing ontologies, while OntoWiki [10] can be used as a complete ontology instances editor, with a user-friendly interface that offers different views and browsing and editing interfaces over existing data. Yet, most of those systems require users to have some knowledge about the Semantic Web at a certain time when using it, since they have to deal with namespaces or URIs. This makes the tools difficult to use for people that are not aware of such models, as in business environments where people need to focus on how to use the tools rather than on how he is being build, i.e. benefit from Semantic Web technologies without having to learn them.

In these tools, semantic annotations are mainly used to create and maintain ontology instances and relationships between them, as well as properties, thus providing a real-world and machine-readable representation of the content described inside the pages. They can help to enhance browsing capabilities of the wiki, by suggesting related pages sharing similar instances or listing all pages featuring a certain property as does Semantic MediaWiki. Moreover, new ways to browse the data are available, like in OntoWiki that features map and calendar view of existing data, while some tools provide a back-end RDF store that allows to query data from the whole wiki and embed query results in wiki pages. Finally, some tools also feature inferencing capabilities in order to deduce new knowledge from the current state of the wiki and thus enrich user experience in discovering new knowledge. For example, IkeWIki and OntoWiki can list all instances of a given type taking into account instances of various subclasses. Eventually, it seems important to reference DBpedia [12], a project that aims to represent Wikipedia content in RDF, as well as other semantic wikis, like SweetWiki [11] which does not focus on ontology population but on using semantic web technologies to let users tag their pages and organize those tags, focusing on pages meta-data rather than modeling content of those pages.

3 Modeling a Semantic Wiki Farm

3.1 Main Features of the System

Regarding various aspects of semantic wikis that have been mentioned before, we created UfoWiki, a new semantic wiki farm system - i.e. a wiki server designed to setup and host several wikis - based on the following features, that will be described in the rest of the paper:

– *Ontology-based knowledge representation.* Data created from wiki pages is represented in RDF and is based on a set of ontologies defined by administrators of the wiki in order to avoid semantic heterogeneity problems of data modeling;
– *Usability.* In extent of the previous point and in order to let users easily produce that ontology-based data, we focused on a combination of plain-text and intuitive forms to edit wiki pages, so that users do neither confront to a new syntax or to Semantic Web modeling principles;
– *Interlinking data* While each wiki of the farm acts independent (regarding users that can access it, topics, and modeled knowledge), the system allows different wikis to exchange and interlink their data even if they do not share hyperlinks between each other, thanks to a common knowledge base for the whole system;
– *Modeling both data and meta-data.* While our approach mainly focuses on modeling knowledge contained within wiki pages, we also separately represent the complete wiki server meta-data (wikis, users, pages, tagging actions ...) in RDF, combined with links between those two distinct levels of representation.
– *Immediate reuse of formalized data.* RDF data created among the wikis must be immediately reusable to enhance browsing and querying capabilities of the system, either for a single wiki or the complete farm. Our system uses inline macros, that can provide semantic back-links in the wiki.

3.2 Global architecture

To achieve these goals, our system involves different components. The first part of the architecture consists in a set of ontologies that are used to model RDF data from the wikis, whether it is data about the pages or about their content. For the latter, ontologies must be defined in RDFS or OWL depending on the needs of the knowledge field of the wiki. Regarding the RDF description of wiki pages and user actions, we are using the SIOC ontology [13] and its *type* module[4], a model to describe social media meta-data with unified semantics. We also model tags and tagging actions using the Tag Ontology [14] and the MOAT ontology [15], so that people can give machine-understandable meanings to their tags, especially using URIs of ontology instances created within other wiki pages. Since for all wiki page, data and meta-data are produced within two distinct RDF documents - so that one can export independently each level of representation - we extended the SIOC ontology with a specific property, `embedsKnowledge` in order to formally represent in RDF the link between a wiki page (described in RDF) and the data embedded in it (Fig. 2). This property provides a way to link any instance of `sioc:Item` - and its subclasses - to the URI of a named graph [16], i.e. in practice the URL of a document that contains a set of RDF triples.

Then, the system features its web interface to create wikis, manage wiki forms and browse and edit wiki pages. This interface uses Drupal and is mainly based on a fork of the flexinode module[5] to let wiki owners define their forms. Each form is related to a given class - e.g. people (related to `foaf:Person`) or software project (`doap:Project`) - and each part of the form (a field or a set of fields) can be related to an ontology property and also to a given class, which is used for the autocompletion features of the system. Thus, the editing interface of each wiki combines plain-text and structured parts in order to easily manage the creation of RDF statements according to the ontologies it uses, as we will see on the next section.

The last component of the system is the knowledge base of the wiki farm, storing all created RDF statements thanks to a triple-store, using the 3store[6] API. By storing in real-time all RDF data as well as ontologies in a single place, it offers querying capabilities for the complete data and meta-data of all the wikis, but nevertheless keeps a trace of each statement thanks to its named graphs compliance, so that queries can involve the complete wiki farm data or only data of a given wiki. This store also manages basic inference capabilities (subclasses and subproperties) and supports SPARQL [5] and some SPARUL patterns (SPARQL update[7]) in order to query and update data created from the wiki pages.Moreover, since all wikis of the wiki farm share the same knowledge base, by querying and updating a single RDF store, data can be re-used across wikis. Thus, an ontology instance created in a given wiki can be linked to an ontology instance from another one, even if there is not direct hyperlink between the pages that embeds this knowledge. It allows our system to create knowledge on a distributed way, even between various communities that do not share the

[4] `http://rdfs.org/sioc/types`
[5] `http://drupal.org/modules/flexinode`
[6] `http://threestore.sf.net`
[7] `http://jena.hpl.hp.com/~afs/SPARQL-Update.html`

same wiki but that produce information about the same ontology instances (Fig. 4).

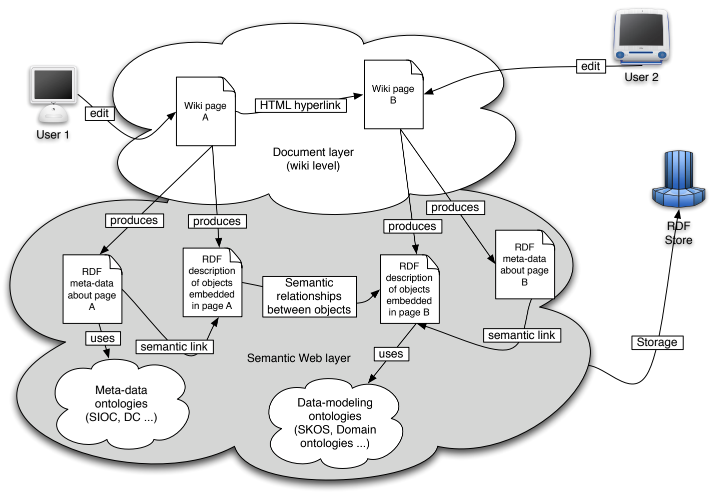

User 1
edit
Wiki page A
HTML hyperlink
Wiki page B
edit
User 2
Document layer (wiki level)
RDF Store
produces
produces
produces
produces
RDF meta-data about page A
RDF description of objects embedded in page A
Semantic relationships between objects
RDF description of objects embedded in page B
RDF meta-data about page B
uses
semantic link
Semantic Web layer
uses
semantic link
Storage
Meta-data ontologies (SIOC, DC ...)
Data-modeling ontologies (SKOS, Domain ontologies ...)

Fig. 2. Architecture of one wiki from the wiki-farm

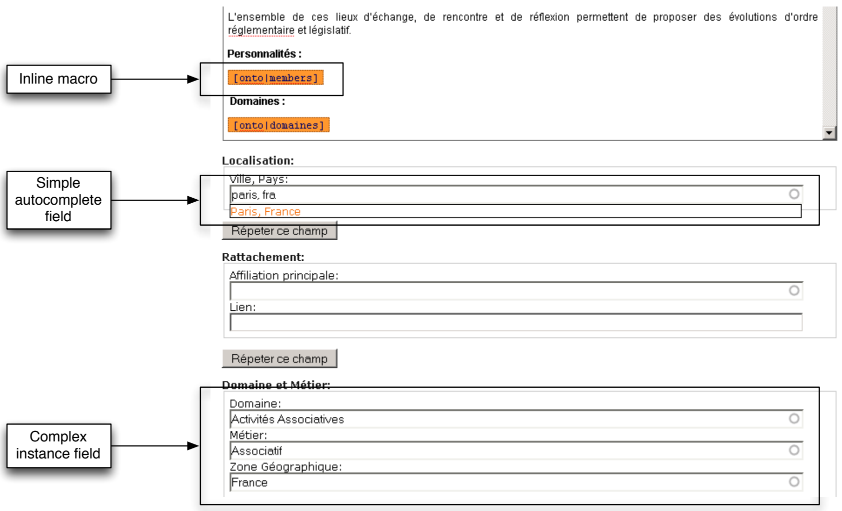
4 Maintaining and interlinking ontology instances between wikis

4.1 Using forms to create and maintain ontology instances

As most semantic wikis, our system automatically creates one main ontology instance for each wiki page, based on the page title. While some wikis do not explicitly assign them a given type and other rely on the page category to define it, our system uses the class assigned to the page form to define it. Regarding definition of properties and relationships of each instance, we use a mix of plain-text and forms in the wiki editing interface, thus separating plain-text content from content to be modeled in RDF, as the Semantic Forms extension[8] for Semantic Wiki or Freebase[9] can do. When creating the page, translation from wiki content to RDF data is then automatically done thanks to the mappings defined by wiki administrators between the form and a set of ontologies. We think that this combination of plain-text and forms to ease the modeling of RDF data (Fig. 3) has numerous advantages:

[8] `http://www.mediawiki.org/wiki/Extension:Semantic_Forms`
[9] `http://www.freebase.com`

- First, as fields are defined by the wiki owner for each type of page and so for each class, users know what kind of knowledge is relevant for the wiki regarding a given page and can focus on essential aspects in this context;
- Moreover, as we kept a simple WYSIWYG field for each page, any other relevant information can be added there. It can also help to participate in evolution of the model itself when regular patterns appear, even if the model must be edited manually in this case;
- Users can benefit from autocompletion features, suggesting possible related instances by querying the RDF store with on-the-fly SPARQL queries, thanks to AJAX technologies;
- At last, in our system, this approach allows to create complex relationships and ontology instances inside a single page. While most semantic wikis allow only to create relationships between existing objects, a form part can correspond to a dedicated class in our system, offering better ways to manage complex ontologies population. Moreover, in the page meta-data representation, we distinguish the main instance and the embedded ones, using two subproperties of `sioc:topic` we especially created to achieve this distinction.



L'ensemble de ces lieux d'échange, de rencontre et de réflexion permettent de proposer des évolutions d'ordre réglementaire et législatif.

Personnalités :

[onto|members]

Domaines :

[onto|domaines]

Localisation:

Ville, Pays: paris, fra

Paris, France

Rattachement:

Affiliation principale:

Lien:

Domaine et Métier:

Domaine: Activités Associatives

Métier: Associatif

Zone Géographique: France

Fig. 3. Wiki editing interface

While each page corresponds to a given ontology instance, instances are also created for each filled relationship field where a class has been assigned. Then, if one later decides to create a wiki page for these instances, properties will be added to the existing ones. Moreover, when instances are not used anymore in any wiki, i.e. do not have any property, they are automatically removed from the

RDF store to avoid orphan instances. From these aspects, the wiki really acts as a collaborative ontology population tool, beneficing from Web 2.0 features to provide this task. An instance can be created by a user, modified by another, then linked to a third one by another one and even can disappear from the knowledge base if a fourth user edit the page that contains its only reference and removes it.

4.2 Interlinking data between wikis

As we saw in the previous section, our system allows various pages of a given wiki to add information about a single ontology instance. For example, we can create an instance in a wiki page and add a relationship from another instance in a different page than the one that creates it. Yet, our system goes further by allowing two different and disconnected wikis to manage information about the same instance in a distributed way, but keeping the trace of which wiki - and which page - helped to create the information. Thanks to the combination of named graphs and the `embedsKnowledge` property we introduced before, the wiki farm can consider either the whole RDF graph, or subgraphs of RDF statements related to a given wiki only (Fig. 4).

Such a scenario might be useful in some corporate environments, where people do not want to allow anyone to access their wiki, but agree on sharing some expertise and data with others. By exporting only some parts of the wiki page in RDF (i.e. the instances and properties created from some fields of the form page), our model allows the webpage itself to be hidden to not-authorized people while the RDF statements can be exported and become available to a larger community. Moreover, due to our technical architecture that uses SPARQL and SPARUL, the system allows wikis that are distributed on a network (and not from the same wiki farm) to exchange data and interlink it the same way, in case they share a single common RDF database. The system currently do not deal with inconsistency between data from different wikis. We think that this issue should be dedicated to some reasoning engine, that would check inconsistency between produced statements thanks to OWL axioms defined in the ontologies.

Moreover, instead of querying the complete knowledge base, queries can be restricted to data created from a single wiki by using this kind of SPARQL query:

```
select ?page ?title
where {
  graph ?data {
    :EDF ?predicate ?object
  } .
  ?page :embedsKnowledge ?data ;
    rdf:type sioct:WikiArticle ;
    dc:title ?title ;
    sioc:has_container <http://example.org/wiki/6> .
  <http://example.org/wiki/6> a sioct:Wiki .
}
```

This process of combining the two levels of representation can also be used in the autocompletion field, by restricting the autocompletion SPARQL queries to data created from a single wiki, rather than to the whole RDF statements.

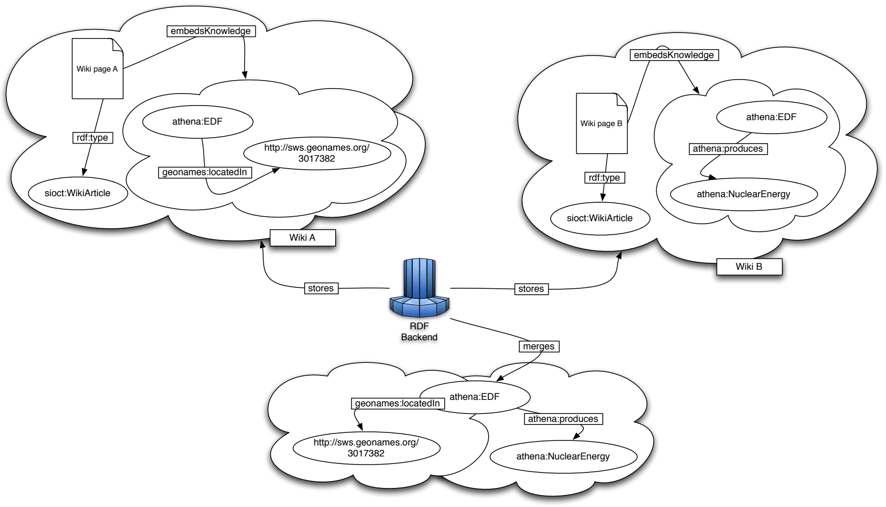


Fig. 4. Interlinking and merging data from different wikis

4.3 Interlinking wiki data with external knowledge

Our system also allows to connect our data to external, publicly available, RDF data. At the moment, a single plug-in is available, to reuse the GeoNames[10] ontology and knowledge base. Each time a form field corresponds to a place and is assigned to the `geonames:Feature` class, the system queries the GeoNames webservice[11] to retrieve the URI of the given instance. Thus, the updated local instances can be linked to external resources, beneficing from a global connection between our data and efforts of communities that help to build such knowledge base. Moreover, we not only link to the URI but also crawl the related RDF file to put in in the wiki knowledge base. Thus, it allow the system to provide geo-location features to end users, without the need for them to type the exact location (i.e. latitude and longitude) of each instance (eg: a people or a company), as they would have done using systems like Semantic MediaWiki and its Semantic Layers extension[12].

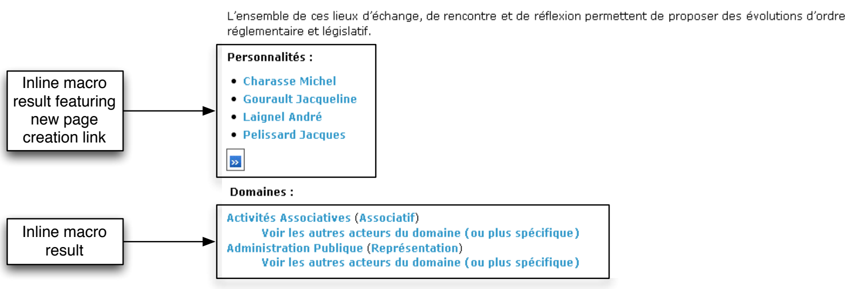
[10] `http://www.geonames.org`
[11] `http://ws.geonames.org`
[12] `http://s89238293.onlinehome.us/w/index.php?title=Main_Page`

In the future, we plan to implement new wrappers and linkage systems for other RDF data, especially ways to link to DBpedia extracted knowledge, which can help to provide additional information about instances created within the wikis, and also contribute to the expansion of the Linked Data Web [17]. Regarding this latest point, linking to data from references datasets can help RDF data from our system to be more easily found on the Semantic Web, thanks to lookup services such a Sindice [18] that help to retrieve all resources using and linking to a given URI.

5 Using created data

5.1 Inline macros

The main feature to enhance wiki browsing capabilities in our system is the use of *inline macros*, similar to inline queries of Semantic Mediawiki. Those macros are defined by wiki administrators themselves, using SPARQL and PHP to render the results and are then called by users in wiki pages with simple hooks. Since all data are based on a set of predefined ontologies, queries can be written without having to deal with semantic heterogeneity problems, as people that would have use different property names for the same one, e.g. `isLocatedIn` versus `has_location`. The system then runs the query over the RDF store when the page loads, so that query results are always up-to-date. While queries can be complex, users simply type function names, with some arguments if needed, to use it in wiki pages. For example, `[onto|members]` will be translated in a query that will retrieve all people that are member of the organization described in a wiki page (Fig. 3, Fig. 5). Such queries take inference capabilities of the system into account, so that, for example, if they must list all organizations instances described in the wiki, they will also lists companies or associations if they have been defined as subclasses of the first one in the ontology. Finally, the administrator can decide that the macro will render a link to add new page in the wiki to create an instance of a given type, thus facilitating the process of creating new data.


L'ensemble de ces lieux d'échange, de rencontre et de réflexion permettent de proposer des évolutions d'ordre réglementaire et législatif.

Personnalités :

- Charasse Michel
- Gourault Jacqueline
- Laignel André
- Pelissard Jacques

Domaines :

Activités Associatives (Associatif)
Voir les autres acteurs du domaine (ou plus spécifique)
Administration Publique (Représentation)
Voir les autres acteurs du domaine (ou plus spécifique)

Fig. 5. Browsing an enhanced wiki page

Moreover, macros can take into account the way we combine modeling of data and meta-data in RDF export of wiki pages, so that a wiki can display a list of pages from another wiki for a given query, as the previous SPARQL snippet showed. It allows one wiki to benefit from the effort of another community done in another wiki.

More generally, such queries can be seen as a way to move from classical wiki back-links to semantic back-links, as we bridged the gap between documents and Semantic Web formalized data. While a typical wiki could list thanks to its back-link feature that an organization page has an incoming link from a people page, our system takes advantage of the data formalism to be more specific about the nature of this link, mentioning that this company employs that person, going from the document to the data layer.

5.2 Advanced data view

Finally, those macros can display results according other rendering inter-faces, such as Google Maps, in case the needed geo-location information is available in the RDF store thanks to the integration of the GeoNames lookup service. Thus, while the result is similar to what can be done with the map view of OntoWiki, users do not have to manually enter the coordinates of each instance (e.g. a company) but simply fill a *"City, (State), Country"* field, that will be used to retrieve the appropriate RDF data - including coordinates - from GeoNames an add it in our knowledge base. Here, we clearly see the benefit of using the same model (i.e. the GeoNames ontology) than an existing RDF dataset to include data from external services at zero-cost. The Fig. 6 displays the output map of a macro that retrieves the location of a given association and all of its members from a single wiki interlinked with GeoNames data in a single SPARQL query.

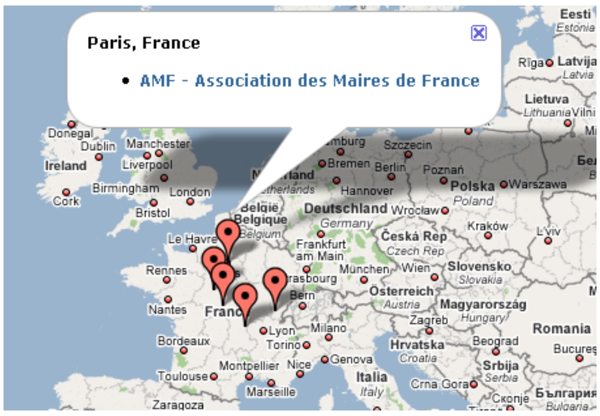


Fig. 6. Map view of the wiki data

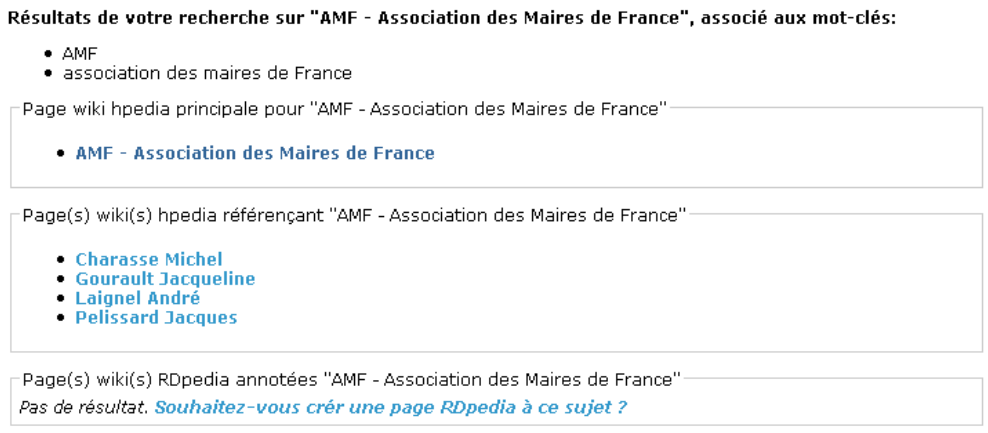
5.3 Semantic search

Another feature of the system is a dedicated semantic search engine, taking into account existing instances described within the wiki (or used in a semantic tagging process) rather than plain-text only when retrieving data. When a user search for a given term in the wiki farm, the system first finds the list of all instances related to this label, using (1) `rdfs:label` that can have be defined thanks to the wiki pages and dedicated forms and (2) the `moat:Tag` instances that contains this term within their label and that are linked to existing instances thanks to a related `moat:Meaning`. Thus, if a user type the search term *"France"*, the system will ask the user if he requires information about *"EDF"* (since it has *"Electricit de France"* as a tag) but also, of course, the *"France"* concept.

Then, the system will list independently:

– All wiki pages - for each wiki, identified by their name - that have this instance as a main topic;
– All wiki pages where the instance is an "alternative" topic (i.e. an instance created within a page);
– All wiki pages *"tagged"* (thanks to MOAT) with this instance.

Thus, it offers various meta-data representation of the wiki.


Résultats de votre recherche sur "AMF - Association des Maires de France", associé aux mot-clés:

- AMF
- association des maires de France

Page wiki hpedia principale pour "AMF - Association des Maires de France"

- **AMF - Association des Maires de France**

Page(s) wiki(s) hpedia référençant "AMF - Association des Maires de France"

- **Charasse Michel**
- **Gourault Jacqueline**
- **Laignel André**
- **Pelissard Jacques**

Page(s) wiki(s) RDpedia annotées "AMF - Association des Maires de France"
Pas de résultat. ***Souhaitez-vous crér une page RDpedia à ce sujet ?***

Fig. 7. Semantic search results example

Moreover, while we do not currently provide user friendly interface to generate new queries or macros, advanced users can run SPARQL queries over the RDF data.

6 Conclusion and future works

In this paper, we described a prototype of wiki that combine structure and Semantic Web modeling capabilities to produce ontology-based and machine-readable data in a collaborative way. We showed how various wikis could be used to model and interlink knowledge about ontology instances in an open and distributed way. We finally showed how such knowledge can be used to enrich functionalities of the wiki. While this system combines some features that already exist in various prototypes, it focuses on usability for end-users, as well as, from the technical side, a way to model and link both data and meta-data, offering capabilities to view different levels of annotation, either from a single wiki or for the complete set of wikis.

The system is currently in use at EDF R&D, where users have created more than 200 instances from various lightweight ontologies. We extensively use the GeoNames integration, making the geo-location feature easy to integrate in order to provide new and interesting ways to browse the wiki content. Inline macros are also useful for end-users since they allow to easily find instances and related wiki pages. For example, we included a macro that lists, in a page dedicated to some company, all other organizations working on the same topics.

Regarding our future works, we will concentrate on adding new value-added functionalities to the wiki for end-users to ease the discovery of relevant information from the set of RDF data, as faceted browsing, as well as interlinking with other existing datasets. We will also focus on how to formalize wiki pages versioning in RDF, in order to see how statements about a given resource can evolve during its lifetime and track more precisely each change of information on a given ontology instance.

Acknowledgements

We would like to thanks the ID-Net team from EDF R&D for their input on the current experiments about our wiki farm.

References

1. McAfee A. P.: Enterprise 2.0: The Dawn of Emergent Collaboration. MIT Sloan Management Review, 47 (3), pp 21-28 (2006).
2. Berners-Lee T., Hendler J., Lassila O.: The Semantic Web. Scientific American, 284 (5), pp 34-43 (2001)
3. Heath T., Motta E.: Ease of interaction plus ease of integration: Combining Web2.0 and the Semantic Web in a reviewing site. Journal of Web Semantics 6 (1) (2008)
4. Gruber T.R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. Formal Ontology in Conceptual Analysis and Knowledge Representation. Guarino N., Poli R. (eds). Kluwer Academic Publishers, Deventer, The Netherlands (1993)

5. Prud'hommeaux E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation 15 January 2008, `http://www.w3.org/TR/rdf-sparql-query/` (2008).
6. Schaffert S., Vlkel M., Decker S.: First Workshop on Semantic Wikis: From Wiki to Semantics (SemWiki2006]) at the 3rd Annual European Semantic Web Conference (ESWC), Budva, Montenegro (2006)
7. Krötzsch M., Vrandecic D., Völkel M.: Semantic MediaWiki. Proceedings of the 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, pp 935-942 (2006)
8. Oren E.: SemperWiki: A Semantic Personal Wiki.: Proceedings of Semantic Desktop Workshop at the ISWC2005, Galway, Ireland (2005)
9. Schaffert S.: IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. 1st International Workshop on Semantic Technologies in Collaborative Applications (STICA'06) (2006)
10. Auer S., Dietzold S., Riechert T.: OntoWiki - A Tool for Social, Semantic Collaboration. Proceedings of 5th International Semantic Web Conference 2006 pp 736-749 (2006)
11. Buffa M., Gandon F.L., Sander P., Faron C., Ereteo G.: SweetWiki: a semantic wiki. Journal of Web Semantics 6 (1) (2008)
12. Auer S., Bizer C., Lehmann J., Kobilarov G., Cyganiak R., Ives Z.: DBpedia: A Nucleus for a Web of Open Data. Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea pp 715-728 (2007)
13. Breslin J.G., Harth A., Bojars U., Decker S.: Towards Semantically-Interlinked Online Communities. Proceedings of the 2nd European Semantic Web Conference (ESWC '05), LNCS vol. 3532, pp. 500-514, Heraklion, Greece (2005)
14. Newman R.: Tag Ontology Design. http://www.holygoat.co.uk/projects/tags/ (2005)
15. Passant A., Laublet P.: Meaning Of A Tag: A collaborative approach to bridge the gap between tagging and Linked Data. Proceedings of the Linked Data on the Web (LDOW2008) workshop at WWW2008, Beijing, China (2008)
16. Carroll J., Bizer C., Hayes P., Stickler P.: Named Graphs, Provenance and Trust. Proceedings The Fourteenth International World Wide Web Conference (WWW2005), Chiba, Japan (2005)
17. Bizer C., Heath T., Ayers D., Raimond Y.: Interlinking Open Data on the Web. Poster, 4th Annual European Semantic Web Conference (ESWC2007), Innsbruck, Austria (2007)
18. Tummarello G., Oren E., Delbru R.: Sindice.com: Weaving the Open Linked Data. Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea 547–560 (2007)

Ad-Hoc Knowledge Engineering with Semantic Knowledge Wikis

Jochen Reutelshoefer, Joachim Baumeister and Frank Puppe

Institute for Computer Science, University of Würzburg, Germany
email: {reutelshoefer,baumeister,puppe}@informatik.uni-wuerzburg.de

Abstract. A couple of semantic wikis have been proposed to serve as collaborative knowledge engineering environments – however, the knowledge of almost all systems is currently based on the expressiveness of OWL (Lite/DL). In this paper we present the concept of a *semantic knowledge wiki* that extends the capabilities of semantic wikis by strong problem-solving methods. We show how problem-solving knowledge is connected with standard ontological knowledge using an upper ontology. Furthermore, we discuss different possibilities to formalize problem-solving knowledge, for example by semantic knowledge annotation, structured text and explicit markups.

1 Introduction

Recently, different approaches of semantic wikis have been presented as applications especially designed for the distributed engineering of ontological knowledge, for example see IkeWiki [1] and OntoWiki [2]. Such systems are used to build ontologies of a specific domain in a collaborative manner and use the well-known metaphor of wikis as the primary user interface. New concepts are usually defined by creating a new wiki page with the name of the concept. Properties of the new concept are described by semantically annotating text phrases of the particular wiki page. The semantic extension of wikis allows for a richer set of possible applications when compared to standard wikis: due to the semantic annotation of content the user is able to semantically search for ontological concepts and/or related concepts. Furthermore, a semantic wiki can be browsed in a semantic way; for example, users can click on semantically relevant (and probably personalized) links that are placed appropriately.

Based on the expressiveness of OWL (Lite/DL) the defined ontologies are able to capture a wide range of general knowledge but lack in the possibility to represent active problem-solving knowledge that is necessary to generate and drive a (semi-)automated problem-solving session with a user. Such knowledge typically relates the class of *findings* – provided as user inputs and describing the current problem – with the class of *solutions* that are derived for the given problem description. In previous work we presented the concept of *knowledge wikis* as an extension of standard wikis adding the possibility to capture, maintain, and share explicit problem-solving knowledge [3, 4]. The presented concept provided strong support to represent explicit knowledge like rules and models, but was not able to capture ontological knowledge beyond subclass hierarchies of solutions and part-of hierarchies of input groups/inputs.

In this paper we describe the (refined) concept of *semantic knowledge wikis* that are interpreted as an extension of semantic wikis. Besides basic ontological knowledge – such as the definition of classes, taxonomic and user-defined properties – a semantic knowledge wiki is able to represent problem-solving knowledge that is applied on selected classes of the ontology. The problem-solving knowledge can be seen as an external knowledge source for changing the values of concept instances; for example in most of the cases the state of a solution instance is set to the value "established". Beyond that it is not intended that the problem-solving knowledge interacts with other knowledge defined in the ontology. In future work, we will consider the exchanging semantics of problem-solving knowledge with the knowledge defined in the ontology, as for example it is currently done in the context of the RIF working group[1]. In summary, a semantic knowledge wiki represents a distributed knowledge engineering environment not only representing semantic relations between the concepts of an ontology but also explicit derivation knowledge.

In contrast to trained knowledge engineers – well educated in knowledge representation and reasoning – we adress experienced users to act as "ad-hoc knowledge engineers". For this reason, the provided interfaces and markups of the wiki need to be as simple as possible in order to lower the barriers of knowledge acquisition. Furthermore, in running projects we experienced the requirement to handle a mixed granularity of the represented knowledge. An interesting research question is to find a collection of suitable markups that can cope with the different types of knowledge and its granularity, respectively.

In this paper, we first describe the basic concepts of a semantic knowledge wiki by introducing an upper ontology for problem-solving. This ontology represents the basic classes and properties that are used in a typical problem-solving process. Moreover, this ontology is used as the basis in every new wiki project, since newly defined concepts are implicitly or explicitly aligned to classes of the upper ontology. We also introduce different types of markups for the definition of problem-solving knowledge. The markups take into account that knowledge can be "formalized" in many different ways, ranging from explicit models and rule bases to semantically annotated text or structured text phrases.

2 An Overview of KnowWE

As discussed above every (semantic) wiki page describes a distinct concept together with formalized properties linking the entity with other classes. In a knowledge wiki we also capture the problem-solving knowledge that is necessary for deriving the particular concept. Then, every wiki page embeds not only semantically annotated text and multimedia but also explicit problem-solving knowledge.

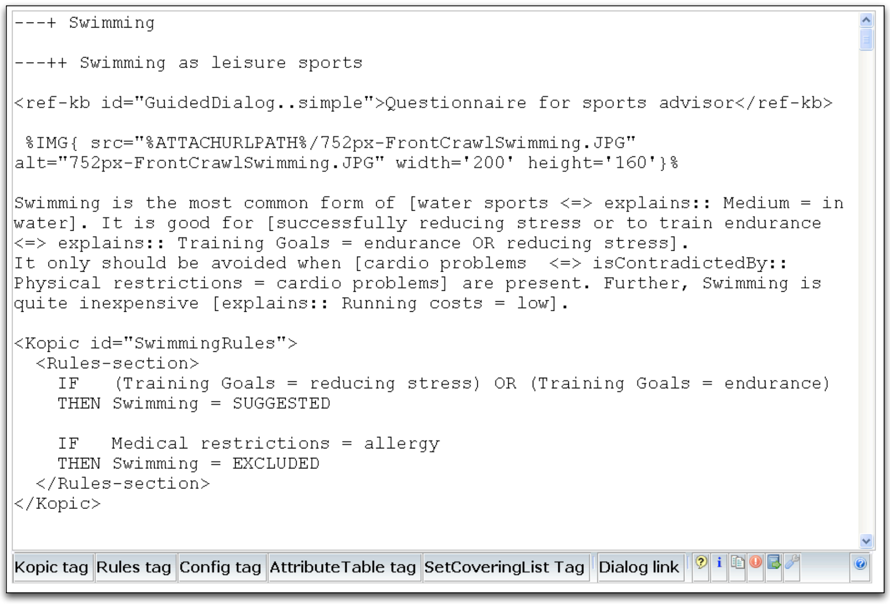
[1] RIF WG wiki: http://www.w3.org/2005/rules/wiki/RIF_Working_Group

As a typical use case, the user of a semantic knowledge wiki can browse the contents of the wiki in a classic web-style –possibly using semantic navigation and/or semantic search features. Moreover, he/she is also able to start an interactive interview where giving a problem description. Based on these inputs an appropriate list of solutions is presented that are in turn linked to the wiki pages representing these particular concepts. Thus, every solution represented in the wiki is considered during a problem-solving process. Instead of using an interactive interview mode the user can enter findings inline by clicking on *inline answers* embedded in the normal wiki page. These inline answers are generated based on the semantic annotations made in the article.

In the following we briefly describe the processes of capturing and sharing knowledge in the semantic knowledge wiki KnowWE [3].

2.1 Knowledge Capture

Semantic annotations and knowledge is edited in the mandatory edit pane of the wiki together with standard wiki content like text and pictures. At the moment

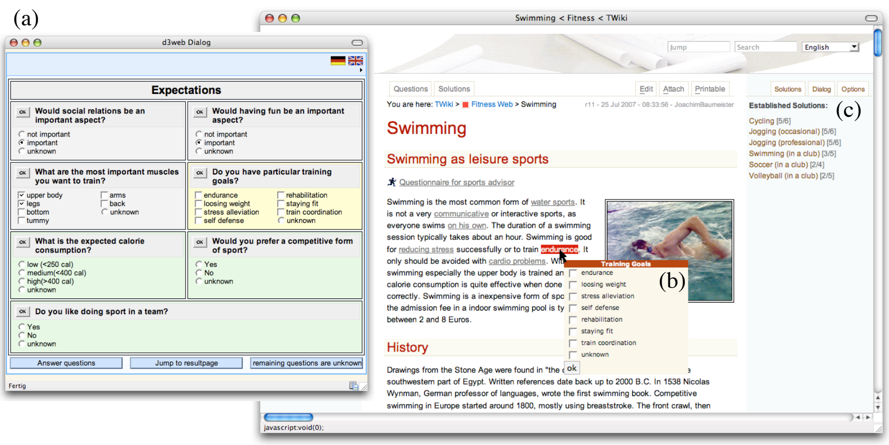


```
---+ Swimming

---++ Swimming as leisure sports

<ref-kb id="GuidedDialog..simple">Questionnaire for sports advisor</ref-kb>

 %IMG{ src="%ATTACHURLPATH%/752px-FrontCrawlSwimming.JPG"
alt="752px-FrontCrawlSwimming.JPG" width='200' height='160'}%

Swimming is the most common form of [water sports <=> explains:: Medium = in
water]. It is good for [successfully reducing stress or to train endurance
<=> explains:: Training Goals = endurance OR reducing stress].
It only should be avoided when [cardio problems  <=> isContradictedBy::
Physical restrictions = cardio problems] are present. Further, Swimming is
quite inexpensive [explains:: Running costs = low].

<Kopic id="SwimmingRules">
  <Rules-section>
    IF   (Training Goals = reducing stress) OR (Training Goals = endurance)
    THEN Swimming = SUGGESTED

    IF   Medical restrictions = allergy
    THEN Swimming = EXCLUDED
  </Rules-section>
</Kopic>
```

Fig. 1. Editing a wiki article of the knowledge wiki KnowWE: A rule base is embedded into the standard wiki text (bottom of the edit pane) and semantic annotations are made in the first paragraph of the text.

KnowWE proposes the textual acquisition of annotations and knowledge in the edit pane, thus a collection of textual markups for annotations and knowledge is required. For a new solution a corresponding wiki page with the solution's name is created. The wiki page includes describing text in natural language and the explicit knowledge for deriving the solution. In this paper, we introduce the concept of distributed knowledge engineering with knowledge wikis, and we demonstrate the methods and techniques using the toy application of a *sports advisor wiki*. The running example considers a wiki providing knowledge about different forms of sports, both in textual and in explicit manner. Explicit knowledge can be used to derive an appropriate form of sport for interactively entered (user) preferences. Besides such a simple recommendation application the wiki can be used for a variety of tasks briefly sketched in the case study.

For example, in Figure 1 we see the edit pane of an article describing the form of sports "Swimming": Standard text is semantically annotated by the particular properties `explains` and `isContradictedBy` for which their meaning is described in the following. Additionally, the first part of a formalized rule base is shown at the bottom of the edit pane. Here, knowledge for deriving and excluding the solution "Swimming" is defined. Besides rules derivation knowledge can be formalized in different manners, for example, explicit set-covering models, structured texts, semantic knowledge annotations. We discuss the different markups in the rest of the paper.



Expectations

Would social relations be an important aspect?
not important
[✓ important]
unknown

Would having fun be an important aspect?
not important
[✓ important]
unknown

What are the most important muscles you want to train?
[✓ upper body]
[✓ legs]
bottom
tummy
arms
back
unknown

Do you have particular training goals?
endurance
loosing weight
stress alleviation
self defense
rehabilitation
staying fit
train coordination
unknown

What is the expected calorie consumption?
low (<250 cal)
medium(<400 cal)
high(>400 cal)
unknown

Would you prefer a competitive form of sport?
Yes
No
unknown

Do you like doing sport in a team?
Yes
No
unknown

Answer questions | Jump to resultpage | remaining questions are unknown

You are here: TWiki > Fitness Web > Swimming r11 - 25 Jul 2007 - 08:33:56 - JoachimBaumeister

Swimming

Swimming as leisure sports

Questionnaire for sports advisor

Swimming is the most common form of water sports. It is not a very communicative or interactive sports, as everyone swims on his own. The duration of a swimming session typically takes about an hour. Swimming is good for reducing stress successfully or to train endurance. It only should be avoided with cardio problems. Wit swimming especially the upper body is trained an calorie consumption is quite effective when done correctly. Swimming is a inexpensive form of spo the admission fee in a indoor swimming pool is ty between 2 and 8 Euros.

Training Goals
endurance
loosing weight
stress alleviation
self defense
rehabilitation
staying fit
train coordination
unknown
ok

History

Drawings from the Stone Age were found in "the c e southwestern part of Egypt. Written references date back up to 2000 B.C. In 1538 Nicolas Wynman, German professor of languages, wrote the first swimming book. Competitive swimming in Europe started around 1800, mostly using breaststroke. The front crawl, then

Established Solutions:
Cycling [5/6]
Jogging (occasional) [5/6]
Jogging (professional) [5/6]
Swimming (in a club) [3/5]
Soccer (in a club) [2/4]
Volleyball (in a club) [2/5]

Fig. 2. Possible interfaces for a problem-solving session: interview mode (a) and in-place answers (b). Derived solutions are presented immediately in the wiki (c).

When saving a wiki article the included knowledge is extracted and compiled to an executable knowledge base. In consequence, we arrive at one separate knowledge base for each wiki article capturing the derivation knowledge for the corresponding concept of the article. With the increasing number of wiki articles the number of knowledge bases will also increase. As we see in the next section the concepts created in the wiki are naturally aligned due to the upper ontology of the knowledge wiki. Furthermore, the developers are encouraged to reuse a pre-defined application ontology which is build on the fixed upper ontology. However, ad–hoc defined findings not corresponding to the application ontology can be easily aligned by expressing alignment rules, that match these concepts with concepts of the application ontology.

2.2 Knowledge Sharing

Besides standard ways of knowledge sharing in (semantic) wikis like (semantic) searching and browsing we provide two ways for a more interactive knowledge sharing in knowledge wikis: first, every wiki page can generate an interactive interview from the included knowledge base by asking questions represented by the findings used in the knowledge base, as for example depicted in Figure 2 a. Second, semantic annotations in text are used to offer inline answers, i.e., clickable text in the article asking for meaningful facts corresponding with the highlighted text, cf. Figure 2 b. In both ways a new finding instance is entered into the knowledge wiki corresponding to the clicked finding object. The instance is propagated to the *knowledge wiki broker* that in turn derives solutions based on the entered findings. The propagation paths of the broker are depicted in Figure 3. The entered finding instances are propagated to the broker which aligns

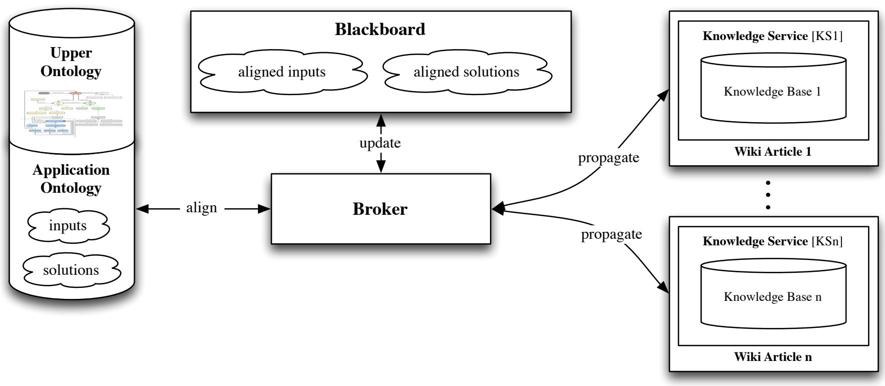


Fig. 3. Blackboard architecture for the distributed problem–solving of the knowledge wiki KnowWE.

the findings to a global application ontology (building on an upper ontology) and then files the aligned instances to a central blackboard. Also, the broker notifies all knowledge bases contained in the wiki for the new fact added to the blackboard and gives the possibility to derive solutions based on the currently available facts. Derived solutions are also propagated by the broker as new facts. With the use of this simple broker/blackboard architecture we are able to allow for a distributed problem-solving incorporating the multiple knowledge bases of the wiki. Therefore, all solutions represented in the knowledge wiki can be derived at any page; already derived solutions are presented at the right pane of the wiki as for example shown in Figure 2 c. Here, the solutions "Cycling" and "Jogging" were derived as the most appropriate solutions, even though the findings were entered on the page describing the solution "Swimming". The presented example can be seen as a specialized case of *semantic navigation*.

In comparison to our previous work [5], we focus on the knowledge acquisition issues of a semantic knowledge wiki: we discuss an upper ontology as an enabling technology for problem-solving and semantic annotation, and we introduce alternative ways to enter problem-solving knowledge into a wiki, for example by using semantic annotations and structured texts using NLP techniques.

3 An Upper Ontology for Classification Tasks

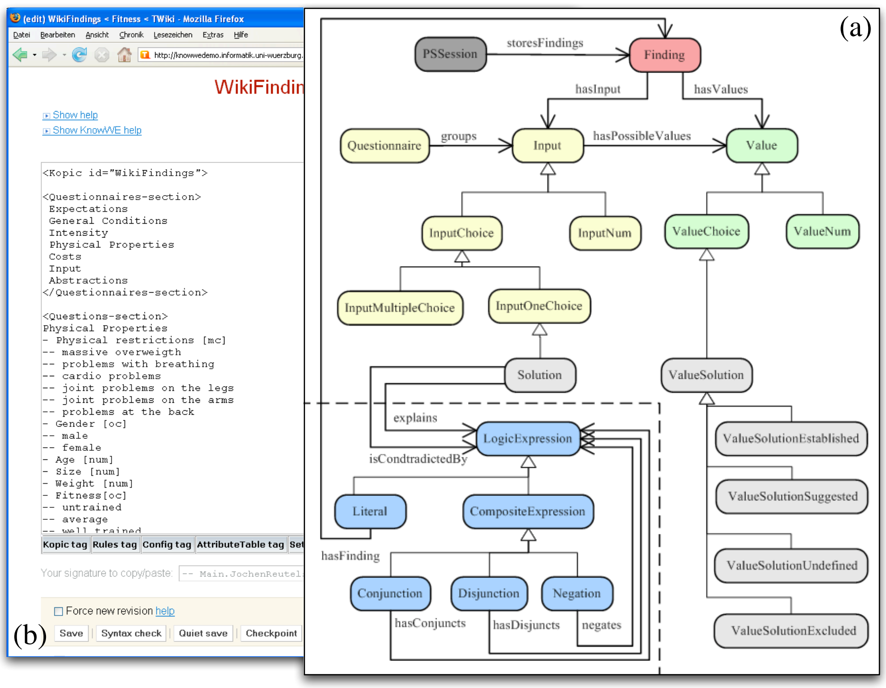
Studer et al. [6] introduced in detail how the input data and output data of problem-solving methods is structured by specific ontologies. Similarly, we introduce an upper ontology for the classification problem class used in semantic wiki context. The upper ontology is the foundation of every new wiki project. The upper ontology includes the general definitions of findings and solutions that are the basic elements of a problem-solving task. A new wiki project maintains an application ontology by creating specific findings and solutions that are subclassing the concepts of the upper ontology.

3.1 Concepts and Properties of the Upper Ontology

In the following we describe an upper ontology for problem-solving that is used in the semantic knowledge wiki KnowWE. An excerpt of the upper ontology is shown in Figure 4 a; we omitted less important concepts like textual inputs and values for clarity. All unlabeled associations denote *subClassOf* relations.

The concept *Input* plays a key role and allows to describe the world state as a set of variables. Inputs are grouped by the concept *Questionnaire* to structure inputs into meaningful clusters. The two main subclasses of Input are *InputChoice* and *InputNum* to define variables with discrete (named) values and numerical value ranges, respectively. Accordingly, a corresponding value subclassing *Value* is assigned to each Input. The concept *Solution* denotes a special type of a one-choice input that is not entered by the user but derived by a knowledge base, thus representing the final output of a problem-solving session. The value range of a solution is restricted to the possible values *Established*, *Suggested*, *Undefined*, and *Excluded* for expressing the current derivation state of the particular solution.

A concrete problem-solving session is represented by an instance of the concept *PSSession* where a knowledge consumer describes his/her current problem by entering the values of the corresponding observed inputs. The reasoning processes of different users are completely independent from each other as each user is describing his own specific problem instance. The assignment of a value to a



```
<Kopic id="WikiFindings">

<Questionnaires-section>
 Expectations
 General Conditions
 Intensity
 Physical Properties
 Costs
 Input
 Abstractions
</Questionnaires-section>

<Questions-section>
Physical Properties
- Physical restrictions [mc]
-- massive overweigth
-- problems with breathing
-- cardio problems
-- joint problems on the legs
-- joint problems on the arms
-- problems at the back
- Gender [oc]
-- male
-- female
- Age [num]
- Size [num]
- Weight [num]
- Fitness[oc]
-- untrained
-- average
-- well trained
```

Fig. 4. a) The upper ontology of the semantic knowledge wiki KnowWE. b) a part of the input definitions (WikiFindings page) of a sports advisor demo.

corresponding input is captured by the concept *Finding*, depicted at the top of Figure 4 a.

The proposed knowledge wiki allows for a free and general use of various, alternative knowledge representations to actually derive the concrete solutions defined in the application ontology. For this reason, derivation knowledge is represented in the upper ontology only in a very general manner, as depicted in the left lower corner of Figure 4 a. For the specification of problem-solving relations we introduce the general object properties *explains* and *isContradictedBy* with *Solution* as the domain and *LogicExpression* as its range: The abstract concept *LogicExpression* is subclassed by *CompositeExpression*, which allows fo the composition of logical expressions over findings by the subclasses *Conjunction*, *Disjunction*, and *Negation*. The semantics of the properties *explains* and *isContradictedBy* are described more detailed in the context of the XCL knowledge representation (eXtensible Covering List) in Section 4.

3.2 Creation and Maintenance of the Application Ontology

The concrete inputs and solutions of a new wiki application are defined in the *application ontology*. With the two special wiki pages WikiFindings and Wiki-

Solutions the structure of the application ontology is maintained: The (user) inputs together with their values are defined in the article WikiFindings using the special textual markup `Kopic`. Within this tag we textually define new inputs and their corresponding values together with questionnaires grouping the particular inputs. Defined solutions and inputs of the application ontology are automatically subclassing the corresponding concepts of the upper ontology. Figure 4 b shows a part of the input hierarchy of the sports advisor demo already mentioned before. With one dash we denote a new input followed by its type definition, for example *[oc]* for one-choice inputs and *[num]* for specifying numeric inputs. For choice inputs the possible values are listed in the following lines with an additional preceding dash. Analogously, the solutions of the application ontology are organized in the article WikiSolutions. In summary, the application ontology is created and modified in a wiki-like way, i.e., by editing the wiki pages WikiFindings and WikiSolutions.

For the knowledge engineering task we propose an evolutionary process model as introduced by Fischer [7]: at the beginning of a project an initial effort – called *seeding phase* – has to be made to create a simple but usable basic application ontology. In the working progress the ontology is extended and restructured in cyclic *evolutionary growth* and *reseeding* phases.

4 Simple Knowledge Representations for Ad-Hoc Knowledge Engineers

In the previous section we introduced an upper ontology for problem-solving representing the basis of an application ontology. This application ontology defines the specific inputs and solutions of the particular application domain. As mentioned earlier, every wiki page is able to capture problem-solving knowledge relating defined inputs with the corresponding solution of the particular wiki page. Since we aim to motivate experienced user to act as ad-hoc knowledge engineers the problem-solving knowledge needs to meet the following requirements:

1. easy to understand and formalize,
2. a compact and intuitive textual representation,
3. yields a transparent and comprehensible inference process.

This will help to break down the initial barriers when

1. making personal knowledge explicit,
2. inserting it into a wiki page text,
3. and finally evaluating the created knowledge.

As complexity in knowledge representation and inference constitutes a major barrier for contribution we face the trade-off between simplicity and expressiveness. Beneath simplicity, we have to make an open world assumption concerning the derivation knowledge for a solution concept. During the development process only a part of the total imaginable/retrievable amount of knowledge is present in the knowledge wiki. Thus, it is desirable that any subset of the (fictional) complete knowledge can be used to derive the best possible results with respect to the given subset. Further, this subsets of course need to be extensible easily. A knowledge representation meeting this requirements needs to be based on small knowledge units which are to a great extend independent of each other. On the one hand this characteristics enables to build very small but already working knowledge bases which then can be extended subsequently step by step. On the other hand the knowledge bases show some robustness with respect to the deletion of small parts and redundant definitions by accident or ignorance which is important within the scope of "ad-hoc knowledge engineering".

Considering the concrete problem-solving process of a knowledge consumer we need to regard that it is unpredictable which exact subset of inputs is actually observable, as real world situations are often diverse and wicked. Thus, it is desirable to design the knowledge and the inference process in a way, that each subset of entered findings will yield appropriate solution ratings. Of course, in this case a confidence value based on the size of the entered finding set needs to be presented along with the resulting solution states. To cope with these challenges we provide the wiki user a simplified but easily extensible version of set-covering models [8], called eXtensible Coverings Lists (XCL). In our approach the extensible-covering model of a solution basically consists of a set of n findings. The weighting of the findings set to $1/n$ as default and we use the individual similarity function. Apriori, the resulting function for a solution rating is therefore restricted to m/n, where m is the number of correct findings and n the number of total findings defined by the model. The possible result spans a real range from $[0, 1]$ which is partitioned into four disjoint intervals representing the corresponding values of a solution *ValueSolutionEstablished*, *ValueSolutionSuggested*, *ValueSolutionUnclear*, and *ValueSolutionExcluded*. Obviously, there is a crucial lack of sensitivity concerning single findings, which is bounded by $1/n$. To improve the limited expressiveness we introduced several extensions to this simple finding list, for example combined findings and exclusion knowledge. In the following section XCL and its textual markup is explained in more detail.

4.1 Embedding Simple Knowledge in Wiki Texts

We present three different textual markups to integrate simple knowledge as described above in standard wiki texts. To demonstrate the similarities and differences of the markups we define knowledge in each way for the solution *Swimming* corresponding to the sports advisor demo.

eXtensible Covering Lists (XCL) The most compact representation of the covering knowledge is its formalization as an *eXtensible Covering List* that is wrapped in a `Kopic` tag. As noted earlier the `Kopic` tag can be placed anywhere in the wiki article and is also used to define other classes of knowledge like the solutions and findings of the application ontology. The names of the inputs and the corresponding values are matched against the definitions made in the application ontology found in the WikiFindings article (cf., Figure 4b). In the following example an XCL model for the solution *Swimming* is shown. Each line represents a positive coverage of the finding by the solution and is called *explains* relation. The order of the listed findings is not relevant for the inference process and thus is arbitrary. If the domain knowledge is already available as informal text, then it denotes a simple task to transfer the key findings described in the text into basic findings contained in an XCL.

```
<Kopic id=‘‘Swimming scmodel’’>
  <XCL-section>
    Swimming  {
      medium = water,
      Type of sport = individual,
      Training goals ALL {endurance, stress reduction},
      Running costs = medium,
      Trained muscles = upper part,
      Trained muscles = back
    }
  </XCL-section>
</Kopic>
```

During the inference process the best rated solution is chosen. A solution is rated by comparing the findings defined in the XCL against the findings entered by the user. The rating of a solution is expressed by its *covering score*. As mentioned before, this numeric score is mapped to four predefined solution states *Unclear* (default), *Suggested*, *Establised* and *Excluded*.

The basic XCL representation can be extended in multiple ways: Besides the simple listing of findings shown above the XCL representation offers further elements to extend/refine the expressiveness and selectivity of the covering model that are briefly discussed in the following (the textual markup is given in paratheses):

Exclusion knowledge `[--]`: This marks a relation such that the derivation of the solution becomes impossible to be positively derived when the relation is fulfilled. This type of relation is called *isContradictedBy* and it sets the state of the solution to *Excluded* when fulfilled. Such a constraining relation is defined by two minus signs in brackets (`[--]`) at the end of the relation line, as for example shown in line 7 of the markup shown below.

Required relations `[!]`: Relations can be marked as *required* by using a bracket containing an exclamation mark (`[!]`). Then a solution can only be established as a possible output when all required relations are fulfilled, i.e., the corresponding findings were positively entered by the user. An example is shown in line 2 in the following markup.

Sufficient relations `[++]`: By adding a bracket with two plus signs (`[++]`) to a relation we define this solution to be a sufficient relation. Then, the solution is always established if the corresponding finding is fulfilled. We call this relation *isSufficientlyDerivedBy*. It is important to know that contradicting relations are dominating sufficient relations.

Adding weights `[ `*num*`]`: In the initial version every explains relation is equally important when compared to the other relations of the XCL, thus having the default value 1. The default value can be overridden for relations in order to express their particular importance. In the textual notation the weight is then entered in brackets at the end of the relation definition, for example `[2]` to double weight a relation. See line 8 in the following markup for a further example.

Logical operators (`AND`, `OR`): A complex relation can be created by combining relations by logical operators. For example in line 2 of the model shown below the findings *medium = water* and *Type of sport = individual* are connected by the logical or-operator (`OR`). The resulting knowledge demands that either *medium = water* or *Type of sport = individual* needs to be observed to fullfil the relation. The three basic operators of propositional logic *or*, *and* and *not* can be used.

Threshold values: When rating a solution the numeric covering score is mapped to a solution state. The mapping function is defined by the threshold values (`establishedThreshold` and `suggestedThreshold`). In most cases the internal default threshold values are adequate, but for distinct solutions they can be overriden as shown in line 12-13 of the following example model. In the example, 70% of the expected and observed findings need to be correctly observed to set the solution tot the state *Established*, and 50% to set the solution to the state *Suggested* (higher states overwrite lower states). Further, with `minSupport` we specify how many percent of the findings defined in the XCL model need to be entered by the user in order to activate the solutions rating process.

The following markup shows a refined version of the previous model of the solution *Swimming* using the described elements. Essentially, the already defined relations were mostly refined by relational extensions like sufficient, contradicting and necessary properties.

```
<Kopic id=‘‘Swimming scmodel’’>
  <XCL-section>
    Swimming  {
      medium = water OR Type of sport = individual [!],
      My favorite sports form = swimming [++],
      Training goals ALL {endurance, stress reduction},
      Favorite color IN {red, green, blue},
      Running costs = medium,
      Running costs = nothing [--],
      Trained muscles = upper part [2],
      Trained muscles = back [2],
      Physical problems = skin allergy [--],
      Type of sport = group [--],
    }[ establishedThreshold = 0.7,
       suggestedThreshold   = 0.5,
       minSupport           = 0.5
      ]
  </XCL-section>
</Kopic>
```

Although the presented representation is experienced to be compact and intuitive for most of the users, it is clearly separated from the remaining text of the wiki article, which usually describes the same concept and its knowledge in natural language. This separation increases the risk of “update anomalies”, for example if users are modifying or extending the wiki article but not the corresponding part of formalized knowledge. Therefore, a tighter integration of formal knowledge and informal text of a wiki article is desirable, and for this aim we introduce two possible approaches in the following.

Inline Annotation Many semantic wikis use inline annotation techniques to describe semantic properties between concepts of the ontology, for example Semantic MediaWiki [9]. Using special properties of the upper ontologies like `explains` and `isContradictedBy` we are able to capture set-covering knowledge by evaluating semantic annotations. The following example shows sentences describing the solution “Swimming”, where text phrases are annotated by the property `explains` for defining positive set-covering relations and the property `isContradictedBy` for the definition of exclusion rules. For example, the first two lines state a relation between the solution `Swimming` (the concept of the page) and the finding `Medium = in water`: the first expression after the opening brace and before the `<=>` is the textual part of the sentence, that will be rendered in the view mode of the article, shown in Figure 2 b. The phrase before <=> can be omitted; then the preceding word before the annotation is highlighted and related to the corresponding relation. The following part of the annotation states the name of the property (e.g., `explains`, `isContradictedBy`) followed by two colons. After that, the actual finding related to the solution concept is specified, i.e., the range of the given property. In the topic view the annotated text can be used as an interview method with inline answers. In this case the input of a set-covering relation is posed as question to the knowledge consumer as shown in Figure 2 b.

```
Swimming is the most common form of [water sports <=> explains::
Medium = in water]. Swimming is good for successfully [reducing
stress or to train endurance <=> explains:: Training Goals =
stress alleviation OR Training Goals = endurance]. It only
should be avoided when [cardio problems <=> isContradictedBy::
Physical restrictions = cardio problems] are present. Further,
Swimming is quite inexpensive [explains:: Running Costs = low].
```

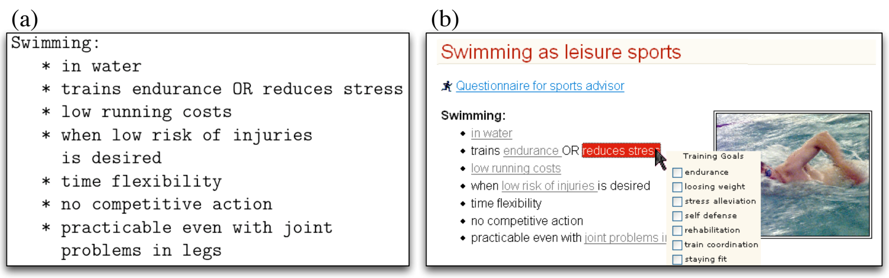
The semantic annotation of existing sentences means less knowledge acquisition workload compared to the explicit markups introduced before. Although, standard wiki text is tightly integrated with formal knowledge the readability of the text suffers from annotations as shown above.

Structured Text A more radical approach is to omit semantic annotations when possible and to use NLP techniques for annotating distinct parts of the wiki text. In the context of our work, we are able to use “structured texts” since 1) the available text needs to be mapped to a rather simple knowledge representation, and 2) we can employ the given application ontology as background knowledge for the concept extraction task. This is very similar to the approach of the DBpedia project[2] described by Auer et al. [10] which uses the article structure of wiki pages to formalize knowledge about the described topic. Instead of creating RDF-triple we want to generate problem-solving knowledge for the classification task. The key problem here is the matching of the natural language expressions to the concepts of the application ontology.

Using this technique we assume, that a distinct block of a wiki article is tagged as a “structured text”. Then, this block is parsed in order to identify findings for which set-covering relations are created. In a first step we are working with semi-structured texts (e.g. bullet lists, tables). In the following example in Figure 5 a we show a bullet list in standard wiki syntax, where each line contains one (combined) finding explaining the solution `Swimming`. While the inline annotations expect exact matches we applied some lightweight linguistic methods for matching findings in structured texts, for example simple string matching combined with stemming and synonym lists already lead to fairly good

[2] DBpedia: http://www.dbpedia.org

results. In the simplest case we can identify an input name that is defined in the application ontology together with a corresponding value name, for example as found in line 5 of Figure 5 a: the text phrase “when low risk of injuries is desired” yields the finding “risk of injury = low”. The finding defined by this input-value-pair tuple can be added as a set-covering relation of the solution *Swimming*. Sometimes only an input is listed and humans implicitly refer to a default value of this input, especially for inputs only having “yes” and “no” as possible values (e.g. “practiced outdoor”). For this type of input we assume “yes” as the default value when creating set-covering relations. For other inputs the default answer needs to be defined in the application ontology, otherwise the finding cannot be completely identified. Another popular case is appearance of the value name in the line as the only indicator of a finding, for example in line 2 `in water`. If the corresponding input can be clearly identified due to the unique


(a)

```
Swimming:
   * in water
   * trains endurance OR reduces stress
   * low running costs
   * when low risk of injuries
     is desired
   * time flexibility
   * no competitive action
   * practicable even with joint
     problems in legs
```

(b)

Swimming as leisure sports

Questionnaire for sports advisor

Swimming:
- in water
- trains endurance OR reduces stress
- low running costs
- when low risk of injuries is desired
- time flexibility
- no competitive action
- practicable even with joint problems i

Fig. 5. a) Wiki syntax of a bullet list in structured text, b) automatically annotated text phrases in a semi-structured text.

name of the found value, then we automatically generate a relation accordingly. We often can disambiguate the occurrence of such a finding, since most of the times humans only reduce the text to the value name if this would not result in an ambiguity. All identified findings are implicitly annotated and can be used to provide inline answers as shown for example in Figure 5 b.

4.2 Knowledge Representations in Explicit Markup

Although in various forms extensible the XCL representation has limited expressiveness for some type of domain knowledge.

```
<Rules-section>
  // Abstraction rule r1 for body mass index calculation
  IF   (Height > 0) AND (Weight > 0)
  THEN  BMI = (Weight / (Height * Height))

  // Derivation rule r2  for solution Running
  IF ("Training goals" = endurance)
  AND ("BMI"<30) AND NOT("Physical Problems" = knees)
  THEN Running = SUGGESTED
</Rules-section>
```

In order to allow for the creation of complex knowledge relations we provide further knowledge representations to be used by more experienced users. The textual markup of alternative representations was introduced in [5]. In the context of this paper we briefly show the definition of rules for the derivation of abstractions – for example to be used for concept mapping – and rules for the derivation of solutions. The rules section shown above contains two rules taken from the sports demo. The first calculates the body mass index (BMI) and the second rule sets the solution *Running* to the value *Suggested*. In addition, we provide decision trees and several table-based representations for rules and set-covering relations.

5 Case Studies

We have implemented the presented approach with the system KnowWE [3], a semantic knowledge wiki, that is still under lively development. For an extensive evaluation of the applicability of our system we made a student based case study considering the creation of knowledge wikis with a group of 45 students. Within about three weeks 11 recommendation systems with a total amount of about 700 knowledge bases containing rules an set-covering relations were created. Beyond further student projects, the system is currently used in the context of the BIOLOG Europe project (http://www.biolog-europe.org). Its purpose is the integration of socioeconomic and landscape ecological research results in order to produce a common understanding of the effects of environmental change on managed ecosystems. Inter- and trans-disciplinary research projects with economists yielded socioeconomic knowledge on how the biodiversity can be supported in managed agro-ecosystems. The research results are present in the form of large amounts of (unstructured) knowledge on landscape diversity of life with respect to the given landscape structures, management decisions and their progression [11], for example described in papers, data sheets, and further multimedia content.

The project wiki LaDy (for "Landscape Diversity") aims to support domain specialists and interested people to collect and share knowledge in the context of the BIOLOG project. The knowledge appears at different levels of detail ranging from textual descriptions and multimedia to formal knowledge covering the effects on landscape diversity. Typically user inputs consider the description of the investigated landscape, whereas solutions are defined with respect to the biodiversity of various taxa, different ecosystem services and management decisions. At the moment, the knowledge wiki is under development incorporating ecological domain specialists distributed all over germany.

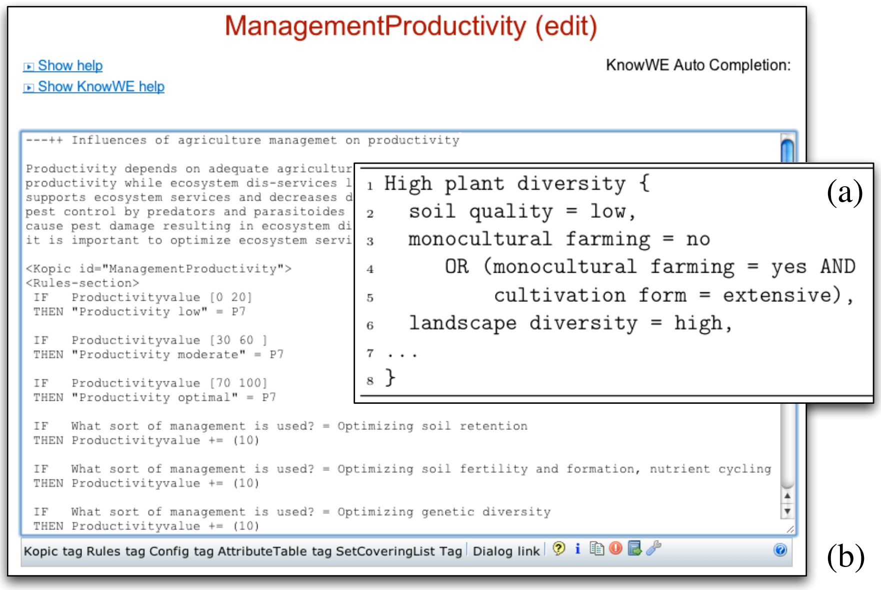
The participating domain specialists have neither background in knowledge representation nor in ontology engineering, and therefore the interfaces need to be as simple and intuitive as possible. In the various kick-of meetings we learned that a simple set-covering list representation was experienced to be intuitive and suitable for the first steps. After some simple examples the requirements of the users concerning the expressiveness usually grew, and in many cases these requirements could be covered by extended covering lists as shown for example in the following. In Figure 6 a, the solution `High plant diversity` is defined by a set-covering list of constrained findings, where the second item (line 3–5) is a complex finding combining a list of atomic findings by a disjunction and a conjunction (simplified example shown).

In other cases we offered to transform the existing knowledge to a rule base, since the largest degree of expressiveness can be provided by a rule-based representation. An excerpt for a rule base is shown in Figure 6 b where the value of management productivity is defined in correspondence of inputs such as genetic diversity and optimized soil retention. Providing a platform for both, exchanging textual knowledge and implementing explicit rules on ecosystem behaviour, LaDy provides a service to condense and to communicate knowledge needed for an efficient management of ecosystem services.

6 Conclusions

We have introduced the concept of a semantic knowledge wiki with the implementation KnowWE that extends the known OWL-based expressiveness of other semantic wikis by active problem-solving capabilities. Whereas related approaches provide strong support to capture ontological knowledge – for example see [1, 2] – our main goal is to make the engineering of executable problem-solving knowledge as simple as possible thus supporting the formation of ad-hoc knowledge engineers. For this reason, we presented an upper ontology connecting ontological knowledge with strong problem-solving knowledge, and we introduced different possible ways to formalize problem-solving knowledge, for example semantic knowledge annotations, (semi-)structured texts, and explicit knowledge markups.

In the future we are planning to improve the power of natural language to be used as direct input for knowledge acquisition, incorporating more linguistic methods and controlled languages. Related work is reported by the Attempto


ManagementProductivity (edit)
Show help
Show KnowWE help
KnowWE Auto Completion:
Kopic tag Rules tag Config tag AttributeTable tag SetCoveringList Tag
Dialog link
(a)
(b)

```
---++ Influences of agriculture managemet on productivity

Productivity depends on adequate agricultur
productivity while ecosystem dis-services l
supports ecosystem services and decreases d
pest control by predators and parasitoides
cause pest damage resulting in ecosystem di
it is important to optimize ecosystem servi

<Kopic id="ManagementProductivity">
<Rules-section>
 IF   Productivityvalue [0 20]
 THEN "Productivity low" = P7

 IF   Productivityvalue [30 60 ]
 THEN "Productivity moderate" = P7

 IF   Productivityvalue [70 100]
 THEN "Productivity optimal" = P7

 IF   What sort of management is used? = Optimizing soil retention
 THEN Productivityvalue += (10)

 IF   What sort of management is used? = Optimizing soil fertility and formation, nutrient cycling
 THEN Productivityvalue += (10)

 IF   What sort of management is used? = Optimizing genetic diversity
 THEN Productivityvalue += (10)
```

```
High plant diversity {
  soil quality = low,
  monocultural farming = no
     OR (monocultural farming = yes AND
         cultivation form = extensive),
  landscape diversity = high,
...
}
```

Fig. 6. a) Excerpt of a simplified version of a set-covering model for deriving "High plant diversity" b) rules describing the value of management productivity depending on inputs such as genetic diversity and optimized soil retention.

project [12], where *Attempto Controlled English* (ACE) uses a knowledge representation that is equivalent to first order logic and is also being combined with a wiki technology in the system AceWiki. Besides more sophisticated methods to formalize knowledge we have further research questions that need to be adressed in the future: In a distributed setting existing methods and tools for the evaluation and the refactoring need to be reconsidered and refined in order to facilitate the maintenance and quality of an evolving semantic knowledge wiki.

References

1. Schaffert, S.: IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In: STICA'06: 1st International Workshop on Semantic Technologies in Collaborative Applications, Manchester, UK (2006)
2. Auer, S., Dietzold, S., Riechert, T.: OntoWiki – A Tool for Social, Semantic Collaboration. In: ISWC'06: Proceedings of the 5th International Semantic Web Conference, Berlin, Springer (2006) 736–749
3. Baumeister, J., Reutelshoefer, J., Puppe, F.: KnowWE – Community–based Knowledge Capture with Knowledge Wikis. In: K-CAP '07: Proceedings of the 4th International Conference on Knowledge Capture, New York, NY, USA, ACM (2007) 189–190

4. Baumeister, J., Puppe, F.: Web-based Knowledge Engineering using Knowledge Wikis. In: Proceedings of Symbiotic Relationships between Semantic Web and Knowledge Engineering (AAAI 2008 Spring Symposium). (2008)
5. Baumeister, J., Reutelshoefer, J., Puppe, F.: Markups for Knowledge Wikis. In: SAAKM'07: Proceedings of the Semantic Authoring, Annotation and Knowledge Markup Workshop, Whistler, Canada (2007) 7–14
6. Studer, R., Eriksson, H., Gennari, J., Tu, S., Fensel, D., Musen, M.: Ontologies and the Configuration of Problem-Solving Methods. In: Proc. of the 10th Knowledge Acquisition for Knowledge-based Systems Workshop, Banff. (1996)
7. Fischer, G.: Seeding, Evolutionary Growth and Reseeding: Constructing, Capturing and Evolving Knowledge in Domain–Oriented Design Environments. Automated Software Engineering **5**(4) (1998) 447–464
8. Reggia, J.A., Nau, D.S., Wang, P.Y.: Diagnostic Expert Systems Based on a Set Covering Model. Journal of Man-Machine Studies **19**(5) (1983) 437–460
9. Krötzsch, M., Vrandecić, D., Völkel, M.: Semantic MediaWiki. In: ISWC'06: Proceedings of the 5th International Semantic Web Conference, LNAI 4273, Berlin, Springer (2006) 935–942
10. Auer, S., Lehmann, J.: What Have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content. In: The Semantic Web: Research and Applications. (2007) 503–517
11. Otte, A., Simmering, D., Wolters, V.: Biodiversity at the Landscape Level: Recent Concepts and Perspectives for Multifunctional Use. Landscape Ecology **22** (2007) 639–642
12. Kuhn, T.: AceRules: Executing Rules in Controlled Natural Language. In: Proceedings of First International Conference on Web Reasoning and Rule Systems. Volume 4524 of LNCS. (2007) 299–308

Hypertext Knowledge Workbench*

Max Völkel

FZI Forschungszentrum Informatik
Universität Karlsruhe (TH), Germany
voelkel@fzi.de, http://www.fzi.de/ipe/

Abstract. This paper presents a tool for semantic personal knowledge management called *Hypertext Knowledge Workbench* (HKW), an editor and browser for semantic personal knowledge models. The tool is designed to be used by a single person to manage her personal notes about any topic that seems relevant. Existing wikis and semantic wikis represent content as pages with a title and content. Hypertext-based Knowledge Workbench (HKW) offers a more powerful yet simple to use conceptual model and allows entering and using knowledge in different degrees of granularity and formality.

1 Introduction

In 1958, Peter F. Drucker [8] was among the first to use the term *knowledge worker* for someone who works primarily with information or one who develops and uses knowledge in the workplace.

Frand and Hixon [10] were among the first to use the term Personal Knowledge Management (PKM) in an academic context, followed by [2, 18]. Higgison [12] defines personal knowledge management as "managing and supporting personal knowledge and information so that it is accessible, meaningful and valuable to the individual; maintaining networks, contacts and communities; making life easier and more enjoyable; and exploiting personal capital".

Polanyi [24] makes a distinction between explicit knowledge encoded in artefacts such as books or web pages, and tacit knowledge which resides in the individual. Concerning *explicitness* of knowledge, Nonaka and Takeuchi [20] distinguish two kinds of explicitness: explicit and tacit. Later works [6, 19] conclude that external and internal (tacit) knowledge are two extremes on a spectrum. Maurer [17] states that knowledge resides in the heads of people and the computer can only store "computerized knowledge" which is to be understood as "shadow knowledge", a "weakish image" of the real knowledge. In PKM, we often deal with knowledge that is somewhere in the middle of these extremes. E. g. note-taking is a core activity of PKM. An individual creates an external representation for internal concepts. Later, the external representation is internalised

* Acknowledgments: Research reported in this paper has been financed by the EU in the Social Semantic Desktop project NEPOMUK (IST-FP6-027705, http://nepomuk.semanticdesktop.org).

again to re-activate the knowledge in the individuals mind. If somebody writes a short informal note to himself it is often completely meaningless to others. The knowledge is thus not fully externalised – Yet such a note is an external reminder about some knowledge that the author would otherwise forget. E. g. a short note like “coffee” could mean anything from “buy coffee”, or “don’t forget to have a coffee with an old friend next Tuesday” to “download and install the latest source code management tool called coffee”. In HKW, knowledge can be represented and organised within each note as well as by connections among notes.

HKW is intended to serve as a personal log book for all kinds of knowledge. E. g. it can be used to record ideas, bookmarks, text modules for or from documents, contact data of persons, notes on sport exercises, cocktail recipes, places to travel to, list of friends who enjoy Chinese cuisine, favoured artists and their interrelationships, nice restaurants, etc. These items often have relations that cannot be represented in specialised tools, e. g. friends living in the cities where I know a nice restaurant; text modules uses in which documents?; idea resulted in which publication?; who knows more about this topic?; who has a record from this rare music artist? The user can decide whether she represents e. g. a relation between a person and a music artist as a tag, a link, typed link or just in natural language. HKW is designed to relief the user from deciding “Will it be worth the effort to take this note?” and “Where do I file this?” Authoring in HKW is “pay as you go”, with very low initial costs yet expressive modelling abilities in the same tool. A user can record any thought at low costs and *might* develop it later into more structured, more linked, more important notes. HKW can represent structured knowledge from any domain, because the relations and types *can* be changed and extended by the user at runtime. Informal knowledge represented as plain text or structured text needs no defined types and relations.

1.1 Existing Approaches

Existing approaches to personal note management are **paper-based approaches** such as sticky-notes, paper notebooks, and a “Zettelkasten” [16]. The paper-based approaches are hard to automate, e. g. in a Zettelkasten one has to traverse the links from note card to note card manually. On paper it is especially costly to *change* the content of notes or relations to other notes. Sometimes a complete note has to be rewritten. Also there is no ability for full-text queries or semantic queries.

There are many **software-based approaches** for note management; almost all of them allow full-text search and a virtually unlimited amount of personal notes. Unfortunately, search is not enough for PKM. As Barreau and Nardi [3] point out, there is also a need to organize notes so that a note is even found if the user was just querying for a related note or browsing to a certain folder or category. On their desktop, users manage their content in files; these files are organized in folders. Personal notes often have internal structure and relations to other notes. These relations are hard to manage in plain text files and the file system.

Software tools designed for knowledge management offer ways to create links between notes. Such tools include blogs, wikis, PIM tools, mind map tools, and desktop wikis. These tools offer better usability for quickly creating structured content and linking it with exiting notes. However, if a knowledge base grows to thousands of notes, mere store and retrieval reaches its limits. E. g. which of your recipes were tasty when you tried but do not contain too much chilli, because Dirk does not like chilli? You want to go to Heidelberg – what was the name of the nice restaurant you went to together with Claudia last summer? Or imagine you keep a diary where you occasionally note your running times. Suddenly you weight more than the year before. Did you went running less often than the year before? Most real-world use-cases are not restricted to single domains, rather it is typical to have links across multiple domains (here: recipes, people, travel, sports).

It is obvious that a system can support searching and browsing better, the more structured, more formal the content is. However, requiring to formalise *all* content is too expensive. Therefore plain ontology editors are not good PKM tools. A system should let the user decide at each step how much formalisation effort to put into the system. For numerical knowledge, spreadsheet applications such as Microsoft Excel, are an example for the *ability* to formalise knowledge. A user can either use a spreadsheet just like a sheet of paper or link different cells into complex formulas. In a similar way, semantic wikis allow the same flexibility for conceptual knowledge. A user *can*—but is not forced to—formalise link and page types.

Semantic wikis are designed and used not only for collaborative use but also for personal knowledge management (PKM) [21, 23]. Semantic wikis do allow stepwise formalisation of content: First a page is created, then filled with text, spell-corrected, structured, re-structured, and linked to other pages. Then links are typed and pages linked to categories. Ironically, just like with paper-based approaches, *changing* things is not that easy in semantic wikis. Tasks such as moving content from one page to another or renaming a relation require typically an administrator to run scripts over the database. Second, a common use-case of PKM tools is the need to import knowledge from external sources. In most semantic wikis, the import of semantic data needs to be represented by artificially generated wiki syntax inserted into pages.

The **Hypertext Knowledge Workbench (HKW)** is different from semantic wikis. HKW (a) is backed by a more flexible data model, (b) allows to create and *change* formal statements easily, and (c) integrates authoring, structuring and formalisation.

1.2 Outline

The remainder of this paper analyzes the conceptual model of wikis and semantic wikis and compare it to the conceptual model of HKW, called Conceptual Data Structures (CDS) (Sec. 2). In Sec. 3 we analyse the annotation abilities of CDS compared to semantic wikis. We report on the HKW user interface in Sec. 4. In Sec. 5 we compare CDS and HKW to related work before we conclude in Sec. 6.

2 Conceptual Models

A conceptual model is not to be confused with a technical data model. As an analogy, compare the technical level of the file system, which consists of nodes, blocks, node tables and file allocation tables, with the user-perceived model: A strict tree containing folders and files.

Wikis have a simple conceptual model: Each page has a name, which is a short string that can be typed on a keyboard and often be remembered by the users. Attached to each page title is the wiki page content. Page content consist of a longer string of characters which are interpreted by the wiki render engine to produce HTML. Special syntax in the page content is interpreted as links to other pages. Links are established by referring to other wiki page titles. Empty wiki pages represent concepts with no description attached.

In **semantic wikis**, e. g. Semantic MediaWiki [14] (SMW), the user can state link *types* and use a part of the page content itself as target of the semantic links. However, it is not possible to link to these content snippets with another link. The content snippets in SMW are not first-class entities.

The conceptual model of HKW — called **Conceptual Data Structures** (CDS) — is a generalisation of that model. CDS [26, 27] consists of two layers: The CDS data model and the CDS relation hierarchy. The semantics of CDS re-use some of the semantics of RDFS [11]. Basically CDS uses sub-classes and sub-properties, extended with inverse properties. The next two sections describe the CDS data model and relation hierarchy. The complete CDS framework has been implemented as Java API, available from `http://cds.xam.de`.

2.1 CDS Data Model

The CDS data model[1] is a technical data model to represent knowledge. However, most parts are *intended* to be directly exposed to the user as a conceptual model.

The conceptual model consists of six primitive types. Fig. 1 shows the technical data model with the conceptual parts shown in bold. We describe briefly how the conceptual model works from a user's perspective. This conceptual model is exposed to the user in HKW.

Model A *Model* can be opened or saved, just like other documents. A *Model* is a container for items. Such items might be items with content (i. e. *NameItems*, *ContentItems*, *Relations*, or *Statements*) or automatically appearing triples.
Under the hood: Each *Model* has a URI. In RDF, each model is represented as a Named Graph [4].

ContentItem These are simple text snippets, like the content of a wiki page or a sheet of paper. One can write anything into such a note. The system records automatically creation date, change date and author. One may use wiki syntax to format the note, or link to *NameItems* by referring to their

[1] It is the successor of the "Semantic Web Content Model" (SWCM) presented in [25]

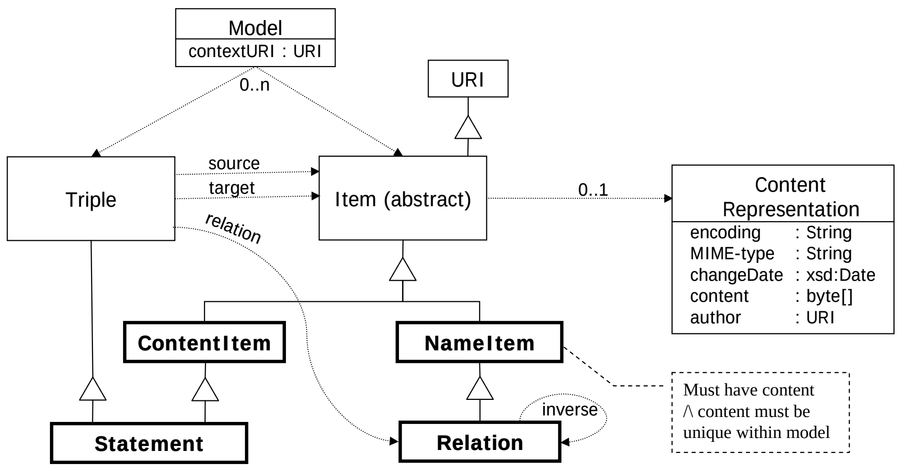


Fig. 1. CDS data model

name. Like a wiki page, a *ContentItems* content size can range from very short to very long. It may also be the case that a *ContentItem* has no content. *Under the hood:* Each item (i. e. *NameItems*, *ContentItems*, *Relations*, and *Statements*) has a unique URI to reference it. Even if the content of an item changes, the references remain the same. No content can appear outside of *Items*. Each piece of content is thus addressable, which makes it easier to record metadata and introduce versioning. Currently there is no versioning of content. The representation of content is modelled after resources on the web (c. f. [9]). All metadata is represented in Resource Description Framework (RDF), binary content is stored in a separate content repository. RDF can store binary data only inconveniently as `xsd:base64Binary` types. Current triple stores are not designed to handle large byte streams.

NameItem A *NameItem* is just a name. It is like the title of a wiki page or the name of a file within a folder. Like in a wiki, a *NameItem* must be unique within a model. *NameItems* do not have any content (besides their name) but it is easy to create links to other items. *NameItems* allows a user to jump directly to certain entities in the knowledge model, similar to navigating to a known wiki page. Other items can be reached indirectly through search or browse actions.
Under the hood: Representing names as first class citizens is handy to allow a user to rename e. g. a *NameItem* without having to change each *Statement* using it. A *NameItem* has three restrictions on its content: First, a *NameItem* has always exactly one content attached to it. Second, a *NameItem* may have only simple textual content, i. e. no line breaks and no wiki syntax. The content of a *NameItem* can easily be entered a human (possibly using an auto-completion mechanism). The MIME-type of the content is always “text/plain”. Third, this textual content must be *unique* within a

knowledge model: No two *NameItems* can have different URIs and the same content. Formally, for two *NameItems* n_1 and n_2 the following holds:

$$n_1.content = n_2.content \Leftrightarrow n_1 = n_2 \tag{1}$$

All these constraints do not hold for *ContentItems*: They can be empty or two items can have the same content. There may also be *ContentItems* having the same content as a *NameItem*.

Relation A *Relation* describes the way two items are linked to each other. There are many pre-defined relations, but one can create new relations as needed. The built-in relation hierarchy is described in the next section. Each relation always has an inverse relation defined, so one can view each link from both sides. E. g. “Dirk knows Claudia” is the same as “Claudia is known by Dirk”, if “knows” has the inverse “is known by”.
Under the hood: A *Relation* is a special kind of *NameItem*. This implies there can not be two different relations having the same name. Each *Relation p* has a *mandatory inverse* Relation $-p$. The inverse of the inverse of a *Relation p* is again p:

$$-(-p) = p \tag{2}$$

In CDS each statement of the form (s, p, o) can additionally be rendered as $(o, -p, s)$ with $-p$ being the inverse of p.

Triple A CDS *Triple* is like a semantic link in a wiki. It connects any two items and denotes the type of the link by a *Relation*. Triples appear in the user interface e. g. as the result of queries using inferencing.
Under the hood: Triples are not items. They have no metadata or content attached and a user has first to promote the triple to a *Statement* before it can be annotated.

Statement A *Statement* is both a *Triple* and an *Item*. As such, it also has a creation date, an author and may even be annotated or have textual content. Annotating statements is useful to state the source of knowledge e. g. in discussion systems.
There are two ways to create *Statements*. First, a user can use the name of a *NameItem* in the text of a *ContentItem*. Then the system automatically creates a statement, where the originating *ContentItem* is recorded as the author. This allows the user to back-trace the origin of a statement. Second, the user can directly create a *Statement* between any kind of item.
Under the hood: Statements are represented on RDF via a kind of reification. Different from RDF, each CDS *Statement does* entail the ground triple (s, p, o). For every *Statement* (s, p, o), the inverse *Statement* $(o, -p, s)$ is inferred, where $-p$ is the inverse of p. That is:

$$\forall s, p, o : (s, p, o) \mapsto (o, -p, s) \tag{3}$$

Note that the URI of the *Statement* does not influence the asserted facts. It is possible that different statements with the same URI assert the same facts but e. g. having different annotations. *Statements* with the *same* URI must have the same content, i. e. the same source, relation and target.

2.2 CDS Relation Hierarchy

On top of this conceptual data model, CDS defines a hierarchy of relations.

The CDS built-in relations have been selected after an analysis of a number of existing information structures in applications used for PKM. The core relation types deal with order, hierarchy, different forms of annotation (i. e. free-text annotations, tagging, and formal typing), and generic hyper-links. As the relation hierarchy is represented in the CDS data model, the user can (and should) extend it in CDS-based tools such as HKW.

The relation hierarchy itself is represented by the built-in relation `cds:hasSubRelation`. Each lower-level *Relation* implies the higher-level *Relations*, just like in RDF Schema (RDFS). The complete relation hierarchy is described in [28].

3 Semantic Annotations

In semantic wikis, users can usually either state the type of link or embed metadata about the current page using special syntax constructs. Some wikis (i. e. SemperWiki) allow embedding arbitrary RDF statements on a page). There is a high variance between the capabilities of semantic wikis to create semantic data. [22] compares some popular semantic wikis with respect to their ability to create annotations. We now analyse HKW with the dimensions given in [22]: Attribution, granularity, representation distinction, terminology reuse, object type, and context.

Attribution Most wikis attribute their annotations to the page where the user is editing the wiki syntax. In HKW (and CDS) links are external to the items. Like in other semantic wikis it is possible to create links between entities via syntax constructs. Internally, such syntax constructs are parsed and result in generated statements. However, only HKW allows creating links which do no originate from a wiki page. This allows e. g. easily importing an existing ontology into the knowledge base without the need to append generated wiki syntax to existing text. Furthermore, each *Statement* in HKW is an *Item* itself and each *Item* in HKW has an author and a creation date. This allows recording the provenance of *Statements* conveniently. The downside of this extended flexibility is versioning. Existing semantic wikis where all semantic statements originate in wiki syntax, the page-based wiki versioning is re-used. In HKW, this is not possible. In fact, HKW has currently no versioning. In the future, we plan to add two kinds of versioning: Item-level versioning for the textual content and model-level versioning for the semantic statements.

Granularity HKW allows creating *ContentItems* of varying size, ranging from single words to full documents, just like the page content of a wiki page. However, different from wiki pages, *ContentItems* have no name, therefore it is cognitively easier to create a large set of them: Imagine an author of a long document would have to name each paragraph in the document individually!

In wikis, every entity that one wants to link to must be written on its own wiki page.

In HKW, the amount, size and relation between *NameItems* and *ContentItems* can be chosen by the user. Therefore HKW can be used to mimic a classic wiki (with *NameItem-ContentItem* pairs), but can also be used in other ways, i. e. linking *ContentItems* with *ContentItems* and *NameItems* with *NameItems*.

Representation Distinction There have been long debates in mailing lists and workshops over the role of URIs that are used to locate information resources on the web and to denote abstract concepts. For practical everyday personal knowledge management tasks this distinction does not matter much. The individual users create their Items with URIs bound to a personal unique namespace, so there is no danger of accidental overlap. As we separate the *NameItems* from the *ContentItems*, they have different URIs. *NameItems* may contain only a short string. Line breaks and formatting are not allowed. This reduces *NameItems* more or less to labels (but unique ones). So there is the ambiguity whether one talks about the *NameItem* or the concept denoted by the *NameItem*. However, the same ambiguity is in our everyday life: Do we talk about the *name* "Dirk Hageman" or the *person* "Dirk Hageman" when we say "Dirk Hageman"? For pragmatic reasons HKW does not distinguish these two cases in the data model. Note that the information-resource-like *ContentItems* are distinguished from the name-like *NameItems*.

Terminology Reuse In HKW, the user is usually not confronted with URIs, so she cannot directly re-use existing URIs. There are two options around this: One is to create explicitly an *Item* with a given URI, another one is to import an existing ontology as a set of *NameItems*. The ontology needs either to have unique labels or labels have to be changed to become unique at import time.

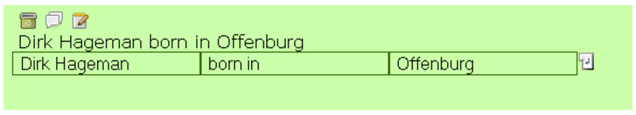
Object Type Most semantic wikis link either to other wiki pages or literal values. In HKW, there is no such distinction. All textual content is addressable by URIs. So the object type is neither page nor literal but *Item*.

In the future, we will integrate the CDS API with the NEPOMUK backbone. This will allow the user to link any semantic desktop item with any other semantic desktop or CDS *Item* and vice versa.

Context As the annotations in HKW are stored as first-class citizens, provenance and context can be stored. E. g. for each *Item* the author and creation date are automatically recorded. In addition to that, each *Statement* can be annotated further by the user. Note that none of the semantic wikis analysed in [22] had a way to record context.

4 User Interface

Fig. 4 shows a screen-shot of the HKW GUI[2] focusing on the *NameItem* “Dirk Hageman”. The screen-shot shows the auto-completion list after entering the letter “c”. The screen is divided into seven colored areas. Below the “Dirk” item, HKW shows the *Items* related via the relation `cds:hasDetail`. E. g. the statement “Dirk Hageman”–“born in”–“Offenburg” is rendered here. This tells the user that ‘born in” is a `cds:hasSubRelation` of `cds:hasDetail`. The inverse relation of `cds:hasDetail` is `cds:hasContext`. *Items* related to the selected *Item* via `cds:hasContext` are rendered above the “Dirk” item. The other colored boxes represent other CDS core relations. The GUI shows relations always in their most specific box. Items are only rendered in different boxes at the same time if the user assigned multiple super-relations to a relation. Behind the word


Dirk Hageman born in Offenburg

Dirk Hageman	born in	Offenburg

Fig. 2. Statement Widget

“Offenburg” there are icons allowing the user to navigate to the *Statement* “Dirk Hageman”–“born in”–“Offenburg”. In a *Statement* view (c. f. Fig. 2), the *Statement* can be changed. E. g. the user can change the *Relation* or create a new source or target. Auto-linking is supported wherever possible. Most actions in

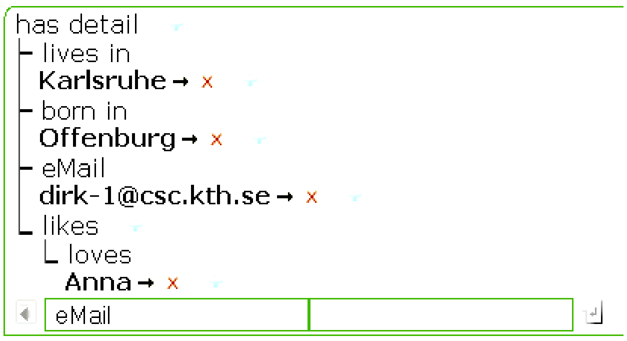

```
has detail
├ lives in
│ Karlsruhe → ×
├ born in
│ Offenburg → ×
├ eMail
│ dirk-1@csc.kth.se → ×
└ likes
  └ loves
    Anna → ×
```

Fig. 3. Relation Tree Widget

HKW are performed in the *Relation Tree Widget* (c. f. Fig. 3). Each relation

[2] Try online or download from `http://wiki.ontoworld.org/wiki/CDS_Editor`

tree widget represents on of the CDS relations (detail, context, before, after, tag, type, annotation, annotation member, related, source, or target). The widget allows deleting existing statements by pressing the little red 'X'; creating new items, relations and corresponding statements. By pressing the blue plus icon next to an existing relation the widget expands and shows two form fields. One to enter a relation name, pre-filled with the relation where the blue plus was selected from and one form field to create a new item or select among the existing *NameItems*. The user is free to enter a different relation name into the relation field, again supported by auto-completion. At any time new relations can be created by simply typing in a new *Relation* name. The *Relation* is automatically a sub-relation of the main *Relation* of a box. I. e. creating a new *Relation* in the top right box ("has annotation") creates a sub-relation of "has annotation". Inverse *Relations* are automatically created and named "inverse of ...". The name can easily be change by the user in a single place. This allows creating new semantically interlinked items easily. If the user enters a longer text or uses line breaks, the system assumes the user creates a *ContentItem*. For short text, the system suggest existing *NameItems* or creates new ones. As a result, a user can always just start typing in the address bar, no matter whether a concept-like *NameItem* or a note-like longer *ContentItem* is going to be created.

The HKW prototype has been realised with the *Google Web Toolkit* (GWT), an open-source AJAX-enabled web user interface toolkit by Google Inc. Styling is done via *Cascading Style Sheets* (CSS). Due to CSS issues, the tool works currently only properly in the Firefox browser. This is not a problem as Firefox is available free of charge for all platforms. GWT applications are web-applications, which can run in any servlet container. The typical use case is to run the server locally on the desktop.

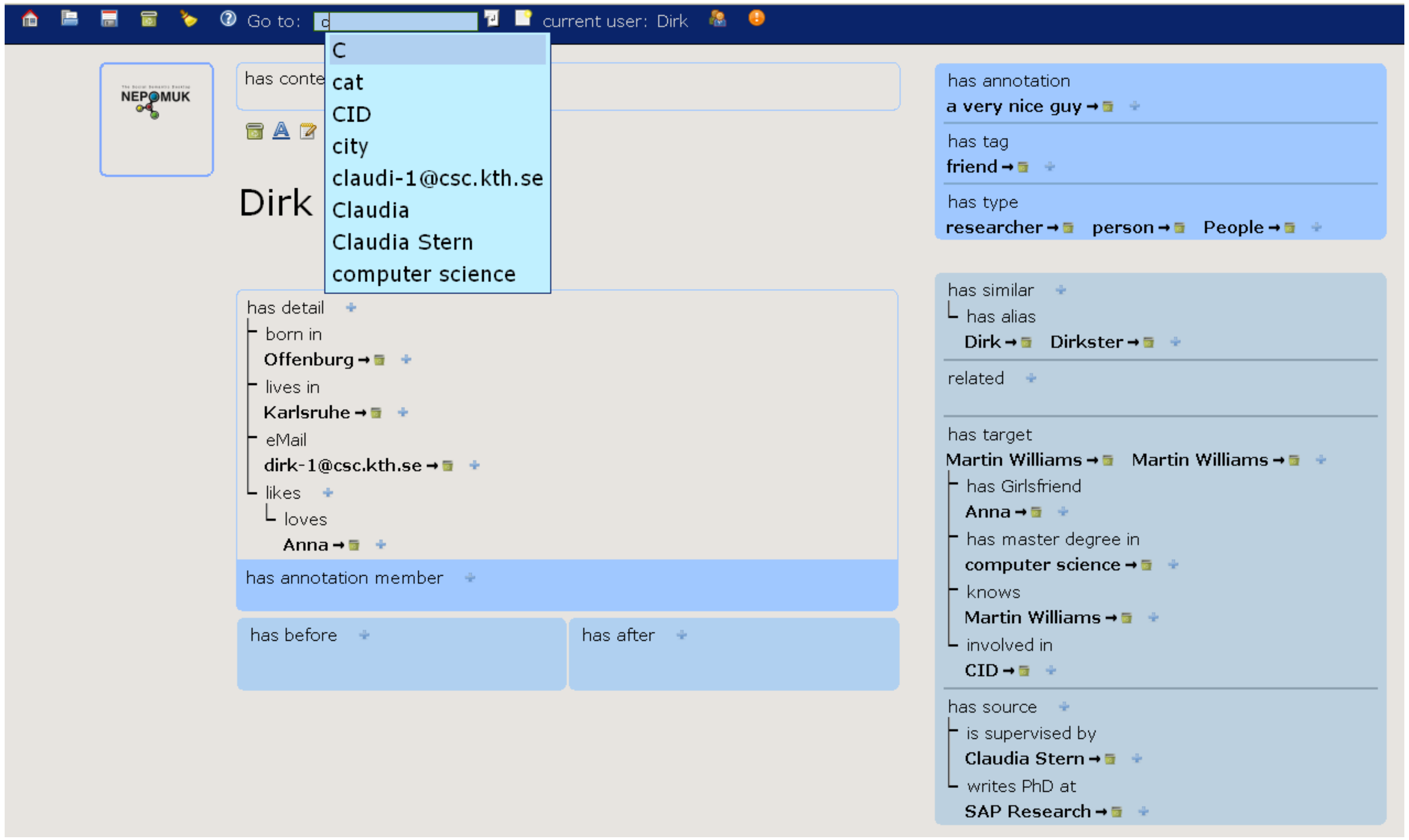
5 Related Work

A unified model for web content and semantic statements is presented in [13]. However, different from [13], the CDS model (i. e. *ContentItems*, *NameItems*, *Relations* and *Statements*) is specifically designed to be exposed to and understood by end-users. A model and system for a unified browsing and querying across document boundaries is presented in [7], but authoring is not considered. Systems similar to HKW include Artificial Memory [15] and Haystack [1]. Artificial Memory shares the idea of CDS to break documents up into small, interlinked parts to minimize redundancy and improve automated processing. But Artificial Memory does not allow the user to create non-structured sloppy entries. We believe letting the user decide how much effort to put into formalisation of a knowledge item is an important feature to keep the total cost of usage low. Haystack emphasizes rendering and linking of RDF-based entities, but lacks ways to author textual content intermingled with semantic facts.

6 Conclusions and Future Work

CDS lets the user express knowledge in the form of text (within an item), structure (structured text in items or structures between items) and formal statements (by using relations with defined semantics). In HKW, searches and navigation do not bring up long documents, but short fragments of text with its relations to other parts.

In the future, we will extend HKW to allow a user to convert a *ContentItem* structured with wiki syntax into a set of corresponding smaller *ContentItems*. This will lower the cost of creating *Items* even further. The reverse operation should also be possible: Merge a set of *Items* into a single *ContentItem*, as wiki syntax. This makes gradual formalisation of knowledge easier: First content is written into *ContentItems*, then these items are structured using wiki syntax, finally they are converted into many smaller *Items* that can further be annotated as needed.



Go to: c

current user: Dirk

C
cat
CID
city
claudi-1@csc.kth.se
Claudia
Claudia Stern
computer science

has conte

Dirk

has detail
born in
Offenburg
lives in
Karlsruhe
eMail
dirk-1@csc.kth.se
likes
loves
Anna

has annotation member

has before

has after

has annotation
a very nice guy

has tag
friend

has type
researcher **person** **People**

has similar
has alias
Dirk **Dirkster**

related

has target
Martin Williams **Martin Williams**
has Girlsfriend
Anna
has master degree in
computer science
knows
Martin Williams
involved in
CID

has source
is supervised by
Claudia Stern
writes PhD at
SAP Research

Fig. 4. HKW prototype screen shot, focusing on *Dirk Hageman*

Bibliography

[1] Adar, E., Karger, D. R. and Stein, L. A. [1999], Haystack: Per-user information environments, *in* 'CIKM', ACM, pp. 413–422.

[2] Avery, S., Brooks, R., Brown, J., Dorsey, P. and O'Conner, M. [2001], Personal knowledge management: Framework for integration and partnerships, *in* 'Proc. of ASCUE Conf.'.

[3] Barreau, D. and Nardi, B. [1995], 'Finding and reminding: File organization from the desktop', *SIGCHI Bulletin* **27**(3), 39–43.

[4] Carroll, J. J., Bizer, C., Hayes, P. and Stickler, P. [2004], Named graphs, provenance and trust, Technical report, HP.

[5] Decker, S., Park, J., Quan, D. and Sauermann, L., eds [2005], *The Semantic Desktop – Next Generation Information Management & Collaboration Infrastructure*, Galway, Ireland.

[6] Despres, C. and Chauvel, D. [2000], *Knowledge Horizons: the present and promise of Knowledge Management*, Butterworth-Heinemann.

[7] Dittrich, J.-P. and Salles, M. A. V. [2006], idm: a unified and versatile data model for personal dataspace management, *in* 'VLDB '06: Proceedings of the 32nd international conference on Very large data bases', VLDB Endowment, pp. 367–378.

[8] Drucker, P. F. [1985], *Management: Tasks, responsibilities, practices (Harper & Row management library)*, Harper & Row.

[9] Fielding, R. T. [2000], Architectural styles and the design of network-based software architectures, PhD thesis, University of California, Irvine.

[10] Frand, J. and Hixon, C. [1999], 'Personal knowledge management : Who, what, why, when, where, how?', Speech. working paper.
URL: *http://www.anderson.ucla.edu/faculty/jason.frand/researcher/speeches/PKM.htm*

[11] Hayes, P. [2004], RDF semantics, Recommendation, W3C.
URL: *http://www.w3.org/TR/rdf-mt/*

[12] Higgison, S. [2005], 'Your say: Personal knowledge management', *Insight Knowledge* **7**(7).

[13] Immaneni, T. and Thirunarayan, K. [2007], A unified approach to retrieving web documents and semantic web data, *in* E. Franconi, M. Kifer and W. May, eds, 'ESWC', Vol. 4519 of *Lecture Notes in Computer Science*, Springer, pp. 579–593.

[14] Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H. and Studer, R. [2007], 'Semantic wikipedia', *Journal of Web Semantics* . To appear.

[15] Ludwig, L. [2005], Semantic personal knowledge management, Technical Report D11.01_v0.01, DERI Galway.

[16] Luhmann, N. [1992], Kommunikation mit zettelkästen. ein erfahrungsbericht, *in* A. Kieserling, ed., 'Universität als Milieu', Kleine Schriften, Haux Verlag, Bielefeld, pp. 53–61. ISBN 3-925471-13-8.

[17] Maurer, H. [1999], The heart of the problem: Knowledge management and knowledge transfer, *in* 'Proc. ENABLE'99', Espoo-Vantaa Institute of Technology, pp. 8–17.

[18] Mitchell, A. [2005], 'The rise of personal km', *Inside Knowledge* **9**(1).

[19] Nonaka, I. and Konno, N. [1998], 'The concept of "ba": Building a foundation for knowledge creation', *California Management Review* **40**(3), 40–54.

[20] Nonaka, I. and Takeuchi, H. [1995], *The Knowledge-Creating Company : How Japanese Companies Create the Dynamics of Innovation*, Oxford University Press.

[21] Oren, E. [2005], SemperWiki: a semantic personal wiki, *in* [5].

[22] Oren, E., Delbru, R., Möller, K., Völkel, M. and Handschuh, S. [2006], Annotation and navigation in semantic wikis, *in* S. Schaffert and M. Völkel, eds, 'Proceedings of the First Workshop on Semantic Wikis - From Wiki to Semantics at the ESWC 2006'.

[23] Oren, E., Völkel, M., Breslin, J. G. and Decker, S. [2006], Semantic wikis for personal knowledge management, *in* 'Database and Expert Systems Applications', Vol. 4080/2006, Springer Berlin / Heidelberg, pp. 509–518.

[24] Polanyi, M. [1966], *Tacit Dimension*, Routledge & Kegan Paul Ltd, London.

[25] Völkel, M. [2007], A semantic web content model and repository, *in* 'Proceedings of the 3rd International Conference on Semantic Technologies'.
URL: *http://xam.de/2007/2007-05-voelkel-ISEMANTICS-swcm-CR.pdf*

[26] Völkel, M. and Haller, H. [2006], Conceptual data structures (cds) – towards an ontology for semi-formal articulation of personal knowledge, *in* 'Proc. of the 14th International Conference on Conceptual Structures 2006', Aalborg University - Denmark.

[27] Völkel, M., Haller, H. and Abecker, A. [2007], Modelling higher-level thought structures - method and tool, *in* 'Proceedings of Workshop on Foundations and Applications of the Social Semantic Desktop'.

[28] Völkel, M., Haller, H., Bolinder, W., Davis, B., Edlund, H., Groth, K., Gudjonsdottir, R., Kotelnikov, M., Lannerö, P., Lundquist, S., Sogrin, M., Sundblad, Y. and Westerlund, B. [2008], Conceptual data structure tools, Deliverable 1.2, nepomuk consortium.
URL: *http://nepomuk.semanticdesktop.org/xwiki/bin/download/IST/WebHome/D1.2_v10_CDS-Tools.pdf*

Mathematical Semantic Markup in a Wiki: the Roles of Symbols and Notations

Christoph Lange

Computer Science, Jacobs University Bremen
`ch.lange@jacobs-university.de`

Abstract. We present semantic markup as a way to exploit the semantics of mathematics in a wiki. Semantic markup makes mathematical knowledge machine-processable and thus allows for a multitude of useful applications. But as it is hard to read and write for humans, an editor needs to understand its inherent semantics and allow for a human-readable presentation. The semantic wiki SWiM offers this support for the OpenMath markup language. Using OpenMath as an example, we present a way of integrating a semantic markup language into a semantic wiki using a document ontology and extracting RDF triples from XML markup. As a benefit gained from making semantics explicit, we show how SWiM supports the collaborative editing of definitions of mathematical symbols and their visual appearance.

1 Making Mathematical Wikis More Semantic

What does a wiki need in order to support mathematics in a semantic way? First, there needs to be a way to edit mathematical formulæ. Many wikis offer a LaTeX-like syntax for that, and they have been used to build large mathematical knowledge collections, such as the mathematical sections of Wikipedia [30] or the mathematics-only encyclopædia PlanetMath [16]. But LaTeX, which is mostly presentation-oriented, despite certain macros like `\frac{num}{denom}` or `\binom{n}{k}`, is not sufficient to capture the *semantics* of mathematics. One could write $\mathcal{O}(n^2 + n)$, which could mean "$\mathcal{O}$ times $n^2 + n$" (with redundant brackets), or "$\mathcal{O}$ (being a function) applied to $n^2 + n$", or the set of all integer functions not growing faster than $n^2 + n$, and just by common notational convention we know that the latter is most likely to hold.

For being able to express the semantics of $\mathcal{O}(n^2+n)$, we need to make explicit that the Landau symbol $\mathcal{O}$ is a set construction operator and n is a variable. The meaning of $\mathcal{O}$ has to be defined in a vocabulary shared among mathematical applications such as our wiki. This is analogous to RDF, where a vocabulary—also called ontology—has to be defined before one can use it to create machine-processable and exchangeable RDF statements. In a mathematical context, these vocabularies are called *content dictionaries* (CDs). As with ontology languages, one can usually do more than just listing symbols and their descriptions in a CD: defining symbols formally in terms of other symbols, declaring their types formally, and specifying their visual appearance. Thus, CDs themselves are special mathematical documents that could again be made available in a mathematical wiki. Then it would be possible to create an unambiguous link from any occurrence of $\mathcal{O}$ in a formula to its definition in the wiki, and knowledge from the wiki could be shared with any other mathematical application supporting this CD. As a practical solution, we present the OpenMath CD language in sect. 2 and its integration into the semantic wiki SWiM in sect. 4.

2 Semantic Markup for Mathematics with OpenMath

Semantic markup languages for mathematics address the problems introduced in sect. 1 by offering an appropriate expressivity and semantics for defining symbols and other structures of mathematical knowledge. This is a common approach to knowledge representation not only in mathematics, but generally in science[1].

OpenMath [7] is a markup language for expressing the logical structure of mathematical formulæ. It provides its own sublanguage for defining CDs—collections of symbol definitions with formal and informal semantics. One symbol definition consists of a mandatory symbol name and a normative textual description of the symbol, as well as other metadata[2]. Formal mathematical properties (FMPs) of the symbol, such as the definition of the sine function, or the commutativity axiom that holds for the multiplication operator, can be added, written in OpenMath and possibly using other symbols (see fig. 1). Type signatures (such as $\sin\colon \mathbb{R} \to \mathbb{R}$) and human-readable notations (see sect. 3) of symbols are defined separately from the CD in a similar fashion.

As semantic markup makes mathematical formulæ machine-understandable, it has leveraged many applications. For OpenMath, it started with data exchange between computer algebra systems, then automated theorem provers, and more recently dynamic geometry systems. OpenMath is also used in multilingual publishing, adaptive learning applications, and web search [10]. OpenMath CDs foster exchange by their modularity. Usually, a CD contains a set of related symbols, e. g. basic operations on matrices (CD `linalg1`) or eigenvalues and related concepts (CD `linalg4`), and a CD group contains a set of related CDs, e. g. all standard CDs about linear algebra (CD group `linalg`). In this setting, agents exchanging mathematical knowledge need not agree upon one large, monolithic mathematical ontology, but can flexibly refer to a specific set of CDs or CD groups they understand[3].

[1] Consider e. g. the chemical markup language CML [23]

[2] OpenMath 2 uses an idiosyncratic schema for metadata, but Dublin Core is likely to be adopted for OpenMath 3.

[3] A communication protocol for such agreements is specified in [7, sect. 5.3].

```
<CDDefinition>
<Name>sin</Name>
<Description>The sine function on real numbers</Description>
<CMP>The sine function is defined in terms of the exponential function as
 sin x = ½(e^{ix} − e^{−ix}).</CMP>
<FMP>
  <OMOBJ xmlns="http://www.openmath.org/OpenMath"
   version="2.0" cdbase="http://www.openmath.org/cd">
    <OMA><OMS cd="relation1" name="eq"/>
      <OMA><OMS name="sin" cd="transc1"/>
        <OMV name="x"/></OMA>
      <OMA><OMS name="divide" cd="arith1"/>
        <OMA><OMS name="minus" cd="arith1"/>
          <OMA><OMS name="exp" cd="transc1"/>
            <OMA><OMS name="times" cd="arith1"/>
              <OMS name="i" cd="nums1"/>
              <OMV name="x"/></OMA></OMA>
          <OMA><OMS name="exp" cd="transc1"/>
            <OMA><OMS name="times" cd="arith1"/>
              <OMA><OMS name="unary_minus" cd="arith1"/>
                <OMS name="i" cd="nums1"/></OMA>
              <OMV name="x"/></OMA></OMA></OMA>
        <OMA><OMS name="times" cd="arith1"/>
          <OMI>2</OMI>
          <OMS name="i" cd="nums1"/></OMA></OMA></OMA></OMOBJ></FMP>
</CDDefinition>
```

Fig. 1. A definition of the sine function in OpenMath. `FMP` stands for "formal mathematical property", whereas the "C" in `CMP` stands for "comment". `OMA` is an application of a symbol (`OMS`) to some arguments, `OMV` denotes a variable, and `OMI` denotes an integer. For any symbol, its CD (resolved relatively to a base URI) and name have to be given. The CD used here are standard ones officially approved by the OpenMath Society.

3 Authoring, Navigating and Presenting Semantic Markup

As semantic markup is obviously hard to read and write for humans (see fig. 1), proper authoring software is desirable for writing it. For reading, it should be transformed to presentation markup. LaTeX or presentational MathML [2] are suitable output formats. With "authoring", we refer both to authoring instances, i. e. formulæ that can use symbols from any CD, but also to authoring ontologies, i. e. CDs. In fact, as with editors for semantic web ontologies, a sophisticated editor should cover both levels, as existing vocabulary (here: mathematical symbols) is not just used to create instance data, but also to define new vocabulary in terms of old one.

Semantic markup is commonly presented by defining the *notation* of every symbol as a mapping from a single semantic symbol—or a pattern matching a set of semantic markup structures in which the symbol can occur—to fragments of presentation markup, where arguments to symbols are presented by recursive application of the rules [14, 19]. One such pattern-based language for notation definitions has been proposed as a part of the CD language of the MathML 3 standard (see sect. 5) and, within the current process of aligning both languages, is likely to be adopted for OpenMath 3 as well. For XML languages, semantics-to-presentation-mappings are commonly given in XSLT [12], either directly, or generated from a more concise representation [14, 19], but there are also non-XSLT implementations (see sect. 5).

In the course of opening up new mathematical areas, definitions of new symbols and their axiomatization are not fixed initially but subject to continuous evolution and refactoring—a workflow that a semantic wiki should support. In this paper, we assume that the semantics of symbols is fixed, but then it is still the *notation* of the symbol that can evolve. On the one hand there is evolution in the course of time. Before the 16th century, a prefixed letter R or r (= radix) had mainly been used for square roots, but then the more abstract symbol $\sqrt{\ }$ was developed [8]. On the other hand the notation of a symbol depends on the *context* the symbol is used in. The context can be determined by the language of the author or the reader, by previous knowledge, by the area of application, and other criteria [14, 27]. For example, a binomial coefficient is written as $\binom{n}{k}$ in German or English, but as $\mathcal{C}_n^k$ in French. A mathematician uses the symbol i for the imaginary unit, whereas an electrical engineer would write j. In a strict style, one would express asymptotic growth as $f \in \mathcal{O}(g)$, but the sloppy style of $f = \mathcal{O}(g)$ is far more common. For applications this means that reusing existing mathematical content in a new context requires adapting the notation [14].

The fact that there is no single, definitive notation for a mathematical symbol leads to the requirement that an integrated editor for mathematical documents and CDs should instantly reflect changes of notation definitions in the places where they are used for presentation. That means that whenever the notation of a symbol σ has changed, all the presentation markup generated from formulæ containing σ has to be invalidated and re-rendered upon the next request. In a single-author environment this frees the author from recompiling all the affected presentation markup and gives instant visual feedback about whether the new notation works, and in a collaborative environment it relieves *other* authors from worrying whether they are looking at an up-to-date presentation of a document.

Moreover, an authoring tool should make the semantic relations between definitions of symbols, their notations, and their type signatures, as well as the relations between instances (here: symbols used in formulæ) and classes explicit in the user interface, as this facilitates orientation both for authors and for readers.

4 OpenMath CDs in the Semantic Wiki SWiM

SWiM aims at satisfying the above requirements for authoring, navigating, and presenting mathematical knowledge in a semantic wiki [18]. It is based on the general-purpose semantic wiki IkeWiki [26] and enhances it by support for OpenMath and subsets of MathML and the more comprehensive OMDoc [13]. In this paper, we focus on OpenMath.

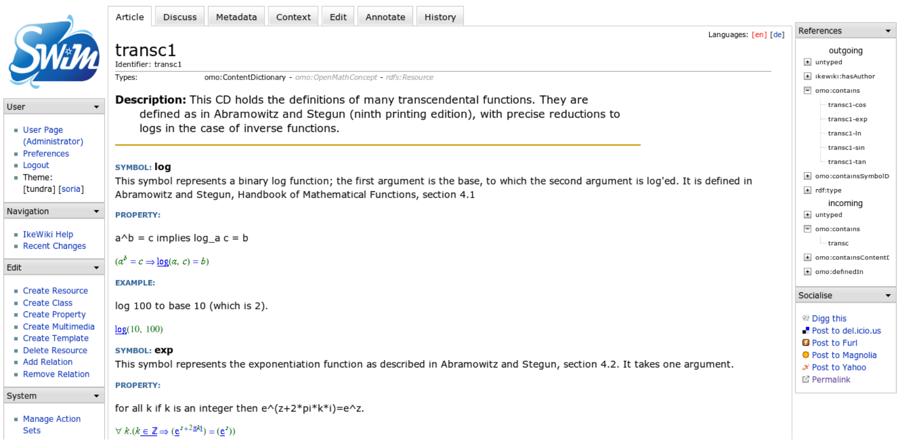
SWiM follows the approach of representing one subject of interest by one wiki page and modeling it as one resource in an RDF graph; this shall henceforth be called "page-level annotation". This way of knowledge representation is the most common in semantic wikis, but nowhere near the only one [24, 6]. Our experience with developing SWiM and working in other semantic wikis following the same approach, such as Semantic MediaWiki [29], shows that this works best if pages are small. Otherwise, too many subjects of interest would be described in subsections of pages and thus could not be represented in RDF. But small pages are advisable in a wiki anyway, as they reduce the potential of editing conflicts. With small pages, page-level annotation is at its most impressive in terms of easy-to-use authoring, navigation, and search from a user's point of view, and easy maintenance within the system[4].

An important task in implementing the OpenMath support in SWiM was the choice of the granularity of pages. The OpenMath Society, the body standardizing OpenMath, considers whole CDs as units that are subject to review or change [7, sect. 4.5], and the OpenMath CD language reflects that by only providing "date" and "author" metadata fields on CD level. But quite a lot of CDs contain more than 10 symbol definitions, covering several screen pages in a browser. With SWiM we are not only aiming at supporting the review process of the OpenMath Society but also the more dynamic preceding phase, when, for example, a working group in some company or research institute is developing a new CD that will then be submitted to the OpenMath Society for approval. Thus, both to keep wiki pages small, and to support a more dynamic workflow, we chose to map symbol definitions and even their subelements (formal and informal properties, and examples) to wiki pages. The latter choice is influenced

[4] These aspects were discussed in more detail on the mailing list of the Semantic Wiki Interest Group [28] in two threads named "Modeling 'third party' relations on Semantic Mediawiki page?" and "Creating Triples Anywhere in a Semantic Wiki" in April 2007.

from our experience with OMDoc [13], where axioms and theorems—defining or asserting additional properties of symbols—and examples are modeled as entities separate from the declarations or definitions of the symbols, allowing for greater flexibility.

Now that this design choice has been found reasonable for SWiM, it should not violate the compatibility with other OpenMath applications. SWiM allows for importing and exporting OpenMath files from and to the local file system or the OpenMath Subversion repository. On import, CDs are split into their subparts as mentioned above, every part being stored on its own wiki page. The containment relations are preserved as XIncludes [21], which are resolved on export again. The XIncludes are also resolved when a document is rendered, so that a whole CD can be viewed at once.



transc1
Identifier: transc1
Types: omo:ContentDictionary - omo:OpenMathConcept - rdfs:Resource

Description: This CD holds the definitions of many transcendental functions. They are defined as in Abramowitz and Stegun (ninth printing edition), with precise reductions to logs in the case of inverse functions.

SYMBOL: **log**
This symbol represents a binary log function; the first argument is the base, to which the second argument is log'ed. It is defined in Abramowitz and Stegun, Handbook of Mathematical Functions, section 4.1

PROPERTY:

a^b = c implies log_a c = b

$(a^b = c \Rightarrow \log(a, c) = b)$

EXAMPLE:

log 100 to base 10 (which is 2).

$\log(10, 100)$

SYMBOL: **exp**
This symbol represents the exponentiation function as described in Abramowitz and Stegun, section 4.2. It takes one argument.

PROPERTY:

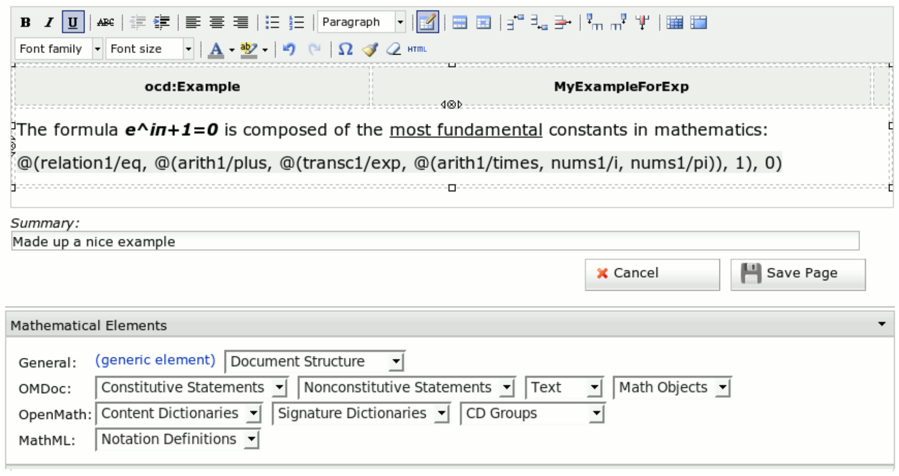
for all k if k is an integer then e^(z+2*pi*k*i)=e^z.

$\forall\, k.(k \in \mathbb{Z} \Rightarrow (e^{z+2\pi k i}) = (e^{z}))$

Fig. 2. A content dictionary in SWiM

SWiM supports editing OpenMath and other semantic markup in a semi-WYSIWYG way. Plain text can be edited and styled visually (but styles are lost on export), for OpenMath objects there is a simple linear ASCII syntax, and other XML structures are made accessible as special HTML tables (see fig. 3).

The semantics of the OpenMath CD language is documented in a human-readable specification [7, chap. 4], which is not explicit enough to make OpenMath CDs directly usable by semantic web applications. Instead, an ontology had to be developed to allow for making RDF statements about resources in CDs, and an automated extraction of RDF statements from the OpenMath markup had to be developed. The OpenMath document ontology models classes and properties for all structural entities found in OpenMath's CD groups, CDs, type signatures, and notation definitions in OWL-DL. Properties from common on-



The formula ***e^in+1=0*** is composed of the most fundamental constants in mathematics:

@(relation1/eq, @(arith1/plus, @(transc1/exp, @(arith1/times, nums1/i, nums1/pi)), 1), 0)

Made up a nice example

Fig. 3. Editing OpenMath in SWiM

tologies like Dublin Core were reused where appropriate. The inner structures of formulæ (also known as "OpenMath objects") were not modeled, for two reasons: First, formulæ are not accessible as entities of their own in SWiM but as children of structural entities like FMPs, which are represented in the document ontology, and secondly, RDF combined with logics used for semantic web reasoning is not expressive enough for capturing the full semantics of mathematical formulæ. The latter should instead be left to a theorem prover or computer algebra system working with OpenMath objects. If we had an ontology for expressing the *syntactic* structure of an OpenMath object like $\forall x.x \in \mathbb{R} \Rightarrow e^x > 0$ in RDF, as discussed in [20], we would be able to make the references to the bound variable x explicit (they would point to the same URI), but still we would not be able to express the notion of α-equivalence (meaning that we could have used any other name for x as well) in the first order logic subsets commonly used for reasoning on the semantic web. There is, however, a property that states that an FMP or an example uses a symbol—which is contained in some OpenMath object inside the FMP or example—and points to the definition of that symbol in some CD. This is not only useful for determining dependencies among CDs (What other CDs do I need to load in order to get a self-contained collection?), but also for rendering formulæ according to notation definitions (see sect. 5).

Whenever a wiki page is stored in SWiM, i. e. whenever it is saved in the editor or imported, an RDF representation in terms of the document ontology is extracted from the markup. Consider the following CD:

```
<CD xml:id="sample">
  <CDName>sample</CDName>
  <CDDate>2008-03-05</CDDate>
  <CDStatus>private</CDStatus>
  <Description>A sample CD</Description>
  <xi:include href="url/of/CDDefinition"/></CD>
```

From this, the following RDF would be extracted (in Turtle syntax [4]), where `omo` is the prefix of the OpenMath document ontology namespace:

```
<#sample>
  rdf:type omo:ContentDictionary ;
  dc:identifier "sample" ;
  dc:date 2008-03-05 ;
  omo:status "private" ;
  dc:description "A sample CD" ;
  omo:containsSymbolDefinition <url/of/cddefinition> .
```

The extracted RDF is stored in the Jena store built into IkeWiki, where the OpenMath document ontology resides as well. IkeWiki uses Jena's builtin RDFS reasoner that implements the RDFS semantics but understands the OWL syntax as well [26]. IkeWiki currently utilizes the RDF graph in order to generate a list of incoming and outgoing links for the current page, grouped by property (shown on the right side in fig. 2), to feed a graphical RDF browser, to preselect properties of pages and links an author would probably want to annotate, and it supports embedding arbitrary inline SPARQL queries [25] into pages.

5 Defining Notations and (Re-)Rendering Formulæ

For defining the notation of a symbol, SWiM employs the pattern-based language proposed for MathML 3 [2, sect. 8.6, "Rendering of Content Elements"]. A notation definition for a symbol consists of a *prototype* that matches a fragment of semantic markup, either matching elements literally or matching subtrees against so-called content metavariables, and a *rendering*, which is a template of presentation markup with so-called element metavariables in those places where the results of rendering the XML trees matched against the correspondent content metavariables are to be inserted. This is, for example, a notation definition for the root operator in the *arith1* CD, specifying the rendering $\sqrt[n]{\mathrm{arg}}$:

```
<notation xml:id="ntn-root">
  <prototype>
    <om:OMA>
      <om:OMS cd="arith1" name="root"/>
      <expr name="arg"/>
      <expr name="n"/></om:OMA></prototype>
  <rendering>
    <m:mroot>
      <render name="arg"/>
      <render name="n"/>
    </m:mroot></rendering></notation>
```

From this, the RDF triple `<#ntn-root> omo:rendersSymbol <url/of/arith1/root> .` would be extracted.

SWiM employs the Java-based mmlproc rendering library [14] for rendering OpenMath objects to presentational (i. e. non-semantic) MathML, which can be viewed in recent versions of the Firefox or Opera browsers. Whenever a wiki page containing notation definitions is saved or imported, the notation definitions are put into a cache read by mmlproc. To symbols without a notation definition, mmlproc applies a default rendering like `root(arg,n)`.

If a notation definition has been added, deleted, or changed, the affected documents have to be re-rendered. In order to do this properly, SWiM has to

1. identify changes to notation definitions
2. identify documents affected by a change

(1) is done by computing an XML diff between the cached and the newly inserted version of a notation definition. (2) is done by querying the RDF graph for all FMPs and examples using the symbol rendered by the respective notation definition, as shown in fig. 4 and 5. Not only the wiki pages holding these FMPs and examples have to be re-rendered, but also those pages (symbol definitions and CDs) that directly or indirectly *include* these fragments.

```
SELECT DISTINCT ?page WHERE {
  <changed-ntn-def> omo:rendersSymbol ?sym .
  { ?page omo:usesSymbol ?sym } UNION
  { ?exOrFMP omo:usesSymbol ?sym .
    { ?page omo:contains ?exOrFMP } UNION
    { ?page omo:contains ?symDef .
      ?symDef omo:contains ?exOrFMP } } }
```

Fig. 4. SPARQL query determining the effect of changing a notation definition; see fig. 5 for a graphical representation.

6 Related Work

SWiM was originally motivated by deficiencies in **mathematical wikis** like Wikipedia or PlanetMath (see sect. 1). For MediaWiki, a semantic web extension has been developed [29], which aims at being used in the MediaWiki-powered Wikipedia. se(ma)^{2}wi [31] was a Wikipedia-independent experiment with a Semantic MediaWiki fed with OMDoc-formatted mathematical knowledge from the ActiveMath learning environment. While the ActiveMath learning metadata are displayed in the wiki, most of the structural semantics explicitly given in OMDoc is, however, lost during this import: The formulæ are converted to

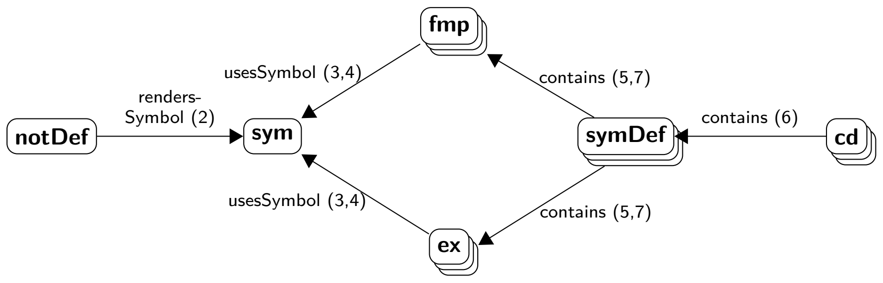


Fig. 5. Finding pages (depicted as stacks of nodes) affected by changes to a notation definition. Numbers refer to lines of listing 4. Note that both *sym* and the *symDef*s are instances of the class *SymbolDefinition*.

presentational-only LaTeX, and the links between wiki pages that represent mathematical statements, for example a link from a theorem to its proof, are not typed and therefore cannot be queried.

In its role as a CD editor, SWiM is comparable to an **ontology editor**. Actually, the IkeWiki base system provides a simple editor for RDFS and OWL ontologies as well. OntoWiki is a more comprehensive, agile collaborative editor for ontologies and instance data that is inspired by wiki principles like ease of use for non-experts and versioning [1]. In OntoWiki, one edits a resource like a database record, namely in a table containing edit boxes for all properties of the resource, whereas a resource in a semantic wiki is represented as a semi-structured document, which the user can enrich with annotations. While an OpenMath CD has a mostly record-like structure, this does not hold for mathematical documents in general; consider e. g. the OpenMath objects inside a CD, or a mathematical lecture written in the more versatile OMDoc language [13].

While SWiM is the first wiki that supports editing **notation definitions**, this has been investigated for text editors before. In the PLATΩ system, the TeXMACS editor has been extended towards semantic markup [3]. The developers focus on notations that use natural language and on parsing text and formulæ the user writes in a presentational style back to a semantic representation. Both features have not yet been investigated in SWiM; here the focus is rather on making the semantic markup editable in a convenient way. As a change to a notation definition in PLATΩ/TeXMACS involves regenerating parser rules, special attention is paid to making this efficient by only regenerating those rules that are affected by a change. Improving this by computing minimal diffs w. r. t. extended equivalence relations for structured document formats (such as ignoring changes to whitespace) and computing and previewing long-range effects of changes is further elaborated in [22].

Formula rendering is a special case of inline query processing. Many semantic wikis support inline queries as a means of automatically generating lists [15]; usually an inline query consists of a predicate $p\colon \text{Page} \to \mathbb{B}$, a specification of the information that is desired for every page satisfying p (e. g. its title), and a formatting style for the result. An OpenMath object in SWiM can be considered a query for the notation definitions of the symbols used in the object, where for every symbol only the most appropriate rendering[5] is included in the result set and the result is "formatted" by rendering the symbols according to the rendering specifications in the result set. In this setting, we can determine whether a change to a page (here: a notation definition) affects the result set of a query (here: a formula) by checking whether the formula contains a particular fixed symbol, which requires linear time w. r. t. the size of the formula. For general queries[6], this is far more complex, as the satisfiability problem for propositional boolean expressions is $\mathcal{NP}$-complete.

The **document ontology** presented here is the first one that has been developed for OpenMath CDs. In previous work, we have introduced a similar ontology for a subset of OMDoc [17]. A different ontology for OpenMath has been developed in the MONET project. It does not model the *document structure* of CDs but can be used to relate OpenMath objects to certain web services operating on them; e. g. one can specify that there is a web service for computing definite integrals, which can operate on any object that applies the *defint* symbol from the *calculus1* CD to certain arguments [9].

7 Conclusion and Outlook

Having motivated that mathematical semantic markup languages can help to make semantic wikis aware of mathematics, we showed how the OpenMath content dictionary language was integrated into the semantic wiki SWiM by choosing an appropriate page granularity, modeling a document ontology, and extracting relevant facts from the markup into RDF. We motivated the need for supporting the maintenance of notation definitions for mathematical symbols and showed how to utilize the information from the RDF graph in order to improve the performance of the system and the usability in terms of navigation and orientation when editing notation definitions.

So far, SWiM assumes that there is at most one notation definition per symbol. The mmlproc renderer supports callbacks to an algorithm that selects the most appropriate out of a set of multiple possible renderings for a symbol. A default implementation considering the static context of a formula (such as the language of the document [section]) is provided with mmlproc [14]. In future, it is planned to provide a user interface inside SWiM that lets the user select his preferred rendering for every symbol.

A visual editor for formulæ will be provided as well. Available editors will be evaluated w. r. t. their extensibility by new symbols and notation definitions. Ideally, the tool palette of a visual formula editor would be supplied dynamically with all known instances of the *SymbolDefinition* class.

[5] See sect. 7 for dealing with multiple renderings per symbol.
[6] In database research, the area of problems touched on here is known as "materialized view maintenance" [11].

Additional reasoning tasks need to be investigated to allow for more powerful queries. For example, the dependency relation between CDs (see sect. 4) and the containment relation between CDs and their subparts (see sect. 5) are transitive, but RDFS cannot express transitivity, and SPARQL cannot compute transitive closures.

Finally, the relationship between the structural semantics of documents and domain knowledge is worth investigating. If we define the Landau symbols $\mathcal{O}$ and Ω in a CD, probably including their type declarations, we have not gained more domain knowledge than that two mathematical concepts $\mathcal{O}$ and Ω exist that map integer functions to sets of integer functions. In the more expressive OMDoc language we could provide a definition of Ω as $\forall f, g. f \in \Omega(g) :\Leftrightarrow g \in \mathcal{O}(f)$ and use that knowledge e. g. to customize presentation: Formulæ using Ω could be rewritten to their equivalents using more the familiar $\mathcal{O}$. Relationships between mathematical concepts are not only given by definitions: Commonly a differentiable function f is defined as a function that has a derivative, but the fact that f also is continuous is only observed afterwards as a theorem. From such definitions or theorems one could extract a domain ontology use it for reasoning. In DL this might look as follows (class names abbreviated) [5]:

$$\mathrm{ContFunc} \sqsubseteq \mathrm{Func} \tag{1}$$

$$\mathrm{DiffableFunc} = \mathrm{Func} \sqcap \exists \mathrm{hasDeriv}.\mathrm{Func} \tag{2}$$

$$\mathrm{DiffableFunc} \sqsubseteq \mathrm{ContFunc} \tag{3}$$

This information could then be utilized e. g. to display a general theorem about continuous functions when the user searches for a theorem about differentiable functions.

Acknowledgments The author would like to thank Michael Kohlhase for inspiration by his vision of a semantic web for mathematics, and James Davenport, representing the OpenMath society, for sharing his insights about CD metadata.

References

1. S. Auer, S. Dietzold, and T. Riechert. Ontowiki – A tool for social, semantic collaboration. In I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *5th International Semantic Web Conference*, volume 4273 of *LNCS*. Springer, 2006.
2. R. Ausbrooks, B. Bos, O. Caprotti, D. Carlisle, G. Chavchanidze, A. Coorg, S. Dalmas, S. Devitt, S. Dooley, M. Hinchcliffe, P. Ion, M. Kohlhase, A. Lazrek, D. Leas, P. Libbrecht, M. Mavrikis, B. Miller, R. Miner, M. Sargent, K. Siegrist, N. Soiffer, S. Watt, and M. Zergaoui. Mathematical Markup Language (MathML) version 3.0. W3C working draft, W3C, 2007. `http://www.w3.org/TR/MathML3`.
3. S. Autexier, A. Fiedler, T. Neumann, and M. Wagner. Supporting user-defined notations when integrating scientific text-editors with proof assistance systems. In M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors, *Towards Mechanized*

Mathematical Assistants. MKM/Calculemus, number 4573 in LNAI. Springer, 2007.

4. D. Beckett. Turtle – terse RDF triple language, 2007. http://www.dajobe.org/2004/01/turtle/.
5. M. Bröcheler. The mathematical semantic web. Bachelor's thesis, Computer Science, Jacobs University, Bremen, 2007.
6. M. Buffa, F. Gandon, G. Ereteo, P. Sander, and C. Faron. Sweetwiki: A semantic wiki. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2008. in press.
7. S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaetano, and M. Kohlhase. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004. http://www.openmath.org/standard/om20.
8. F. Cajori. *A History of Mathematical Notations.* Courier Dover Publications, 1993. Originally published in 1929.
9. O. Caprotti, M. Dewar, and D. Turi. Mathematical service matching using description logic and OWL. In A. Asperti, G. Bancerek, and A. Trybulec, editors, *Mathematical Knowledge Management*, number 3119 in LNAI. Springer, 2004.
10. J. H. Davenport. OpenMath in a (semantic) web. In O. Caprotti, S. Xambó, M.-A. Huertas, M. Kohlhase, and M. Seppälä, editors, *3rd JEM Workshop – Joining International Mathematics*, 2008. http://jem-thematic.net/workshop3.
11. A. Gupta and I. S. Mumick, editors. *Materialized views: techniques, implementations, and applications.* MIT Press, Cambridge, MA, USA, 1999.
12. M. Kay. XSL Transformations (XSLT) Version 2.0. W3C Recommendation, W3C, 2007. http://www.w3.org/TR/2007/REC-xslt20-20070123/.
13. M. Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2].* Number 4180 in LNAI. Springer, 2006.
14. M. Kohlhase, C. Müller, and F. Rabe. Notations for living mathematical documents. In *Mathematical Knowledge Management, MKM'08*, LNAI. Springer Verlag, 2008. in press.
15. M. Krötzsch, S. Schaffert, and D. Vrandečić. Reasoning in semantic wikis. In G. Antoniou, U. Aßmann, C. Baroglio, S. Decker, N. Henze, P.-L. Pătrânjan, and R. Tolksdorf, editors, *3rd Reasoning Web Summer School*, volume 4636 of *LNCS*. Springer, 2007.
16. A. Krowne. An architecture for collaborative math and science digital libraries. Master's thesis, Virginia Tech, 2003. http://scholar.lib.vt.edu/theses/available/etd-09022003-150851/.
17. C. Lange. SWiM – a semantic wiki for mathematical knowledge management. Technical Report 5, Jacobs University Bremen, 2007. http://kwarc.info/projects/swim/pubs/tr-swim.pdf.
18. C. Lange. SWiM: A semantic wiki for mathematical knowledge management. http://kwarc.info/projects/swim/, 2008.
19. S. Manzoor, P. Libbrecht, C. Ullrich, and E. Melis. Authoring Presentation for OpenMath. In M. Kohlhase, editor, *Mathematical Knowledge Management*, number 3863 in LNAI. Springer, 2005.
20. M. Marchiori. The mathematical semantic web. In A. Asperti, B. Buchberger, and J. H. Davenport, editors, *Mathematical Knowledge Management*, number 2594 in LNCS. Springer, 2003.
21. J. Marsh, D. Orchard, and D. Veillard. XML inclusions (XInclude) version 1.0 (second edition). W3C Recommendation, World Wide Web Consortium (W3C), Nov. 2006. Available at http://www.w3.org/TR/2006/REC-xinclude-20061115/.

22. N. Müller and M. Wagner. Towards Improving Interactive Mathematical Authoring by Ontology-driven Management of Change. In A. Hinneburg, editor, *LWA (Lernen, Wissensentdeckung und Adaptivität)*, 2007.
23. P. Murray-Rust, H. S. Rzepa, and M. Wright. Development of chemical markup language (cml) as a system for handling complex chemical content. *New Journal of Chemistry Articles*, 25:618–634, 2001.
24. E. Oren, R. Delbru, K. Möller, M. Völkel, and S. Handschuh. Annotation and navigation in semantic wikis. In M. Völkel, S. Schaffert, and S. Decker, editors, *1st Workshop on Semantic Wikis*, volume 206 of *CEUR Workshop Proceedings*, 2006.
25. E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C Recommendation, W3C, 2006. `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`.
26. S. Schaffert. IkeWiki: A semantic wiki for collaborative knowledge management. In *1st International Workshop on Semantic Technologies in Collaborative Applications (STICA)*, 2006.
27. E. Smirnova and S. M. Watt. Notation Selection in Mathematical Computing Environments. In *Proc. Transgressive Computing (TC) 2006*, 2006.
28. Mailing list of the semantic wiki interest group. swikig@aifb.uni-karlsruhe.de, `http://www.aifb.uni-karlsruhe.de/mailman/listinfo/swikig`.
29. M. Völkel, M. Krötzsch, D. Vrandečić, H. Haller, and R. Studer. Semantic Wikipedia. In *15^{th} WWW conference*, 2006.
30. Wikipedia, the free encyclopedia. `http://www.wikipedia.org`, 2001–2007.
31. C. Zinn. Bootstrapping a semantic wiki application for learning mathematics. In Y. Sure and S. Schaffert, editors, *Semantics: From Visions to Applications*, 2006.

Poster | Edit | Discussion | History

SWiM – A Semantic Wiki for Mathematical Knowledge Management

SWiM is a semantic wiki for collaboratively building, editing, browsing, and discussing collections of mathematical knowledge represented in structural semantic markup. It motivates users to contribute by instantly sharing the benefits of knowledge-powered services with them.

Mathematical Knowledge Management

Goal: web collaboration on structured mathematical knowledge

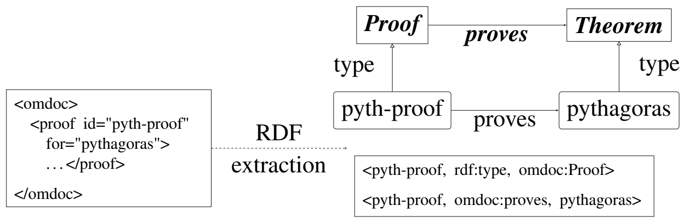
- semantic markup common for documents in mathematics: MathML, OpenMath, OMDoc, sTEX
- layers of knowledge: symbols, statements, theories, documents
- applications: e-learning, publishing, proof verification
- but how to acquire the knowledge?
- ⇒ services to motivate the user and support the authoring workflow

```
<apply>
  <csymbol definitionURL=
    "http://openmath.org/cd/
    arith1#plus"/>
  <cn type="integer">1</cn>
  <ci>n</ci>
</apply>
```

A “simple” semantic formula

Semantic Wiki and Ontologies

- Semantic wikis found usable to support collaborative formalisation
- Difference here is: deeply nested markup, lots of cross-references
- Right granularity of pages: one page = one theory, one statement, one formula?
- ⇒ extract knowledge relevant for search and navigation, build most services on top of that
- RDF graph in terms of an ontology that models the semantics of the markup; direct and inferred relationships: dependency, containment


Proof
proves
Theorem
type
type
pyth-proof
proves
pythagoras
<omdoc>
<proof id="pyth-proof" for="pythagoras">
…</proof>
</omdoc>
RDF
extraction
<pyth-proof, rdf:type, omdoc:Proof>
<pyth-proof, omdoc:proves, pythagoras>

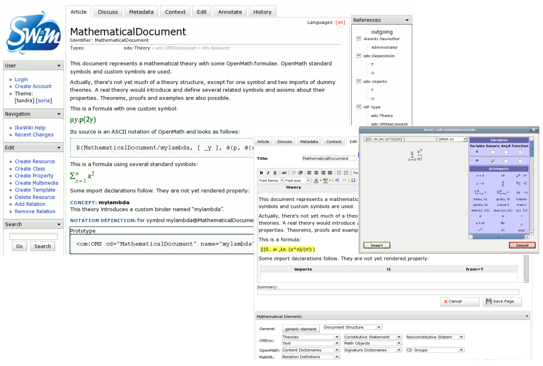
SWiM = IkeWiki + OMDoc + $\Sigma_{i=1}^{\infty}\sigma_i$, $\sigma_i \in$ service

Technical foundations:

- IkeWiki [Schaffert]
- OMDoc [Kohlhase]

Features:

- editing
- presentation
- navigation
- discourse
- import/export
- refactoring
- semantic services



MathematicalDocument
Identifier: MathematicalDocument
Types: odo:Theory - odo:OMDocConcept - rdfs:Resource
Languages: [en]

This document represents a mathematical theory with some OpenMath formulae. OpenMath standard symbols and custom symbols are used.

Actually, there's not yet much of a theory structure, except for one symbol and two imports of dummy theories. A real theory would introduce and define several related symbols and axioms about their properties. Theorems, proofs and examples are also possible.

This is a formula with one custom symbol:

$\mu y.p(2y)$

Its source is an ASCII notation of OpenMath and looks as follows:

```
B(MathematicalDocument/mylambda, [ _y ], @(p, @(a
```

This is a formula using several standard symbols:

$\sum_{x=1}^{n} x^2$

Some import declarations follow. They are not yet rendered properly:

CONCEPT: **mylambda**
This theory introduces a custom binder named “mylambda”.

NOTATION DEFINITION: for symbol mylambda@MathematicalDocume

Prototype

```
<om:OMS cd="MathematicalDocument" name="mylambda"
```

References: outgoing — ikewiki:hasAuthor: Administrator; odo:dependsOn: T, U; odo:imports: T, U; rdf:type: odo:Theory, odo:OMDocConcept

Title: MathematicalDocument

This is a formula:

Σ(0.. ∞ ,λn. (x^n)/(n!))

imports | i1 | from=T

Summary:

Mathematical Elements — General: generic element, Document Structure; OMDoc: Theories, Text, Constitutive Statement, Math Objects, Nonconstitutive Statem; OpenMath: Content Dictionaries, Signature Dictionaries, CD Groups; MathML: Notation Definitions

OpenMath 3 Case Study

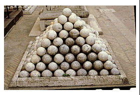
- revision of the content dictionaries (collections of symbol definitions)
- user interface: editing formulae, metadata, symbol notations

Examples for notational preferences:

- language: $\binom{n}{k}$, $\mathcal{C}_n^k$, $\mathcal{C}_k^n$
- domain: $\sqrt{-1} = i$, $\sqrt{-1} = j$
- taste: $f''(x)$, $f^{(2)}(x)$, $\frac{d^2y}{dx^2}$, $\frac{d^2}{dx^2}f(x)$
- exactness: $f \in \mathcal{O}(n)$, $f = \mathcal{O}(n)$

Flyspeck Case Study

- **F**ormalising a **P**roof of the **K**epler conjecture [Hales 1998]
- hundreds of proof sketches (400 pages LaTeX), collaboratively transform them into something machine-verifiable
- formalising, annotating, discussing, project management



User

Christoph Lange
`<ch.lange@jacobs-university.de>`

School of Engineering and Science
Jacobs University Bremen, Germany

JACOBS UNIVERSITY

Digital Enterprise Research Institute
NUI Galway, Ireland

DERI

Search

$\int_{?}^{?} s^2(t)dt$

Go!

Import

- OpenMath CD
- OMDoc
- Ontology
- LaTeX

Export

- OpenMath CD
- OMDoc
- XHTML+MathML
- RDF
- PDF

References

- `hasDemo`
  `http://swim.kwarc.info`
- `homepageURL`
  `http://kwarc.info/projects/swim/`
- `presentedAt`
  Semantic Wiki Workshop
  JEM Workshop (Joining Educational Mathematics)
  MathUI Workshop
- `rdfs:seeAlso`
  PlanetMath
  Semantic MediaWiki
  Connexions
  ActiveMath
- `rdf:type`
  Semantic Wiki
  Mathematical Editor
  Collaboration Tool
  Browser

Conclusion

- SWiM makes mathematical documents editable collaboratively and facilitates common workflows by exploiting the knowledge they contain.
- Domain-specific markup and ontology allows for advantages over generic semantic wikis and non-semantic mathematical wikis w. r. t. knowledge management
- Approach transferable to other domains (e. g. chemistry): decide on page granularity, capture semantics in ontology, extract RDF, integrate suitable editors

Roadmap

- ontology for narrative structures
- formalisation workflow
- dependency graph navigation
- refactoring support
- adaptive presentation

Acknowledgments

science foundation ireland
fondúireacht eolaíochta éireann

kwarc

Studienstiftung des deutschen Volkes

Position paper: A real Semantic Web for mathematics deserves a real semantics

P. Corbineau, H. Geuvers, C. Kaliszyk, J. McKinna, F. Wiedijk

ICIS, Radboud University Nijmegen, the Netherlands

Abstract. Mathematical documents, and their instrumentation by computers, have rich structure at the layers of presentation, metadata and semantics, as objects in a system for formal mathematical logic. Semantic Web tools [2] support the first two of these, with little, if any, contribution to the third, while Proof Assistants [17] instrument the third layer, typically with bespoke approaches to the first two. Our position is that a web of mathematical documents, definitions and proofs should be given a fully-fledged semantics in terms of the third layer. We propose a "Math-Wiki" to harness Web 2.0 tools and techniques to the rich semantics furnished by contemporary Proof Assistants.

1 Background and state of the art

We can identify four worlds of mathematical discourse available on the Web:

- Traditional mathematical practice: a systematic body of knowledge, organised around documents written by experts, most often in LaTeX, to varying degrees of sophistication. The intended audience is an expert readership, and the content is of high quality and reliability, having been through a rigorous editorial process. Indexing and cross-referencing is managed externally by journals themselves, augmented by tools such as CiteSeer, Google Scholar, and archival sites such as ArXiv;
- Wikipedia, MathWorld, *etc.*: universal readership and authorship, wide coverage, but relatively shallow and of variable reliability, with little systematic development of larger theories, and little or no critical gloss on the material;
- The Semantic Web, with the OMDoc standard [9] and tools like SWiM [11], for organising structured documents around a basic notion of "falling under a concept" (such concepts then further organised into content dictionaries);
- The language and (checked) libraries of *proof assistants*, in which concepts, definitions, statements, and most importantly, *proofs* of theorems are represented in a machine-checkable format.

We focus in this paper on the fourth world, as we expect it to be least familiar to readers of the paper, but more importantly because we believe that proof assistants offer a real, that is to say, formal *mathematical* semantics to (a Semantic Web of) mathematical documents. Our aim, and that of our partners in a European consortium, is to integrate all four worlds into a coherent whole, and develop a "MathWiki", a system for the collaborative authoring and communication of computer mathematics to the world.

Proof Assistants The basic idea of using computer programs to check mathematical proofs goes back to the archaeology of AI research. The 1960s saw the emergence of two basic paradigms: de Bruijn's AUTOMATH [16], and Milner's LCF. Both provide highly generic foundational approaches to representing mathematics: as a series of checked objects (definitions etc.) extending a body of knowledge from an initial axiomatisation (e.g. of arithmetic or set theory). In LCF the objects, including proofs of their properties, are obtained by running programs to produce values of an abstract datatype `thm`, that is to say they are *ephemeral* phenomena associated to the persistent program texts which give rise to them. In AUTOMATH, the objects — λ-terms in a dependently-typed language uniformly representing definitions and proofs — are themselves persistent and in principle may be independently rechecked, or otherwise processed.

Modern systems have elaborated these ideas with great sophistication, extensive libraries, and highly non-trivial formalisations:

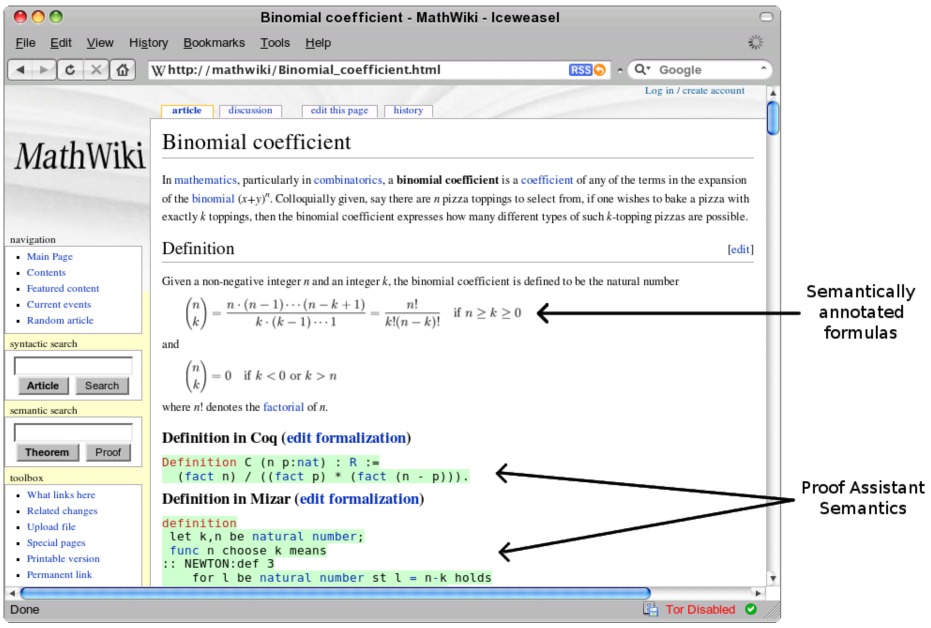
- The HOL Light system [8] is an LCF-style checker for higher-order logic; Harrison recently announced a proof of the analytic Prime Number Theorem;
- The Isabelle system [15] is also LCF-like, but adds a generic twist in terms of an AUTOMATH-like theory of representation: it is a *logical framework*, that is, it is generic over the underlying choice of logic and axiomatisation. It is available with libraries for both higher-order logic, and for ZF set theory. It has been used to formalise Gödel's completeness theorem, the consistency of the axiom of choice, the Prime Number Theorem, etc.;
- The Coq system [3] is type-theoretic, within which objects and proofs are λ-terms in a calculus of inductive and coinductive definitions; a notable development is Gonthier's formalisation of the Four Colour Theorem [6];
- The Mizar system [13], a proof checker for a strong version of set theory, emphasises developing a formalised library of standard, classical mathematics.

The decisive semantic advantage of all these systems over existing approaches to mathematical documents comes from the infrastructure of a formalised meta-level: names and binding to support substitutive definitions, definitional equality, hypothetical and general reasoning. The Proof Assistant and Semantic Web communities seem to differ over what constitutes a (mathematical) definition:

- in the Semantic Web a definition is a reference to a (canonical) textual description of the defined object; while
- for the proof assistant community a definition is a binding with a dynamic semantics given by a substitutive notion of *definitional equality*, namely the replacement of the named object (*definiendum*) by a body (*definiens*).

2 A project proposal: MathWiki

The MathWiki project proposes to combine a Wikipedia-like encyclopedia of mathematical notions and results, with a web-based integrated formal environment for collaboratively working with multiple proof assistants. Wikipedia has shown that it is possible to create large bodies of coherent knowledge, by providing lightweight (web-based) functionality to add material. In the MathWiki project we similarly want to provide lightweight web-based functionality to contribute to a repository of formalised mathematics. This should provide both a means to do large joint formalisations in a distributed way, but also the means to search and retrieve material, both at a low level, in terms of proof assistant-specific text, and at the high level of standard mathematical documents.



Binomial coefficient

In mathematics, particularly in combinatorics, a **binomial coefficient** is a coefficient of any of the terms in the expansion of the binomial $(x+y)^n$. Colloquially given, say there are n pizza toppings to select from, if one wishes to bake a pizza with exactly k toppings, then the binomial coefficient expresses how many different types of such k-topping pizzas are possible.

Definition [edit]

Given a non-negative integer n and an integer k, the binomial coefficient is defined to be the natural number

$$\binom{n}{k} = \frac{n \cdot (n-1) \cdots (n-k+1)}{k \cdot (k-1) \cdots 1} = \frac{n!}{k!(n-k)!} \quad \text{if } n \geq k \geq 0$$

and

$$\binom{n}{k} = 0 \quad \text{if } k < 0 \text{ or } k > n$$

where $n!$ denotes the factorial of n.

Definition in Coq (edit formalization)

```
Definition C (n p:nat) : R :=
  (fact n) / ((fact p) * (fact (n - p))).
```

Definition in Mizar (edit formalization)

```
definition
 let k,n be natural number;
 func n choose k means
:: NEWTON:def 3
    for l be natural number st l = n-k holds
```

Fig. 1. An example MathWiki page for the binomial coefficient

The MathWiki repository will include knowledge about mathematical concepts by the means of high level concept description pages. Those pages will include links to pages containing the finer details, which are, in the end, checked proof assistant code. We plan to directly incorporate into our project a certain number of state-of-the-art proof assistants. But the MathWiki itself will be open to other systems and it should be easy to incorporate them. The repository will contain all the large libraries of formal mathematics that already exist for the included proof assistants, like the Coq user contributions (*contribs*) and the Archive of Formal Proofs for Isabelle, in order to facilitate access to them.

We have created a prototype [4] that only supports Coq (without any semantic aspects yet), which suggests the project is technically feasible. In Figure 1 we sketch how the eventual system might look (including quoted material from Wikipedia for illustrative purposes).

Our first claim is that a mathematical semantic web where the mathematical notions refer to objects with a real formal semantics in a proof assistant will be profitable for users of mathematics because it improves preciseness and correctness. Our planned MathWiki system should substantiate that claim and open up to a wider community the rich collections of knowledge stored in the repositories of proof assistants and to facilitate the extension and editing of these repositories by outside users.

Our second claim is that the "medium" of *computer checkable formal proofs* will become a valuable asset in ICT, notably in verification and correctness of software and systems. At this moment there is not *one* type of medium for computer checkable formal proofs: basically each proof assistant has its own "media type". We think that in the future these media types will more and more converge and become exchangeable. A real mathematical semantic web is the platform for studying, comparing and exchanging these media types.

3 Why now: QED 15 years later?

The motivation for initiating this project precisely now is the convergence of several decisive factors. One of them is the success of the Wiki approach in general, and mostly the success of its application to the encyclopedic endeavour. This example shows that the collaborative approach is a good way of developing bodies of shared knowledge.

Another key factor is the availability of mature proof assistants with solid reputations and a certain quantity of formal developments. These proof assistants are way past toy examples and now allow outstanding results; they can handle large developments spanning hundreds of files.

Semantic web techniques now available provide a relevant presentation layer to the user. Although formal proofs are highly structured and hence easy to index, it is this extremely precise structure that can leave the user lost in the details, or unable to search or browse effectively.

The last key element we wish to stress is the availability of Web 2.0 technologies, which support the creation of web-based complex user interfaces. These technologies are important for our project since interactive proof development is by far the most popular way of using proof assistants.

Already in 1993 the authors of the QED Manifesto [1] had this vision: to let the whole world participate in creating a shared repository of formalised mathematics. We can speculate as to why this was an idea before its time: inevitably, user communities around each system felt keenly the supposed strengths of their own approach, and the perceived deficiencies of others'. The relative maturity of systems and their libraries has greatly mitigated this state of affairs.

The difficulty of formal proof also restrained the ambition of proof projects attempted, but with eyes on a bigger prize, collective development has become common practice in the formal proof community. This is how the biggest achievements were possible. Mizar and its MML are the primary example of the success of collective development though not very focused. More focused examples are the CompCert project in which a whole team participated in the verification of a C compiler and the Nijmegen repository of formalised mathematics (CoRN) [5]. The ongoing Flyspeck project [7] is another instance.

Proof assistants proposed to be part of the MathWiki project in the initial phase are Coq, Isabelle and Mizar. They cover three different foundational theories (Type Theory, Higher-Order Logic and Set Theory), and embrace classical as well as intuitionistic mathematics. They also have three different interaction modes: de Bruijn style, LCF-style and batch-mode interaction. Thus the three of them provide an excellent coverage of the variety among existing proof assistants.

4 Conclusion

The power of Wiki technology is to make building a new encyclopedia of mathematics a truly global democratic enterprise. Contemporary proof assistant technology has reached the point where we can imagine such a richly structured web of mathematics with a fully-fledged semantics in a formal system. A real Semantic Web for mathematics deserves a real semantics.

References

1. R. Boyer et al. The QED Manifesto. In Bundy, ed., *Automated Deduction – CADE 12*, *LNAI* 814. Springer, 1994.
2. S. Buswell, O. Caprotti, D.P. Carlisle, M.C. Dewar, M. Gaëtano, and M. Kohlhase. The OpenMath Standard, version 2.0, 2002.
3. Coq Team. *The Coq Proof Assistant Reference Manual V8.1.* INRIA, 2006.
4. P. Corbineau and C. Kaliszyk. Cooperative repositories for formal proofs. In Kauers *et al.* eds., *Calculemus/MKM*, *LNCS* 4573. Springer, 2007.
5. L. Cruz-Filipe, H. Geuvers, and F. Wiedijk. C-CoRN, the constructive Coq repository at Nijmegen. In Asperti *et al.* eds., *MKM*, *LNCS* 3119. Springer, 2004.
6. G. Gonthier. A computer-checked proof of the Four Colour Theorem, 2006.
7. T. C. Hales. Introduction to the flyspeck project. In Coquand *et al.* eds., *Mathematics, Algorithms, Proofs*, *Dagstuhl Proceedings* 05021. IBFI, Germany, 2005.
8. J. Harrison. HOL light: A tutorial introduction. In Srivas and Camilleri, eds., *Proceedings of FMCAD'96*, *LNCS* 1166. Springer, 1996.
9. M. Kohlhase. *OMDoc - An Open Markup Format for Mathematical Documents [version 1.2]*, *LNCS* 4180. Springer, 2006.
10. A.P. Krowne. An architecture for collaborative math and science digital libraries. Master's thesis, Virginia Tech Dept. of Computer Science, Blacksburg, VA, 2003.
11. C. Lange. SWiM – a semantic wiki for mathematical knowledge management. In Bechhofer *et al.* eds., *ESWC*, *LNCS* 5021. Springer, 2008.
12. C. Lange, S. McLaughlin, and F. Rabe. Flyspeck in a semantic wiki. Unpublished.
13. M. Muzalewski. *An Outline of PC Mizar.* Fondation Philippe le Hodey, 1993.
14. R.P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer. *Selected Papers on Automath*, *Studies in Logic and the Foundations of Mathematics* 133. Elsevier, 1994.
15. T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL - A Proof Assistant for Higher-Order Logic*, *LNCS* 2283. Springer, 2002.
16. D.T. van Daalen. A description of Automath and some aspects of its language theory. In Nederpelt et al. [14]. Article A.3.
17. F. Wiedijk, ed. *The Seventeen Provers of the World*, *LNCS* 3600. Springer, 2006.

Flyspeck in a Semantic Wiki
Collaborating on a Large Scale Formalization of the Kepler Conjecture

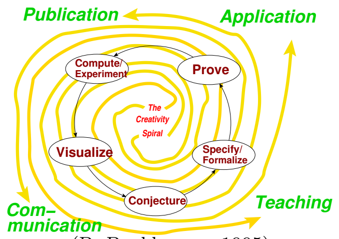
Christoph Lange[1], Sean McLaughlin[2], and Florian Rabe[1]

[1] Computer Science, Jacobs University Bremen,
{ch.lange,f.rabe}@jacobs-university.de
[2] School of Computer Science, Carnegie Mellon University, Pittsburgh,
seanmcl@gmail.com

Abstract. Semantic wikis have been successfully applied to many problems in knowledge management and collaborative authoring. They are particularly appropriate for scientific and mathematical collaboration. In previous work we described an ontology for mathematical knowledge based on the semantic markup language OMDoc and a semantic wiki using both. We are now evaluating these technologies in concrete application scenarios. In this paper we evaluate the applicability of our infrastructure to mathematical knowledge management by focusing on *the Flyspeck project*, a formalization of Thomas Hales' proof of the Kepler Conjecture. After describing the Flyspeck project and its requirements in detail, we evaluate the applicability of two wiki prototypes to Flyspeck, one based on Semantic MediaWiki and another on our mathematics-specific semantic wiki SWiM.

1 Scientific Communication and the Flyspeck Project

Scientific communication consists mainly of exchanging documents, and a great deal of scientific work consists of collaboratively authoring them. Common instances are writing down first hypotheses, commenting on results of experiments or project steps, and structuring, annotating, or re-organizing existing items of knowledge, as depicted in Buchberger's figure on the right.



(B. Buchberger, 1995)

Semantic markup languages for representing structures of scientific knowledge, and editing tools understanding them, are a promising approach to supporting this work.Besides generic approaches like SALT [6], the most extensive work in semantic markup has been in the domain of mathematics. Mathematical logic, depending on symbols and relationships between symbols, naturally lends itself well to formal exposition. Languages like MathML [24], OpenMath [29], and OMDoc [14] were developed to represent the clearly defined and hierarchical structures of mathematics in a way that preserves the intricate relationships. OMDoc employs Content MathML or OpenMath for structurally representing mathematical *objects* (symbols, numbers, equations, etc.) and adds two layers on top: Objects or informal text can be annotated as mathematical *statements* (symbol declarations, definitions, axioms, theorems, proofs, examples, etc.), and collections of interrelated statements can be grouped into *theories*.

With SWiM, a semantic wiki for mathematical knowledge management [22], we have investigated collaborative editing of OMDoc documents. Additionally, we host a public knowledge base and experimental ground about mathematical knowledge management on the web, powered by Semantic MediaWiki[3]. It has become evident that a wiki is a suitable tool for supporting the workflow of incremental formalization inherent to scientific writing. Wikis have not only shown to be appropriate for *writing*, but are also effective for project management, e. g. in corporate settings [23, 36]. We are therefore interested in applying our technologies to scientific knowledge engineering projects.

The target of our case study is the Flyspeck Project, which seeks to formally verify Thomas Hales' proof of the Kepler Conjecture [8, 9]. This conjecture asserts that the density of a packing of unit spheres in 3 dimensions is at most $\pi/(3\sqrt{2})$, the density of the face centered cubic and hexagonal close packings. Posed by Kepler in 1611, it formed part of Hilbert's 18th problem, and until its solution was recognized as one of the most famous unsolved problems of mathematics. Hales' proof, completed in 1995, was not accepted immediately by the mathematical community. Besides its considerable length, the proof relies essentially on computer calculations. The 300 pages of text and many thousands of lines of computer code made checking the proof for errors in the referee process unusually difficult, leading to a publication delay of nearly 10 years. In 2003, Hales proposed using computers to rigorously check the entire proof in detail, including the computer code. He dubbed this effort *Flyspeck*[4]. The software systems used in such formalizations are called *theorem provers* or *proof assistants*[5], examples being Isabelle [31], Coq [3], and Twelf [32]. With adequate human assistance they can verify that a purported proof follows from a given set of axioms and inference rules.

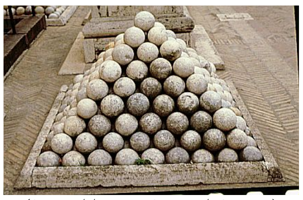

(http://tinyurl.com/3bxx2t)

Fig. 1. The face centered cubic packing

Modern proof assistants are still far from being able to check proofs at the level given in most journals and textbooks. A typical estimate is that it takes about a week to formalize a single page of mathematical text. Hales expects that it will take around 20 man-years to complete Flyspeck. Hales is compiling a LaTeX

[3] `http://mathweb.org/wiki/`

[4] The word "flyspeck" means, "to examine closely". It was found by Hales using a regular expression search of an English dictionary for the expression "F.*P.*K", for "Formal Proof of Kepler"

[5] The word "formalize" is used in many contexts in this field. In the remainder of this paper, we use "formal" and "formalize" loosely, possibly referring to any degree of colloquial or scientific formalization. We use "computerized" to mean that a theorem, proof or definition has been expressed in a proof assistant. Note that we consider computerized definitions and proofs formal "documents" as well.

book [10] of lemmas from different areas of mathematics that are needed in his proof. Its 450 pages contain a significant percentage of the mathematical results used in the proof, covering such disparate topics as plane, solid, and spherical geometry, graph theory and hypermaps, single and multivariable calculus, and plane and spherical trigonometry.

The first steps toward a computerized proof have already been taken. Nipkow and Bauer [27] proved the correctness of a fundamental algorithm in Isabelle. The other two main parts of the computer code, linear programming and global optimization, are currently being investigated in doctoral dissertations [39, 28]. A project page documents some of this progress and has a source repository containing the book of lemmas, as well as the formalized definitions of some important functions and inequalities [11]. Despite this considerable progress on the computer code, the bulk of the mathematical formalization remains to be done. This formalization will consist of two broad phases. First, a number of elementary mathematical theories (e.g. spherical geometry) need to be defined and the relevant lemmas proved. Then the specific aspects of the Kepler proof that relies on the elementary results need to be formalized. Given the content of the book mentioned above, we suspect that Flyspeck, in its final form, will consist of dozens of theories, with hundreds of definitions and thousands of lemmas.

Flyspeck is particularly appealing as a use case for a semantic wiki approach. While the ultimate result is to be a highly formal computerized proof, the current proof involves both highly formal and semi-formal mathematical knowledge. It contains descriptive and motivating yet informal text that should be preserved for human understanding. This quasi-formal information would be difficult to present in a strictly formal setting of a proof assistant. Secondly, the large number of lemmas, many independent or only loosely coupled, suggests a "crowdsourcing" approach will be beneficial. Both can be supported by a (semantic) wiki, as we will show in the following.

2 Supporting Flyspeck in a Semantic Wiki

Our focus in this work is on making the extent and structure of Flyspeck comprehensible, communicating where work needs to be done, and allowing collaborators to improve the structure and finally to contribute computerized proofs. For this the outline of the whole proof from the book [10] needs to be represented in the wiki, where the mathematical statements (including definitions, lemmas, and theorems) are available in a human-readable way (with formulae in LaTeX or presentational MathML) as well as a computerized presentation suitable for using in a theorem prover. In order to obtain a well-structured network of knowledge items, each mathematical statement should be presented on one wiki page, which shows its human-readable representation taken from the book, offers additional space for annotation, and allows for downloading a formal representation. Here, we are not yet considering formal proof checking *inside* the wiki, but rather using the wiki for communication about the projects and annotation of informal text.

2.1 Scenario

An example usage scenario is as follows (cf. fig. 2). A user wishes to contribute to Flyspeck. She looks at our wiki main page, which shows her what still needs to be done. Preferring trigonometry, she searches for open problems in that field. This returns a list of lemmas related to analysis from which she can choose one that seems possible given her time constraints. She reads the text of a paper proof culled from Hales' book and annotated by other wiki collaborators and downloads the relevant formal definitions and lemmas. She uses a proof assistant to begin formalizing the paper proof. At some point, she needs clarification on some definition and additionally has an idea on how to generalize this lemma. She thus asks for help, makes comments on the discussion pages of the wiki, and refines the annotations of the lemma. She completes her proof, and uploads the proof assistant file to the wiki. The wiki uses a theorem prover to check the proof for correctness and, if it is correct, adds it to the database.

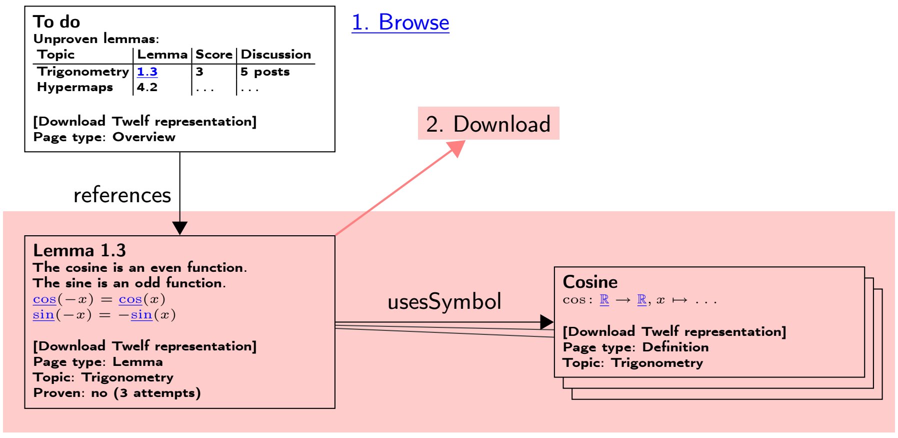


Fig. 2. Page Structure and Navigation

2.2 Requirements

With this scenario in mind, we propose that the wiki should minimally offer:

A knowledge base of the theory, constant, and lemma definitions.
A theory browser where a user can browse the knowledge by category, or search with keywords.
An editor to annotate and structure informal texts on their way to computerization.
A download area where one can download existing computerized definitions, lemmas, and proofs.
An upload area where one can upload new proofs.
Discussion pages to discuss issues involved in the formalizations.

The following set of annotations should support this minimal infrastructure:

Categorization by topic: In the beginning, one would mirror the narrative structure of the book (e. g. "sphere" being a subsection of "primitive volumes", which in turn is a section of the chapter "volume calculations"). Standardized ways of classifying mathematical topics, such as the Mathematical Subject Classification (MSC) [1], could be added later.

Project-organization metadata such as whether the proof of a lemma has already been computerized, or if someone is currently attempting a proof. This is essential so that two people do not duplicate work.

Dependency links: These can be links from individual symbols in mathematical formulae to the place where they are declared, or from any page p to other pages containing knowledge that is required for understanding p: either pages in the same wiki, or external resources like PlanetMath or Wikipedia articles. Authors should be able to add them where they are missing.

Discussion posts should be strongly tied to the topic being discussed, and classified into categories like question, answer, explanation, etc.

An enticing page for visitors and potential collaborators should give an impression of the extent and structure of the project (e. g. its size and its specialization into diverse fields of mathematics). For the developer, there should be tools for browsing and querying the knowledge. Not only should it be possible to query knowledge items by their annotations, but important query results must also be available as dynamically generated lists. Examples for queries are:

1. "Which lemmas about composite regions need to be proved?"
2. "What lemmas are difficult to prove?"
   (a) . . . in the sense that many people have already attempted them, but given up
   (b) . . . in the sense that many people have asked questions in the related discussion
3. "Are there textual resources I can read in order to understand the Jordan Curve Theorem?"
4. "What other lemmas could help me to prove this one?" (e. g. because they prove a related statement)

A volunteer who is willing to work out and contribute a computerized proof for a lemma should be able to download a self-contained computerized representation of this lemma and everything it depends on. Different notions of "dependency" can be supported, the most straightforward being that a lemma depends on the declarations and definitions of all symbols it uses and on the transitive closure of all symbols used by the latter. Related lemmas could be downloaded and assumed as axioms, under the assumption that those will be proved later, perhaps by other collaborators. Finally, assuming that the Flyspeck book [10] is written in a linear order, *all* definitions and lemmas before the current one in the narrative order could be used.

During the formalization of the knowledge, we anticipate that the definitions will undergo refactoring in order to facilitate the actual development of the proofs. (Historically, this has been the case with many large computerized proofs, cf. [5].) Refactoring support by the wiki would thus be advantageous. In fact, as definitions rely so heavily on each other, and the lemma statements rely on the definitions, Hales needs to oversee the computerization of the definitions so that the mathematical constants are correct[6]. This could be done by allowing him and other experienced mathematicians to *rate* the contributions of the collaborators.

3 Case Studies and Evaluation

So far, the Flyspeck project has four core members who collaborate via GoogleCode [11]. While the services offered by GoogleCode (a Subversion repository, a mailing list, and others) were found to be sufficient for the core development team, we were not satisfied with the wiki integrated into the GoogleCode web interface. Lacking support for mathematical formulae, it would not even allow for presenting the theorems and lemmas to be computerized in a human-readable fashion. This is important, as we suspect people would prefer to look at traditional mathematics text than proof assistant scripts when browsing. Furthermore, GoogleCode offers very little *structuring* support, which we believe will be essential for browsing and querying Flyspeck's large knowledge collection.

In the following sections, we evaluate two semantic wiki prototypes for their applicability to Flyspeck with regard to their support for annotations, browsing, and querying, as specified in section 2.2. One is based on Semantic MediaWiki, the other one on our own semantic wiki SWiM. For the case study, we took a simplified view of Flyspeck, using only the TEX sources of the Flyspeck book [10] and a Twelf [32] computerization of the definitions and lemmas of the chapter dealing with the foundations of trigonometry. The goal was to present the trigonometry chapter in a compelling way that we believed would scale 2-3 orders of magnitude.

Both systems are semantic wikis, where one resource (e. g. one mathematical theorem) is represented by one wiki page and relations between resources by links between pages. Both pages and links can be typed with terms from ontologies [30], which are either preloaded into the wiki or modeled ad hoc [17]. This is the prevalent approach of adding semantics to wikis, although other ways have been investigated [37]. Note that we have developed an ontology for mathematical knowledge (see sec. 3.2), but as this only focuses on the most essential structures, keeping it extensible in the wiki may be beneficial. Semantic wikis offer enhanced navigation capabilities. For example, they can usually display a summary of all typed links, grouped by type, for each page. They support searching for pages by type or by a page being source or target of a typed link[7]. Such queries can either be executed interactively or automated as *inline* queries embedded into

[6] For example, one can represent a vector as a function from the integers to the reals, or as a tuple of reals. The operations of vector spaces will depend on this representation, etc.

[7] Both explicit and inferred links (RDF triples) can be considered [17]

the content of a page [17]. Both systems we consider support this basic set of semantic wiki features.

3.1 Semantic MediaWiki 1.0

Semantic MediaWiki [17] is a semantic extension to MediaWiki, the system driving Wikipedia. Plain MediaWiki supports mathematical formulae written in LaTeX and allows for categorizing pages. Semantic MediaWiki interprets category membership as an instance-of relationship and supports the creation and editing of typed links (called properties). External ontologies can be referenced from the wiki, but at most sites powered by Semantic MediaWiki, site-specific ontologies are developed in an ad hoc manner [34].

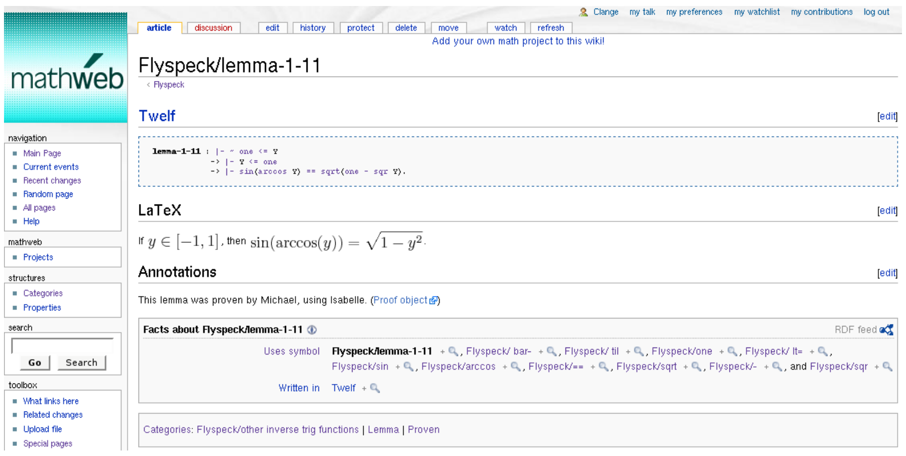
Prototype In Semantic MediaWiki, we imported the Twelf master source of Flyspeck via a custom upload page. The Twelf file was first enhanced by special comment lines marking the beginning and end of a declaration with information about topical categorization. The Twelf upload page handler breaks an uploaded file down into declarations and creates two wiki pages for each Twelf declaration: one page that just contains the Twelf listing, categorized in the OMDoc document ontology (e. g. *Lemma*; see section 3.2), and one container page that includes the Twelf page via MediaWiki's template inclusion mechanism, but also allows for including a LaTeX representation and leaves space for free-form annotations made by the contributors. Additionally, MediaWiki offers a discussion page for each page of mathematical content. The Twelf pages are overwritten on every import from the master source, whereas existing container pages remain untouched. This allows one to change the computerized version of a Twelf constant in the master source (e. g. if it is incorrectly specified) and re-importing it without losing the semantic markup and comments. During the import of a new symbol x, the upload extension recognizes all previously imported symbols y in the definition of the new symbol and creates links between x, y in the wiki.

The generated annotations can be used for browsing, either via the "fact box" (the summary of all typed links), or by the special "browse" page. For querying, Semantic MediaWiki offers a simple triple search, as well as inline queries. The query language corresponds to the small description logic $\mathcal{EL}^{++}$ [17], which, for example, does not support unrestricted negation. A query for unproven lemmas about a certain topic could only be performed if the "unprovenness" were explicitly annotated. The following queries additionally ask for lemmas available in a Twelf formalization:

```
<ask>[[Category:Unproven]] [[Category:Lemma]]
     [[Category:Trigonometry]] [[written in::Twelf]]</ask>
```

Exporting computerized representations of knowledge items is not yet supported conveniently. The Twelf listings can be viewed on their own pages, but due to the auto-generated symbol links in the source code, these are not suitable

[8] See http://mathweb.org/wiki/Flyspeck



Add your own math project to this wiki!

Flyspeck/lemma-1-11

‹ Flyspeck

Twelf [edit]

```
lemma-1-11 : |- ~ one <= Y
           -> |- Y <= one
           -> |- sin(arccos Y) == sqrt(one - sqr Y).
```

LaTeX [edit]

If $y \in [-1, 1]$, then $\sin(\arccos(y)) = \sqrt{1 - y^2}$.

Annotations [edit]

This lemma was proven by Michael, using Isabelle. (Proof object)

Facts about Flyspeck/lemma-1-11 — RDF feed

Uses symbol	Flyspeck/lemma-1-11 +, Flyspeck/ bar- +, Flyspeck/ til +, Flyspeck/one +, Flyspeck/ lt= +, Flyspeck/sin +, Flyspeck/arccos +, Flyspeck/== +, Flyspeck/sqrt +, Flyspeck/- +, and Flyspeck/sqr +
Written in	Twelf +

Categories: Flyspeck/other inverse trig functions | Lemma | Proven

Fig. 3. A Flyspeck lemma in Semantic MediaWiki[8]

for download. One would either have to implement a special Twelf download page that cleans these sources again, or one would have to implement the symbol linking as an extension of the rendering process.

Evaluation We found the ad hoc ontology development useful while prototyping the annotations that might be required for Flyspeck, e. g. project-related metadata like the information whether a lemma has already been proven, or categorization by topic. Semantic MediaWiki did not meet the requirements in places where ontologies already existed. For example, in structures of mathematical documents, it was possible to reference *vocabulary* from the OMDoc document ontology (see below), but not to apply further inference rules given there to items of mathematical knowledge. This is because Semantic MediaWiki does not support a full *import* of external ontologies. Most annotations were modeled by categorization, i. e. instantiation of classes—certainly not the most formal way of structuring knowledge in view of many classes just corresponding to narrative sections of the book, but the one that is supported best by Semantic MediaWiki. The inline queries were intuitive to write but not as powerful as required. Complex reasoning tasks like inference of dependencies are not possible in Semantic MediaWiki; in the restricted domain-specific setting of Flyspeck one could realize them by hard-coded extension functions. Semantic MediaWiki does not understand the semantics of mathematical formulae, as the LaTeX formulae cannot be annotated. The Twelf listings could be annotated, but at the cost of making them harder to download.

3.2 SWiM 0.2

SWiM is a semantic wiki targeted at mathematical knowledge management. Based on the general-purpose semantic wiki IkeWiki [17], it adds support for browsing, editing, rendering, importing and exporting mathematical documents written in OMDoc. The semantics of mathematical knowledge is mainly captured in the OMDoc markup, and more explicitly in a *document ontology*; whenever a wiki page containing OMDoc fragments is saved, its type and its (typed) relations to other items of mathematical knowledge in the wiki are extracted from the OMDoc XML markup and explicitly represented as RDF triples using terms of the OMDoc document ontology [18]. This ontology models those aspects of the three layers of mathematical knowledge supported by OMDoc to the extent supported by the expressivity of OWL-DL [25], including a limited inference of dependencies. Modeling all modules of the OMDoc specification in this ontology is not totally complete, though most mathematical statements as well as key aspects of theories have been implemented. Relevant classes for Flyspeck would be *Lemma*/*Theorem*/*Corollary*/... (all being subclasses of *Assertion*), *Proof*, *Symbol* (a symbol declaration), *Definition*, and the properties *Proof–proves–Assertion* and *Symbol–hasDefinition–Definition*.

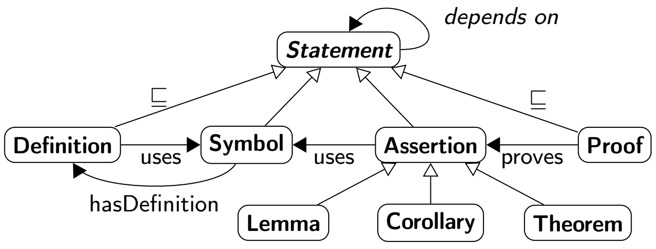


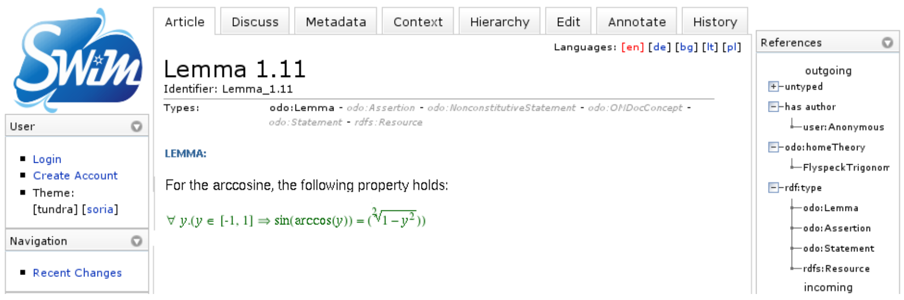
Fig. 4. A relevant subset of the OMDoc document ontology

In the current version 0.2 of SWiM, the browsing of mathematical documents is powered by the document ontology; whenever RDF triples having the current page as subject or object are available[9] the IkeWiki user interface can display them either as navigation links (see figure 5) or in a graph view. Documents are presented as XHTML+MathML, with mathematical symbols linked to their declarations.

Prototype We manually converted part of the trigonometry lemmas to OMDoc for SWiM. Additionally, we can auto-generate OMDoc documents from the Twelf source with a converter and import them into SWiM using the built-in import functionality.

As every SWiM page has an associated discussion page and discussion posts are semantically represented using the SIOC ontology [35], one can support the coordination of the project by queries like query 2b from section 2.2. Work on determining a relevant subset of OMDoc and its document ontology for discussions

[9] In a mathematical document such as those we consider, most of these triples use from the OMDoc document ontology.



Languages: [en] [de] [bg] [lt] [pl]

Lemma 1.11

Identifier: Lemma_1.11

Types: odo:Lemma - *odo:Assertion* - *odo:NonconstitutiveStatement* - *odo:OMDocConcept* - *odo:Statement* - *rdfs:Resource*

LEMMA:

For the arccosine, the following property holds:

$\forall\, y.(y \in [-1, 1] \Rightarrow \sin(\arccos(y)) = (\sqrt[2]{1 - y^2}))$

```
outgoing
[+]-untyped
[-]-has author
    └─user:Anonymous
[-]-odo:homeTheory
    └─FlyspeckTrigonom
[-]-rdf:type
    ├─odo:Lemma
    ├─odo:Assertion
    ├─odo:Statement
    └─rdfs:Resource
incoming
```

Fig. 5. A Flyspeck lemma in SWiM

is currently in progress. Pages and non-OMDoc links can be annotated with types from ontologies loaded into the wiki[10].

Another powerful feature of SWiM is that authors can embed inline SPARQL queries into wiki pages. Query 1 can be posed without explicitly annotating "unprovenness", making use of negation as failure [33]:

```
SELECT ?l WHERE { ?l rdf:type odo:Lemma .
                  ?l swrc:isAbout <Composite_Regions> .
                  OPTIONAL { ?p rdf:type odo:Proof .
                             ?p odo:proves ?l . }
                  FILTER ( ! bound(?p) ) }
```

As OMDoc supports all degrees of formalizing mathematical knowledge, computerized data can be downloaded in their OMDoc representation using SWiM's export feature and then be converted to Twelf by client-side software [14, chap. 25.2].

Evaluation Annotating mathematical structures with SWiM is easy if the built-in OMDoc editor is used. Other annotations required for Flyspeck, such as categorizations or information about the progress of the project, can be made, but not in an ad hoc way, which we would have found useful in the prototyping phase. Instead, one would have to import an existing ontology into the wiki, or create it using the built-in ontology editor, and then one would be able to annotate documents using terms from that ontology.

Browsing is well supported, with incoming and outgoing navigation links being displayed. Additionally, the neighborhood of the current resource in the RDF graph can be browsed visually.

Queries are powerful, but not always short and intuitive (see above). Alternatively, one could enhance the ontology and make use of the integrated Pellet OWL-DL reasoner (see [17]), which supports a more powerful logic than Semantic MediaWiki, and get the same result with a simple query for instances

[10] Types of OMDoc links are automatically extracted from the markup; see above.

of a specially defined class. For unproven lemmas, the following axiom would suffice:

$$\text{LemmaWithoutProof} \equiv \text{Lemma} \sqcap \neg(\exists \text{proves}^{-1}.\text{Proof})$$

However, it remains to be evaluated how well the wiki scales with DL reasoning enabled. First experiments with Pellet let the system considerably slow down (an experience also made by the IkeWiki author [17]), so alternatives will have to be investigated as well.

System	Semantic MediaWiki	SWiM
Ontology availability	none built in	sufficient (OMDoc)
Ontology editing/ extensibility	easy, ad hoc in place	easy, but only via dedicated user interface
Page annotation	easy but not sufficiently expressive	easy and expressive
Inline queries	easy to write but not sufficiently powerful	harder to write but more powerful
Browsing	intuitive	intuitive, optional graph browser
Reasoning	not sufficient	powerful but slow
Semantics/annotation of formulæ	not supported	very powerful but harder to author
Annotation of computerized content	not directly supported by our extension	powerful (OMDoc markup)

Fig. 6. Summary of the evaluation of the features

4 Related Work

Outside of wikis, the combination of computerized proofs and human-readable text has been investigated in Isar [38], an alternative literate programming language for Isabelle, and in Mizar [26], whose language of Mizar is close to mathematical vernacular. In contrast to Isar, there is a large web-based library of Mizar proofs. It is browsable and searchable on the web but managed in a centralized and hierarchical way, which is not comparable to wiki collaboration.

Informal mathematical knowledge is currently managed in comprehensive encyclopediæ like the mathematical sections of Wikipedia[11] or in PlanetMath[11], which focuses on mathematics and is powered by a highly customized wiki-like system. The pages in these systems are categorized and searchable in full-text, with additional metadata records in PlanetMath. Neither of the systems is a *semantic* wiki, and for lacking typed links they fail to answer queries essential for

[11] See `http://www.wikipedia.org` or `http://www.planetmath.org`, respectively, and [19] for a more comprehensive evaluation.

Flyspeck, such as query 1 from section 2.2, and they do not link mathematical symbols to their declarations; instead, the author has to provide links he considers relevant in the text surrounding the formula.

Recently, there is a growing interest in integrating proof assistants with wikis. *Logiweb* is not a wiki but a distributed system for publishing machine checked mathematics in high-quality PDF that shares part of the key wiki principles [7]. Anybody can contribute to a Logiweb site and edit new pages in a simple text syntax. On the other hand, Logiweb does not offer other essential features. For example, browsing by traversing links is supported neither in the editor nor in the generated PDF, and a built-in search or query facility is not offered. Logiweb does not allow for *exchanging* knowledge as required for Flyspeck: Documents can only be exported in presentational formats like PDF or TEX, but their semantic structures cannot be exported in mathematical markup or theorem proving languages. The way Logiweb checks proofs is not compatible with other theorem provers, as all calculi and proof tactics need to be defined in the Logiweb system itself. *ProofWiki* is an integration of the ProofWeb Coq frontend into MediaWiki [4]. Coq's export tools are used to generate browsable HTML or LATEX with linked symbols from the proof scripts. Generating index pages, such as lists of all definitions or all theorems, is planned, but not yet in a way that could be customized by users. So far, there is just text search, and dependencies among knowledge items are only computed for exporting proofs but not used for browsing inside the system. Pages can either be formal proof scripts (with restricted possibilities to include informal comments) or informal wiki pages. Semi-formal documents or stepwise formalizing of knowledge are not supported. Importing and exporting Coq proof scripts to and from the wiki is possible. While the authors provide instructions on how to integrate other theorem provers, doing so would be a lot of work, as there is no abstraction layer or metalanguage for exchanging or converting data. Both Logiweb and ProofWiki are "semantic" in the sense that the integrated proof checker utilizes the mathematical knowledge in the wiki pages. But the semantics is not utilized for anything else, such as facilitating browsing or editing, or connecting to semantic web services. Developing and verifying formal proofs in the wiki is not yet the focus of Flyspeck in this early stage, but it may be required later.

5 Conclusion and Further Work

Our preliminary experiments lead us to believe that, due to its rich semantic web and OMDoc infrastructure, future work toward supporting Flyspeck should continue in the SWiM infrastructure. For the text-based page format of MediaWiki, features that rely on structures like the linking of symbols could only be realized in an ad hoc way using, say, regular expressions. Relying on the XML infrastructure of OMDoc, these features are either already available or easier to develop. However, rapidly *prototyping* our first ideas about the wiki support required for Flyspeck was easier in Semantic MediaWiki due to its ability to design ad hoc ontologies and its implementation in the interpreted language PHP.

Importing For this case study, we created OMDoc from Twelf. OMDoc also offers support for the alternative workflow of stepwise formalization as well. One could either start by converting the Flyspeck book from LaTeX to HTML with MathML formulæ and formalize the presentation markup into content markup step by step, or one could start the formalization on the TeX side. There, one would formalize the book to sTeX, a content-oriented TeX notation for OMDoc, which can then be converted to OMDoc [13]. Either way involves a TeX-to-XML transformation, which has been tested in large scale in our group [2].

Annotating The case study showed that the editing of ontologies in SWiM should become more flexible. While a fixed OMDoc document ontology can be preloaded, it should be possible to add other annotations ad hoc. We have not focused on document *editing* in detail here, but additional editing services relying on the document ontology are planned for SWiM 0.3 [20, 21]. Finally, using the module system of OMDoc and refactoring the knowledge into more smaller theories could help to simplify the structure of Flyspeck for browsing and to explicate the dependencies between components of the proof.

Browsing In the Semantic MediaWiki prototype we realized that the narrative structure of the book is not adequately represented by a simple hierarchy of categories. OMDoc has more powerful ways of putting content into narrative structures [15]. We are going to cover them with the document ontology and utilize them for browsing.

Querying Proof search will be greatly simplified if the semantic-aware search engine MathWebSearch [16] is used. It applies substitution tree indexing to mathematical formulae. That means, for example, that a query for $\int f(x \mathbin{?} z)dx$ would also find $\int f(y + z)dy$. Equivalence up to α-renaming of bound variables is obviously essential for a serious query language.

Different Theorem Provers If several parts of the proof are done in different theorem provers, highly non-trivial and mostly novel translations become necessary to provide one single proof object. Here OMDoc could be used as an exchange format between theorem prover languages, and formal translations could be specified in OMDoc itself. While this line of research is interesting, it is difficult for us to foresee what kinds of translations, if any, will be needed.

Download Dependencies, which we need for bundling download packages, can partly be inferred by a DL reasoner using the document ontology, but for a complete support of OMDoc's notion of dependency, an OMDoc-specific calculus will have to be applied, which is currently in development.

Upload We have not implemented uploading a proof directly to the wiki to have it checked. This is easy in theory as we simply need to hook up the theorem prover, but requires some effort to get the theorem prover to run on the wiki server. This should be done soon, as it will relieve the maintainers.

Acknowledgments We would like to thank Stefan Decker, Michael Kohlhase, and Immanuel Normann for their feedback particularly during the case studies.

References

1. American Mathematical Society. 2000 mathematics subject classification. `http://www.ams.org/msc/`, 2000.
2. `arXMLiv`: Translating the arχiv to xml+mathml, 2007. `http://kwarc.info/projects/arXMLiv/`.
3. Y. Bertot and P. Castéran. *Interactive theorem proving and program development: Coq'Art: the Calculus of Inductive Constructions.* Texts in theoretical computer science. Springer, 2004.
4. P. Corbineau and C. Kaliszyk. Cooperative repositories for formal proofs. In Kauers et al. [12].
5. G. Gonthier. A computer-checked proof of the four colour theorem. Unpublished manuscript, 2005.
6. T. Groza, S. Handschuh, K. Möller, and S. Decker. SALT – Semantically Annotated LaTeX for scientific publications. In E. Franconi, M. Kifer, and W. May, editors, *ESWC*, volume 4519 of *Lecture Notes in Computer Science.* Springer, 2007.
7. K. Grue. The layers of Logiweb. In Kauers et al. [12].
8. T. Hales. A proof of the Kepler conjecture. *Annals of Mathematics*, 162:1065–1185, 2005.
9. T. Hales. The Kepler conjecture. *Discrete and Computational Geometry*, 36(1):1–269, 2006.
10. T. Hales. Flyspeck : A Blueprint of the Formal Proof of the Kepler Conjecture. Unpublished manuscript, 2008.
11. T. Hales and S. McLaughlin. The Flyspeck Project. `http://code.google.com/p/flyspeck`, 2007.
12. M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors. *MKM/Calculemus 2007*, number 4573 in LNAI. Springer, 2007.
13. M. Kohlhase. sTeX: A LaTeX-based workflow for OMDoc. In OMDoc – *An open markup format for mathematical documents [Version 1.2]* [14], chapter 26.15.
14. M. Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2].* Number 4180 in LNAI. Springer, 2006.
15. M. Kohlhase, C. Müller, and N. Müller. Documents with flexible notation contexts as interfaces to mathematical knowledge. In P. Libbrecht, editor, *Mathematical User Interfaces Workshop*, 2007.
16. M. Kohlhase and I. Şucan. A search engine for mathematical formulae. In T. Ida, J. Calmet, and D. Wang, editors, *Artificial Intelligence and Symbolic Computation, AISC*, number 4120 in LNAI. Springer, 2006.
17. M. Krötzsch, S. Schaffert, and D. Vrandečić. Reasoning in semantic wikis. In G. Antoniou, U. Aßmann, C. Baroglio, S. Decker, N. Henze, P.-L. Pătrânjan, and R. Tolksdorf, editors, *3rd Reasoning Web Summer School*, volume 4636 of *LNCS*. Springer, 2007.

18. C. Lange. The OMDoc document ontology. http://kwarc.info/projects/docOnto/omdoc.html, 2007.
19. C. Lange. SWiM – a semantic wiki for mathematical knowledge management. Technical Report 5, Jacobs University Bremen, 2007.
20. C. Lange. SWIM development roadmap. https://trac.kwarc.info/swim/roadmap/, 2007.
21. C. Lange. Towards scientific collaboration in a semantic wiki. In A. Hotho and B. Hoser, editors, *Bridging the Gap between Semantic Web and Web 2.0*, 2007.
22. C. Lange. SWiM – a semantic wiki for mathematical knowledge management. In S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, editors, *ESWC*, volume 5021 of *Lecture Notes in Computer Science*, pages 832–837. Springer, 2008.
23. B. Leuf and W. Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley Professional, 2001.
24. Mathematical Markup Language (MathML) version 3.0. W3C working draft, World Wide Web Consortium, 2007. http://www.w3.org/TR/MathML3.
25. D. L. McGuinness and F. van Harmelen. OWL web ontology language overview. W3C recommendation, W3C, 2004.
26. Mizar mathematical library. Web Page at http://mizar.org/library/.
27. T. Nipkow, G. Bauer, and P. Schultz. Flyspeck I: Tame Graphs. In U. Furbach and N. Shankar, editors, *International Joint Conference on Automated Reasoning*, volume 4130 of *LNCS*. Springer, 2006.
28. S. Obua. Proving bounds for real linear programs in isabelle/HOL. In J. Hurd and T. F. Melham, editors, *Theorem Proving in Higher Order Logics*, volume 3603 of *LNCS*. Springer, 2005.
29. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004. http://www.openmath.org/standard/om20.
30. E. Oren, R. Delbru, K. Möller, M. Völkel, and S. Handschuh. Annotation and navigation in semantic wikis. In Völkel et al. [37].
31. L. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *LNCS*. Springer, 1994.
32. F. Pfenning and C. Schürmann. System description: Twelf : A meta-logical framework for deductive systems. In H. Ganzinger, editor, *16th International Conference on Automated Deduction (CADE)*, volume 1632 of *LNAI*. Springer, 1999.
33. E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C Recommendation, World Wide Web Consortium, 2008. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.
34. Sites using Semantic MediaWiki. http://www.semantic-mediawiki.org/w/index.php?title=Sites_using_Semanti%c_MediaWiki&oldid=781, 2008.
35. SIOC – Semantically-Interlinked Online Communities, 2007. http://sioc-project.org/.
36. D. Tapscott and A. D. Williams. *Wikinomics – How Mass Collaboration Changes Everything*. Portfolio, 2006.
37. M. Völkel, S. Schaffert, and S. Decker, editors. *1st Workshop on Semantic Wikis*, volume 206 of *CEUR Workshop Proceedings*, 2006.
38. M. Wenzel. Isar — a generic interpretative approach to readable formal proof documents. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Théry, editors, *Theorem Proving in Higher Order Logics: TPHOLs'99*, volume 1690 of *LNCS*, pages 167–184. Springer, 1999.
39. R. Zumkeller. Formal global optimisation with taylor models. In U. Furbach and N. Shankar, editors, *International Joint Conference on Automated Reasoning*, volume 4130 of *LNCS*. Springer, 2006.

Using Attention and Context Information for Annotations in a Semantic Wiki

Malte Kiesel, Sven Schwarz, Ludger van Elst, and Georg Buscher

Knowledge Management Department
German Research Center for Artificial Intelligence DFKI GmbH,
Trippstadter Straße 122, 67663 Kaiserslautern, Germany

{firstname.lastname}@dfki.de

Abstract. For document-centric work, meta-information in form of annotations has proven useful to enhance search and other retrieval tasks. Since creating annotations manually is a lot of work, it is desirable to also tap less obtrusive sources of meta-information such as the user's context (projects the user is working on, currently relevant topics, etc.) and attention information (what text passages did the user read?).
The Mymory project uses the semantic wiki *Kaukolu* that allows storing attention and context information in addition to standard semantic wiki metadata. Attention annotations are generated automatically using an eyetracker. All types of annotations get enriched with contextual information gathered by a context elicitation component.
In this paper, an overview of the Mymory system is presented.

Shortened version. Please see `http://www.dfki.de/mymory` *for the full paper.*

1 Annotations in document-centered Work

Generally speaking, an *annotation* "is extra information asserted with a particular point in a document or other piece of information"[1], and as such, it is a widely used element in document-centered work: We put general comments ("This is similar to xy!") or ratings ("Important!") of text passages into annotations or annotate text with imperative statements ("Verify!"). Thus, annotating is a means to individualize and personalize documents which, for the rest, might be alike for a group of information consumers. *Formal* annotations as they are commonly applied in the Semantic Web context, in contrast, contain mainly *meta-data*, i. e., they primarily aim at *machine*-understandability of a document's content in order to enable automated information services (see [7, 2] for overviews of the role of annotation in document-centric Knowledge Management and the Semantic Web). For personal knowledge management, in addition to annotations that formalize document semantics, personal annotations along with attention and user context information are needed. Mymory addresses this by supporting

[1] `http://en.wikipedia.org/w/index.php?title=Annotation&oldid=175839314`

additional types of annotations: **Conceptual annotations** are used to classify (Web 2.0 speak: “tag”) document passages. Mymory relies on a Personal Information Model (Ontology) (PIMO), i.e., the user(s)’s conceptualization of his/their (knowledge work) world. Instead of tagging passages with simple text labels, PIMO concepts are used to annotate and classify the passages. **Attention annotations** are used to store how much attention the user invested for which parts of the document. Mymory uses several user observation techniques to gather attentional evidences. For example, an eye tracking device is used to recognize which passages the user really *reads*, which ones he has *skimmed over*, and which ones the user did not seem to have viewed at all. **Highlightings and comments** are comparatively simple annotations. However, *context-sensitive* management (i.e., context-sensitive storage and retrieval) of such annotations conveys a context-sensitive view of the document as the user (himself) had in the past when he created these annotations. It allows a quick “flashback” and reminds him of his past understanding and usage of that document. **Contextualized annotations** are annotations containing meta-information about the user’s context at the time of the creation of the annotation. This allows context-sensitive views of a document and is a way to enable scalable, massive usage of annotations in a multi-user/multi-context scenario as is typical in shared document repositories.

2 Overview over Mymory Components

The **PIMO: The Personal Information Model (Ontology)** [3] provides a vocabulary for describing information elements on an individual desktop, thereby comprehensively reflecting a user’s personal view on his information landscape. The Mymory PIMO offers rather general concepts of knowledge work (Person, Organization, Location, Document, etc.) and allows for extensions of these upper models with more specific group or personal concepts (e. g., concrete project types or organizational structures). It provides the basic vocabulary for annotating concrete information elements and for describing a user’s context.

Automatic User Context Capturing — Mymory aims at transforming the knowledge worker’s workplace into a *context-sensitive* document-centric work environment. Automatic user observation generates a continuous stream of contextual evidences which are then fed into a context elicitation framework[2]. That way, the user’s context is captured without disturbing the user. A detailed description of the user observation and context elicitation framework is beyond the scope of this paper. See [4–6] for an overview of modeling, using, and accessing user context for knowledge management scenarios. The `User Observation Hub`[3]

[2] Technically, this is achieved by observing the usage of a number of applications such as the web browser and editor using plug-ins, keeping track of events within these applications, and matching these events and the content displayed in the applications against a number of rules that are partially autogenerated from the user’s PIMO.

[3] `http://usercontext.opendfki.de/wiki/UserObservationHub`

is an open-source (Java) project responsible for the gathering and distribution/forwarding of user observation data.

Our scenario assumes that the knowledge worker's world is conceptualized and modeled using the PIMO. Mymory's context elicitation automatically gathers evidences to estimate for every PIMO concept a degree of attention/relevance with respect to the user's current context. Thus, the context as elicited and processed by the system consists of a map associating such an attention level with every PIMO concept. As the user works and his behavior continuously adapts during his current task, the attention levels of the concepts adapt accordingly.

The Mymory Workbench AKA Kaukolu Wiki — The wiki component of Mymory is implemented by Kaukolu, a semantic wiki research prototype. Kaukolu is an extension of JSPWiki[4]. Its new frontend features are implemented using Dojo[5], a JavaScript framework that allows higly interactive user interfaces.

Typically, semantic wikis associate wiki pages with semantic resources, and allow links between pages to get typed, possibly according to some ontology known to the wiki. However, while this approach is elegant in terms of simplicity and ease of use, there are several drawbacks:

– The rigid mapping between wiki pages and semantic resources imposes severe limits on the possible use cases. Mapping complex ontologies to the wiki or creating proper instances for these ontologies in a wiki is as difficult as capturing the knowledge present in a large wiki page (which would correspond to a resource with hundreds of properties).
– Handling of existing documents, be it existing wiki pages or other documents, is difficult. Metadata has to be added into the page text, changing the actual document. For texts such as law documents or finalized versions of documents this might not be desirable.
– Handling of further information concerning annotations such as provenance or context information is difficult. Personal annotations are not supported.

For Mymory another way of creating annotations has been implemented. Annotations can get created for any text part[6]–in this regard, annotations in Kaukolu are similar to annotations or *notes* created in a standard word processing application. These are displayed in connection with the text they are associated with but do not show up as text characters or markup in neither editing nor viewing mode unless requested by the user.

An example can be seen in figure 1 that depicts a software license text and its annotations. This text contains lots of separate mentions of certain facts–after all, a license is a collection of legal statements. Expressing semantics of this type of document is impossible using the page-resource mapping technique. Only

[4] `http://www.jspwiki.org/`
[5] `http://dojotoolkit.org/`
[6] Technically, these annotations are implemented by creating an *AnnotationAnchor* for each RDF annotation, associating the RDF resource of the annotation with a part of the wiki markup by means of storing character offsets of the annotated text. Offsets are updated on markup edits by a modified text diff algorithm.

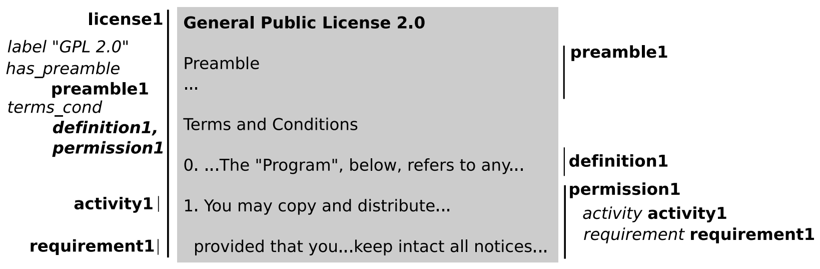


Fig. 1. Annotating a software license in Kaukolu.

annotating a license with a few rather generic information fragments would be possible, de facto using an ontology with one or few classes and lots of properties, and creating one large instance.

Using the text annotation approach, fine-grained annotation is possible. Both text decomposition and assigning complex fact representations to individual text fragments can be done. Since a tight connection between annotations (or more precisely, the information contained therein) and the text exists, text passages concerning or expressing certain facts can be retrieved quite easily.

3 Features of the Mymory Workbench

Creating Annotations in Kaukolu: In Mymory, annotations are created in two different ways: Users may manually create annotations, and an eyetracker component can automatically mark read or skimmed passages, keeping track of the user context in the course[7]. From user perspective, creating manual annotations in Kaukolu is pretty straightforward. Once a text part to be annotated is selected, right-clicking opens an annotation window where possible annotation types are displayed. These types and corresponding dialogs are fetched from ontologies loaded in Kaukolu's RDF repository or, if configured accordingly, also from external sources using a custom implementation.

Using Annotations in Search: All data and metadata found in the system can be queried in Kaukolu's advanced search feature. Advanced search is implemented as a faceted search paradigm: The user can select one or more restrictions on text and annotation characteristics graphically ("Search for paragraphs I read yesterday in context of project Firestart“). Search always returns wiki text paragraphs as results; searching for standalone RDF resources or authors directly is not supported. This was done to keep the system simple and to keep some resemblance with a normal wiki search in which users expect text passages to be returned.

[7] Information on the algorithms used for reading detection using the eyetracker can be found in [1]. Note that the eyetracker used is built-in in the display and does not require wearing glasses or other equipment.

Creating Documents based on Search Results: Text passages found in search can be used to create new documents. The idea here is that this way it is possible to “remix” texts to form documents that fit to new requirements. Passages are copied, not referenced—however, provenance information indicating the source of copied paragraphs is kept as a new annotation.

Using Annotations for Personalized Views: Kaukolu also supports filtering a page’s display based on annotations during normal page view. While many possible implementations of this can be imagined, we currently implemented two major ways of filtering. One can filter wiki pages by attention information—in practice, this means that passages bearing a *read* or *skimmed* annotation are displayed as usual while all other passages are grayed out. This helps finding (formerly) relevant passages in large documents. Filtering by context information is implemented differently: As context information is meta-information about annotations, we filter the annotations to display in the wiki page. For example, one can choose to display only annotations created in the context of a certain project or topic.

Acknowledgement

The Mymory project is funded by the Bundesministerium für Bildung und Forschung under grant 01 IW F01 and by the IST Programme of the European Union under grant FP6-027705.

References

1. Georg Buscher. Attention-based information retrieval. In *SIGIR ’07: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval (doctoral consortium)*, 2007.
2. Siegfried Handschuh. *Creating Ontology-based Metadata by Annotation for the Semantic Web.* PhD thesis.
3. Leopold Sauermann, Andreas Dengel, Ludger van Elst, Andreas Lauer, Heiko Maus, and Sven Schwarz. Personalization in the epos project. In M. Bouzid and N. Henze, editors, *Proceedings of the International Workshop on Semantic Web Personalization, Budva, Montenegro, June 12, 2006*, pages 42–52, 2006.
4. Sven Schwarz. A context model for personal knowledge management applications. In Th. Roth-Berghofer, St. Schulz, and D. B. Leake, editors, *Modeling and Retrieval of Context, Second International Workshop, MRC 2005, Edinburgh, UK*, volume 3946 of *Lecture Notes in Computer Science*, pages 18–33. Springer, 2006.
5. Sven Schwarz and Thomas Roth-Berghofer. Towards goal elicitation by user observation. In A. Hotho and G. Stumme, editors, *Proceedings of the LLWA 2003*, pages 224–228, Karlsruhe, oct 2003. AIFB Karlsruhe, GI.
6. Roza Shkundina and Sven Schwarz. A similarity measure for task contexts. In *Proceedings of the Workshop Similarities - Processes - Workflows in conjunction with the 6th International Conference on Case-Based Reasoning*, 2005.
7. Victoria S. Uren, Philipp Cimiano, José Iria, Siegfried Handschuh, Maria Vargas-Vera, Enrico Motta, and Fabio Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Journal of Web Semantics*, 4(1):14–28, 2006.

RDF Authoring in Wikis

Florian Schmedding, Christoph Hanke, and Thomas Hornung

Institute of Computer Science, Albert-Ludwigs University Freiburg, Germany
{schmeddi, hankec, hornungt}@informatik.uni-freiburg.de

Abstract Although the Semantic Web vision is gaining momentum and the underlying technologies are used in many different areas, there still seems to be no agreement on how they should be used in everyday documents, such as news, blogs or wiki pages. In this paper we argue that two aspects are crucial for the enrichment of this documents with semantic annotations: full support for RDF and close integration of the annotations with the continuous text. The former is necessary because many common relationships cannot be expressed by attribute-value-pairs, the latter reduces redundancy and enables Web browsers to help readers using the contained data. To gain further insights, we implemented an RDFa-capable extension for MediaWiki and report on improvements for wiki use cases and other applications on top of the contained data.

1 Introduction

Since the vision of the Semantic Web [1] has been described, different knowledge markup and ontology definition languages, such as RDF [2] and OWL [3] have been proposed and standardized. A recent survey has shown that these languages are mostly applied to highly-structured domains with a well-understood semantics, e.g. for drug discovery [4]. In the revisited version [5] of the original vision the authors acknowledge that the Semantic Web has not reached the expected adoption yet. We believe that this is because the lion's share of the content on the Web is only available in presentation-oriented HTML documents without any semantic markup. Therefore to reach a critical user base, a clear benefit for the users of everyday documents, such as news, blogs or wiki pages, has to be established.

So far, semantic annotations were added to HTML documents in a very informal and restricted manner with respect to the semantic complexity of the information. In our opinion these approaches still suffer from two major drawbacks: lack of expressiveness and separation of text and annotations. The goal of our approach is to have full RDF expressivity while retaining the proximity of metadata and normal textual content. The latter is especially important for reusing existing external applications or enabling third parties to make use of the semantic content in a standardized way, e.g. for extracting calendar data.

Projects such as DBpedia[1] have shown that Wikipedia[2] already contains a lot of relevant structured metadata, e.g. the population of cities, and hence is

[1] http://dbpedia.org
[2] http://wikipedia.org

an ideal candidate for the adoption of Semantic Web technologies to enrich the existing content with semantic annotations. In this paper we describe an extension for MediaWiki[3], which allows to directly embed these semantic annotations while editing the wiki article. The main features are full support of RDF, including blank nodes, and the direct embedding of the resulting annotations in the generated XHTML presentation of the wiki article.

The remainder of the paper is organized as follows. In Section 2 we introduce and evaluate semantic annotation formats with respect to our requirements. In Section 3 we describe our extension to MediaWiki and present a use case about geo-political facts of countries in Section 4. In Section 5 we discuss related work, Section 6 gives an outlook on future work, and we conclude with Section 7.

2 Semantic Annotation Proposals

In line with our proposal, all semantic annotations should be embedded in the written article to avoid the administrative overhead of maintaining separate documents (HTML and e.g. RDF/XML). Additionally, there are less redundancies and update problems with a single document which is both human- and machine-readable from a single Web address.

Currently, there are three competing proposals for annotating semantics in HTML documents: Microformats[4], eRDF[5], and RDFa [6]. For obvious reasons, RDF/XML [7] is out of the question because it is not designed to contain readable text. Because eRDF only supports a subset of RDF and the informal semantics of Microformats we chose RDFa as annotation language.

Another orthogonal approach to embedding semantics in HTML documents is GRDDL [8]. It is designed to extract RDF data from any XML document via specialized XSLT transformations. Redundancy in text and RDF data is therefore omitted, but there is no connection between text and RDF data.

3 RDFa Wiki Extension

We implemented a prototype as an extension of MediaWiki to evaluate our approach. Here, the main focus is on augmenting the existing wiki syntax to enable users to embed arbitrary RDF statements into regular articles. The syntax design especially considers the following three requirements:

1. Subject and object of a statement can be any desired URI (or blank nodes),
2. Subject and predicate should be invisible to the reader and literals should be masqueradable,
3. Single statements are made within one unit, as distributed statements are vulnerable to partial deletion, which could alter the semantics.

[3] `http://www.mediawiki.org`
[4] `http://microformats.org`
[5] `http://research.talis.com/2005/erdf/wiki/Main/RdfInHtml`

In general, the semantic statements in our wiki extension include subject, predicate and object, although the subject is not mandatory. To annotate an existing wiki text, the user has to choose the desired object in the wiki text and place the predicate and optionally a subject in front of it. The whole semantic statement is delimited by <sem>-tags. URIs for subject and object are expressed using the common MediaWiki link notations. The predicate has to be written in CURIE [6] style, e.g. `cc:license` instead of the expanded URL `http://creativecommons.org/ns#license`. Applying these rules a statement about the external page `www.mypage.de` would look like this:

```
My homepage is licensed under <sem> [http://www.mypage.de] cc:license
[http://mylicense.org/ my own license] </sem>.
```

Omitting the subject (`[http://www.mypage.de]`) would create an equivalent statement but about the current page. In both cases the only value visible to the user is the URI, or rather the label of the object.

If the object is not denoted in link notation, the object value is interpreted as a (XML-)Literal. Literals can additionally receive a dataype and a label. This can be used to provide a date in a machine readable format, which means we masquerade the machine-readable data with an alternative representation. For example the sentence *The meeting takes place on 7th of August 2008* could be annotated in the following way, where "2008-08-07" represents the machine-readable date and *7th of August 2008* is the alternative representation:

```
The meeting takes place on <sem>[http://www.futuremeeting.com] dc:date
"2008−08−07"^^xsd:date 7th of August 2008 </sem>.
```

A further feature of our extension is the possibility of using blank nodes. For this we introduce a three bracket notation to provide a name for the blank node variable. This concept is useful for adequate modeling of n-ary relationships [9], e.g. to describe the border between two countries, where also the length of the border is of interest. An exemplary statement is given here:

```
The border between <sem>[[[border]]] mond:bordering [[Spain]]</sem> and
<sem>[[[border]]] mond:bordering [[France]]</sem>
has a length of <sem>[[[border]]] mond:length 623</sem> km.
```

Additonally the type of the subject can be classified using the `inof` attribute of the <sem>-tag:

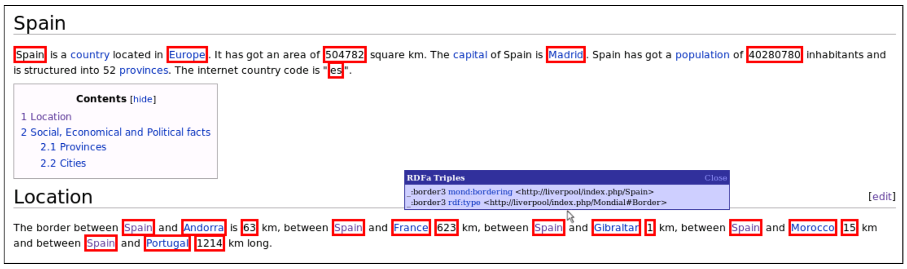
```
<sem inof="mond:Border">[[[border]]] mond:bordering [[Spain]]</sem>
```

Figure 1 shows a geographical wiki page about Spain. The JavaScript tool RDFa-Highlight[6] can be used in any browser to mark all semantically annotated areas.

4 Use Cases

Mondial [10] is a collection of political and geographical data, which covers typical concepts that we expect in a semantically enhanced version of Wikipedia.

[6] `http://www.w3.org/2001/sw/BestPractices/HTML/rdfa-bookmarklet/`


Spain

Spain is a country located in Europe. It has got an area of 504782 square km. The capital of Spain is Madrid. Spain has got a population of 40280780 inhabitants and is structured into 52 provinces. The internet country code is "es".

Contents [hide]
1 Location
2 Social, Economical and Political facts
2.1 Provinces
2.2 Cities

RDFa Triples Close
```
_:border3 mond:bordering <http://liverpool/index.php/Spain>
_:border3 rdf:type <http://liverpool/index.php/Mondial#Border>
```

Location [edit]

The border between Spain and Andorra is 63 km, between Spain and France 623 km, between Spain and Gibraltar 1 km, between Spain and Morocco 15 km and between Spain and Portugal 1214 km long.

Figure 1. Wiki page about Spain with marked semantic annotations (red). The blue box shows two statements about a blank node.

For this reason, we chose it as the basis to populate our prototype wiki with approximately 5,500 test pages.

4.1 Wiki Page Generation

We used a simplified version of the Mondial RDFS ontology[7] for the generation of the wiki pages, e.g. an excerpt for Spain is depicted in Figure 1. For the sake of illustrating the features of the ontology, we concentrated on the therein defined concepts and relationships. Obviously, we could have also enriched regular Wikipedia articles with additional semantic annotations.

Although the pages were generated according to a pre-defined template the simplified articles demonstrate the feasibility of our approach for real-world scenarios. It is also a good example of how to bootstrap semantic wikis from existing database content.

4.2 Ontology Maintenance

In contrast to other wikis, which separate the metadata from wiki articles, in our approach RDF vocabularies can be defined within the articles itself. This follows as an immediate consequence of the RDF support in our syntax. New definitions could be stated on any arbitrary wiki page or in a more structured way, using a reserved wiki category or special page. For example, the abovementioned Mondial ontology is defined in a separate article by means of our new wiki syntax. This enables the wiki community to collectively define and evolve ontologies with the same syntax used for authoring semantic articles.

4.3 Data Import

Since the RDFa standard is on its way to becoming a W3C recommendation, we expect the number of accessible XHTML+RDFa pages to constantly increase in the near future. Each of these pages could be seen as a remote information source,

[7] `http://www.dbis.informatik.uni-goettingen.de/Mondial/`

analogously to a SPARQL endpoint. This would enable us to use a coherent query language to both specify queries on our wiki and to include these remote sources as well in our wiki articles as dynamic data sources. An example from the Mondial theme would be to include the gross domestic product of a (future) semantically annotated version of the CIA World Factbook[8]. In this case the changes would occur only once a year, but the same general concept applies to including the most current publications of researchers in the relevant articles. Since for most data on the Web, especially homepages, it is not realistic to expect the data to be availabe via SPARQL endpoints, we expect a reasonable application area of our coherent integration approach.

5 Related Work

Similar to Semantic MediaWiki [11], we base our extension on MediaWiki. But unlike this project, our main focus was to have maximum RDF support for authors, instead of maintaining the current wiki syntax. Although this requires additional effort on the side of the authors, we believe that the benefits of the added semantics outweigh this inconvenience. For example, we support subjects different from the current page. BOWiki [12] is an extension of Semantic MediaWiki and is additionally capable of representing n-ary predicates but is restricted to a specialized biological domain. Kaukolu [13] also supports subjects different from the current page but has no full support for blank nodes. IkeWiki [14] is geared towards knowledge engineers and provides a sophisticated user interface and ontology reasoning. Our approach is geared towards shallow ontologies [5] and regular users. OntoWiki [15] offers a visual editor for easy editing of RDF content and provides semantically enhanced search strategies. Its main focus is on the acquisition of instance data and knowledge engineering projects. We are more interested in enriching normal wiki texts with embedded semantics. SweetWiki [16] also uses RDFa to embed semantics directly in the articles. However, their major focus is on providing keywords, or so-called tags, for specific articles or objects, e.g. images, inside the article. Our focus is on authoring complex RDF relations between several entities within the article. Finally, the internal structure of the Maariwa [17] wiki is based on an ontology meta model, i.e. each page either represents a class, an individual or a set of individuals. The annotations are then interpreted as properties of the class or individual, respectively. To query information they introduce a proprietary query language called MarQL. As discussed in Section 4.2 we propose a less rigid approach to specifying ontologies within arbitrary articles.

6 Future Work

To allow non expert users to formulate complex annotations in their articles it is crucial to provide an intuitive and easy-to-use editing environment. Inspired by

[8] `https://www.cia.gov/library/publications/the-world-factbook/index.html`

the successful WYSIWYG principle, we are currently working on a rich internet application for editing and annotating semantic Wiki articles in an integrated fashion.

Up to now, the available visualization tools for contained RDFa annotations in (X)HTML pages are rather limited. Given the specific Wiki content of Mondial we envision a more sophisticated graphical presentation of contained RDFa annotations. Currently, we are exploring different approaches of how to better support the user in browsing and understanding the contained annotations.

Additionally, the close proximity of semantic annotations to the textual content opens the door for new information retrieval applications, e.g. to combine keyword-based searches with semantic enhancements. By only querying the contained RDF data in a triple store, unannotated text is not considered. At the moment, we are investigating the impact of combining the approach by [18] with RDFa annotated pages.

Measurement units are currently not considered, but could be handled similar to the Semantic MediaWiki [11] proposal. An open question is how to handle articles in different languages about the same concept: should the annotations be shared between the different versions or does each language belong in a separate semantic unit? This a general question, which is not specific to our approach, and is relevant for each semantic wiki to some extent.

7 Conclusion

We have shown a semantic extension for MediaWiki and how it helps to improve the application of semantic wikis as well as the benefits of the directly embedded annotations for other applications, e.g. for developing semantic-aware search engines. Additionally, our approach could contribute to the proliferation of semantically enriched content on the Web, especially with a higher-level editing environment that hides the syntactic details of the wiki syntax. If the advantages of these semantic annotations would be visible to and demanded by end users the willingness of authors to employ these techniques would increase significantly. We believe this is possible in the near future due to the standardization of RDFa by the W3C and expect a wide adoption and support in Web browsers as well as innovative uses by other third party tools.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (May 2001)
2. Manola, F., Miller, E.: RDF Primer. `http://www.w3.org/TR/rdf-primer` (2004)
3. Smith, M.K., Welty, C., McGuinness, D.L.: OWL Web Ontology Language Guide. `http://www.w3.org/TR/owl-guide/` (2004)
4. Feigenbaum, L., Herman, I., Hongsermeier, T., Neumann, E., Stephens, S.: The Semantic Web in Action. Scientific American **297** (December 2007) 90–97
5. Shadbolt, N., Berners-Lee, T., Hall, W.: The Semantic Web Revisited. IEEE Intelligent Systems **21**(3) (July 2006) 96–101

6. Adida, B., Birbeck, M., McCarron, S., Pemberton, S.: RDFa in XHTML: Syntax and Processing. `http://www.w3.org/TR/rdfa-syntax/` (2008)
7. Beckett, D.: RDF/XML Syntax Specification (Revised). `http://www.w3.org/TR/rdf-syntax-grammar/` (2004)
8. Connolly, D.: Gleaning Resource Descriptions from Dialects of Languages (GRDDL). http://www.w3.org/TR/grddl/ (2007)
9. Noy, N., Rector, A.: Defining N-ary Relations on the Semantic Web. `http://www.w3.org/TR/swbp-n-aryRelations/` (2006)
10. May, W.: Information extraction and integration with FLORID: The MONDIAL case study. Technical Report 131, Universität Freiburg, Institut für Informatik (1999)
11. Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., Studer, R.: Semantic Wikipedia. Journal of Web Semantics **5** (2007) 251–261
12. Backhaus, M., Kelso, J., Bacher, J., Herre, H., Hoehndorf, R., Loebe, F., Visagie, J.: BOWiki – A Collaborative Annotation and Ontology Curation Framework. In: CKC. (2007)
13. Kiesel, M.: Kaukolu: Hub of the Semantic Corporate Intranet. In Völkel, M., Schaffert, S., eds.: SemWiki. (2006)
14. Schaffert, S.: IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In: WETICE. (2006) 388–396
15. Auer, S., Dietzold, S., Riechert, T.: OntoWiki - A Tool for Social, Semantic Collaboration. In: ISWC. (2006) 736–749
16. Buffa, M., Gandon, F.L., Ereteo, G., Sander, P., Faron, C.: SweetWiki: A Semantic Wiki. J. Web Sem. **6**(1) (2008) 84–97
17. Landefeld, R., Sack, H.: Collaborative Web-Publishing with a Semantic Wiki. In: CSSW. (2007) 23–34
18. Bast, H., Chitea, A., Suchanek, F.M., Weber, I.: ESTER: Efficient Search on Text, Entities, and Relations. In: SIGIR. (2007) 671–678

AceWiki: Collaborative Ontology Management in Controlled Natural Language

Tobias Kuhn

Department of Informatics, University of Zurich, Switzerland
tkuhn@ifi.uzh.ch
http://www.ifi.uzh.ch/cl/tkuhn

Abstract. AceWiki is a prototype that shows how a semantic wiki using controlled natural language — Attempto Controlled English (ACE) in our case — can make ontology management easy for everybody. Sentences in ACE can automatically be translated into first-order logic, OWL, or SWRL. AceWiki integrates the OWL reasoner Pellet and ensures that the ontology is always consistent. Previous results have shown that people with no background in logic are able to add formal knowledge to AceWiki without being instructed or trained in advance.

1 Introduction

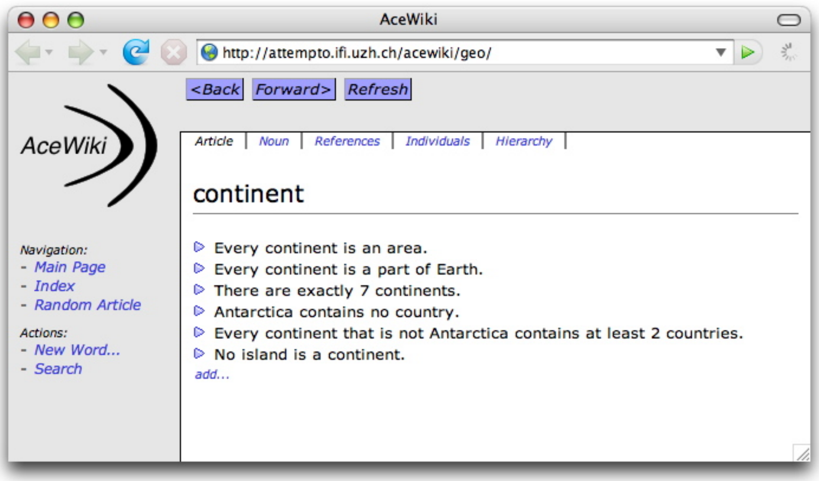
Since ontologies are often defined within communities, semantic wikis could be used for their collaborative creation and management. Unfortunately, most of the existing semantic wikis do not support expressive ontology languages in a general way. They do not allow the users to add complex axioms like "every landlocked country borders no sea". Furthermore, the existing semantic wikis are often hard to understand for people who are not familiar with the technical terms of logic and ontologies.

AceWiki[1] tries to solve both problems by using controlled natural language. Ordinary people who have no background in logic should be able to understand, modify, and extend the formal content of a wiki.

Many existing semantic wikis are classical wikis enriched with semantic annotations. The goal is not to manage stand-alone ontologies, but rather to give some kind of formal backbone to the wiki articles. We follow a different approach — similar e.g. to the *myOntology* project [7] — by providing a wiki that is dedicated to building and maintaining ontologies. In contrast to myOntology, we do not restrict ourselves to lightweight (i.e. relatively inexpressive) ontologies. The use of controlled natural language allows us to express also complex axioms in a natural way. Figure 1 shows a screenshot of the AceWiki interface.

In our usage scenario, a community of domain experts uses AceWiki to create and maintain a formal knowledge base in a collaborative manner. There are two exemplary wiki instances — one about geography and the other about protein interactions — that demonstrate how AceWiki could be used to represent knowledge of such communities.

[1] See [6] and http://attempto.ifi.uzh.ch/acewiki



continent

- Every continent is an area.
- Every continent is a part of Earth.
- There are exactly 7 continents.
- Antarctica contains no country.
- Every continent that is not Antarctica contains at least 2 countries.
- No island is a continent.

Fig. 1. The web interface of the AceWiki prototype

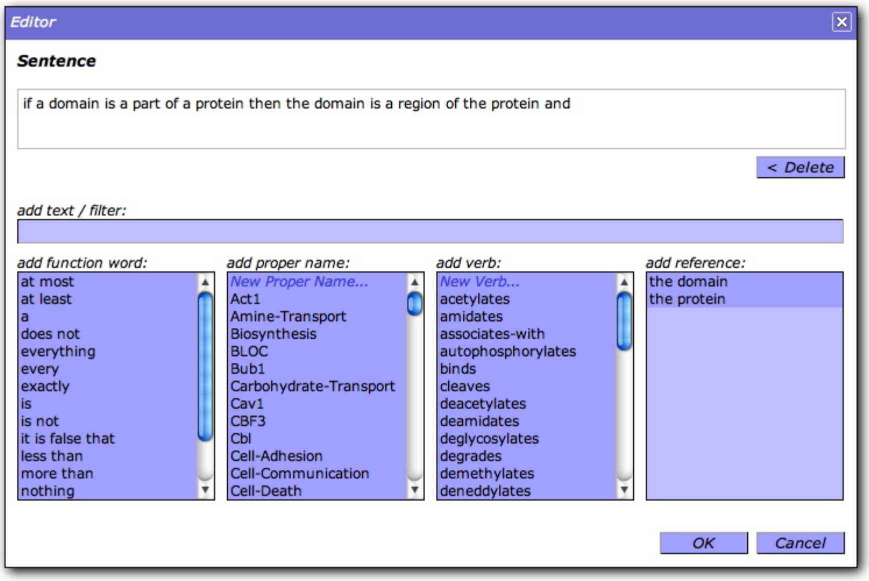
AceWiki has been introduced in [6]. Since then, several new features have been added, for example the integration of a reasoner and the support for number restrictions (“at most 3”, “exactly 5”, etc.).

2 Attempto Controlled English

Attempto Controlled English (ACE)[2] is the controlled natural language that is used for AceWiki. ACE appears completely natural since it is a subset of English. Restrictions of the syntax and the definition of a small set of interpretation rules make it a formal language that is automatically translatable into first-order logic. ACE supports a wide range of natural language constructs: singular and plural noun phrases, active and passive voice, relative phrases, anaphoric references, existential and universal quantifiers, negation, modality, and more. In the past, ACE has successfully been applied for different tasks in different research areas, for example as a query language for ontologies [1], as a knowledge representation language for the biomedical domain [4], and as a rule language for a multi-semantics rule engine [5].

Furthermore, ACE has been used as a natural language front-end to OWL with a bidirectional mapping of ACE to OWL [3]. This mapping covers all of OWL 2 except data properties and some very complex class descriptions. AceWiki relies on this work for translating ACE sentences into OWL, which allows us then to do reasoning with existing OWL reasoners.

[2] See [2] and `http://attempto.ifi.uzh.ch`


Editor

Sentence

if a domain is a part of a protein then the domain is a region of the protein and

< Delete

add text / filter:

add function word:
at most
at least
a
does not
everything
every
exactly
is
is not
it is false that
less than
more than
nothing

add proper name:
New Proper Name...
Act1
Amine-Transport
Biosynthesis
BLOC
Bub1
Carbohydrate-Transport
Cav1
CBF3
Cbl
Cell-Adhesion
Cell-Communication
Cell-Death

add verb:
New Verb...
acetylates
amidates
associates-with
autophosphorylates
binds
cleaves
deacetylates
deamidates
deglycosylates
degrades
demethylates
deneddylates

add reference:
the domain
the protein

OK
Cancel

Fig. 2. The predictive editor of AceWiki

3 Design and Evaluation

The goal of AceWiki is to show that semantic wikis can be more natural and at the same time more expressive than existing semantic wikis.

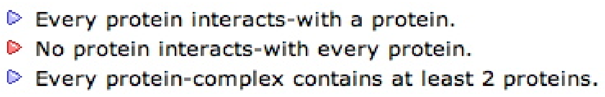
Naturalness is achieved by representing the formal statements in ACE. Since ACE is a subset of natural English, every English speaker can immediately read and understand the content of the wiki. In order to enable easy creation of ACE sentences, AceWiki provides a predictive editor that shows step-by-step the words that are syntactically possible at a given position in the sentence. Figure 2 shows a screenshot of the predictive editor of AceWiki. Furthermore, the AceWiki interface does not use technical terms like “ontological element”, “property”, or “subclass” but uses instead terms like “word”, “transitive verb”, or “hierarchy” which should be much more familiar to people with no background in logic.

AceWiki makes use of the high expressivity of ACE that goes beyond OWL and SWRL. We do not like the idea of cutting down the expressivity just for the sake of reasoning performance. Even if some statements become so complex that it is almost impossible to do reasoning with them, it is better to have them formalized than just left out. We do not lose anything, since we are free to ignore those complex statements for certain reasoning tasks.

In our previous work [6], we conducted a user experiment that proved that ordinary people with no background in logic are able to deal with AceWiki. The participants — without being instructed how to interact with the interface — were asked to add knowledge to AceWiki. About 80% of the created sentences were correct and sensible. This is remarkable since most of the sentences were quite complex: more than 60% of them contained an implication or a negation or both. Using the predictive editor which the participants had never seen before, they needed on average only five minutes to create their first correct sentence.

4 Reasoning in AceWiki

We have started to integrate the OWL reasoner Pellet[3] into AceWiki. Since ACE sentences can be beyond the expressivity of OWL, the reasoner cannot consider all sentences. In order to make this clear to the users, each sentence is tagged as blue (inside of OWL) or red (outside of OWL):

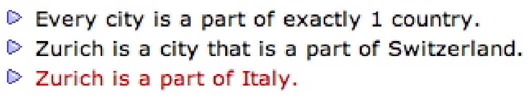


▷ Every protein interacts-with a protein.
▶ No protein interacts-with every protein.
▷ Every protein-complex contains at least 2 proteins.

In this way, it is easy to explain to the users that only the blue statements are considered when the reasoner is used. We plan to provide an interface that allows skilled users to export the formal content of the wiki and to use it within an external reasoner or rule-engine. Thus, even though the red statements cannot be interpreted by the built-in reasoner they can still be useful.

Consistency checking plays a crucial role because any other reasoning task requires a consistent ontology in order to return useful results. Most other semantic wikis do not have this problem since their languages are simply not expressive enough to ever run into inconsistency.

In order to ensure that the ontology is always consistent, AceWiki checks every new sentence — immediately after its creation — whether it is consistent with the current ontology. Otherwise, the sentence is not included in the ontology:



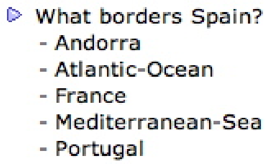
▷ Every city is a part of exactly 1 country.
▷ Zurich is a city that is a part of Switzerland.
▷ Zurich is a part of Italy.

After the user created the last sentence of this example, AceWiki detected that it contradicts the current ontology. The sentence is included in the wiki article but the red font indicates that it is not included in the ontology. The user can remove this sentence again, or keep it and try to reassert it later when the rest of the ontology has changed.

For this approach, it is very important to perform incremental reasoning which Pellet supports only partially at the moment. For that reason, AceWiki does not scale very well. We expect that future reasoners will be able to run much faster in such incremental scenarios.

[3] `http://pellet.owldl.com/`

Not only asserted but also inferred knowledge can be represented in ACE. At the moment, AceWiki can show inferred class hierarchies and class memberships. Furthermore, we are working on a query feature for AceWiki. Questions will be formulated in ACE and evaluated by the reasoner:



What borders Spain?
- Andorra
- Atlantic-Ocean
- France
- Mediterranean-Sea
- Portugal

Thus, ACE can be used not only as an ontology- and rule-language, but also as a query-language.

5 Conclusions

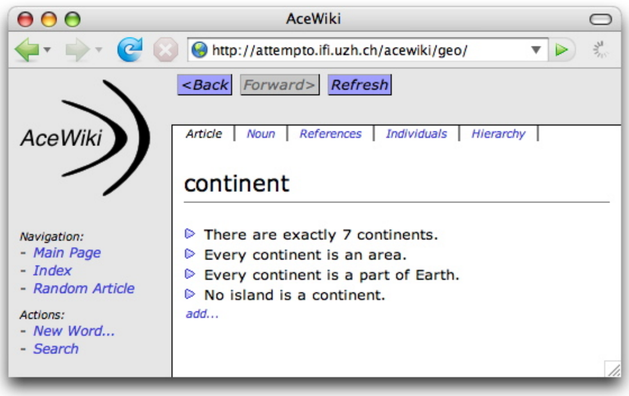
The AceWiki prototype shows how ontologies can be managed in a natural way within a wiki. It demonstrates how semantic wikis using controlled natural language can be expressive and easy to use at the same time. Our previous evaluation showed that AceWiki is indeed easy to learn. We explained how AceWiki ensures — in a very simple way — the consistency of the ontology which is the basis for other integrated reasoning services.

References

1. Abraham Bernstein, Esther Kaufmann, Norbert E. Fuchs, June von Bonin. Talking to the Semantic Web — A Controlled English Query Interface for Ontologies. *Proc. 14th Workshop on Information Technology and Systems*, 2004
2. Norbert E. Fuchs, Kaarel Kaljurand, Gerold Schneider. Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces. *Proc. 19th International FLAIRS Conference (FLAIRS'2006)*, 2006
3. Kaarel Kaljurand. *Attempto Controlled English as a Semantic Web Language.* PhD thesis, Faculty of Mathematics and Computer Science, University of Tartu, 2007
4. Tobias Kuhn, Loïc Royer, Norbert E. Fuchs, Michael Schroeder. Improving Text Mining with Controlled Natural Language: A Case Study for Protein Interactions. *Proc. Third International Workshop on Data Integration in the Life Sciences 2006 (DILS'06)*, Springer, 2006
5. Tobias Kuhn. AceRules: Executing Rules in Controlled Natural Language. *Proc. First International Conference on Web Reasoning and Rule Systems (RR2007)*, Springer, 2007
6. Tobias Kuhn. AceWiki: A Natural and Expressive Semantic Wiki. *Proc. of Semantic Web User Interaction at CHI 2008: Exploring HCI Challenges*, CEUR Workshop Proceedings, 2008
7. Katharina Siorpaes, Martin Hepp. myOntology: The Marriage of Ontology Engineering and Collective Intelligence. *Proc. Bridging the Gap between Semantic Web and Web 2.0*, 2007

Natural

AceWiki is a semantic wiki that is easily understandable by everybody. The wiki articles are written in ACE (Attempto Controlled English) that is a controlled natural language.



continent

- ▷ There are exactly 7 continents.
- ▷ Every continent is an area.
- ▷ Every continent is a part of Earth.
- ▷ No island is a continent.

ACE supports a wide range of natural language constructs:

- Proper names, nouns, verbs, *of*-constructs, ...
- Singular and plural noun phrases
- Relative phrases
- Anaphoric references
- Existential and universal quantifiers
- Negation

... and much more. Some examples:

- ▷ Every country controls every city of the country.
- ▷ Italy is a country that is a part of Europe.
- ▷ Switzerland borders exactly 5 countries.
- ▷ Switzerland does not border a sea.

Expressive

ACE is a formal language that is translatable into logic and other languages, for example OWL. Some examples of OWL-compliant sentences:

- ▷ Every country that borders no sea is a landlocked-country.
- ▷ If something X borders something Y then Y borders X.
- ▷ Antarctica contains no country.
- ▷ There are exactly 7 continents.

ACE is more expressive than OWL. You can express complex statements (e.g. rules) that are beyond OWL:

- ▷ No country borders every country.
- ▷ If Liechtenstein is a country then Andorra is a country.
- ▷ If a country contains an area and does not control the area then the country is an unstable-country.

ACE supports also questions which are used in AceWiki to state queries:

- ▷ Which continents contain more than 10 countries?
- ▷ Zurich is a part of what?
- ▷ Which cities overlap-with more than 1 continent?
- ▷ Switzerland borders which country that borders Spain?

Thus, ACE is an all-in-one language.

ALL-IN-ONE

- ✓ Ontology Language
- ✓ Rule Language
- ✓ Query Language

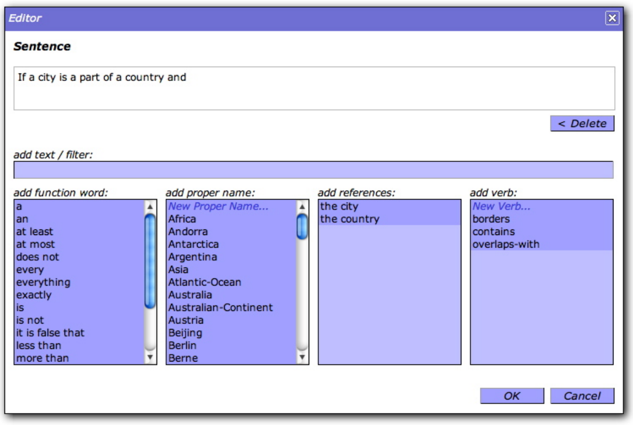
AceWiki

Attempto Group
University of Zurich
Department of Informatics

Tobias Kuhn
tkuhn@ifi.uzh.ch
http://attempto.ifi.uzh.ch/acewiki

Easy to Use

AceWiki is not only easy to read and understand, but also easy to modify and extend. Users do not need to understand the details of ACE. A predictive editor helps them word-by-word to create sentences that comply with the ACE syntax:



If a city is a part of a country and

add function word:	add proper name:	add references:	add verb:
a	New Proper Name...	the city	New Verb...
an	Africa	the country	borders
at least	Andorra		contains
at most	Antarctica		overlaps-with
does not	Argentina		
every	Asia		
everything	Atlantic-Ocean		
exactly	Australia		
is	Australian-Continent		
is not	Austria		
it is false that	Beijing		
less than	Berlin		
more than	Berne		

We performed a small-scale user experiment to test the usability of AceWiki. The participants were asked to add general knowledge to AceWiki:

- 20 participants (no background in logic and ontologies)
- No instructions how to use the interface
- They spent on average 47 minutes on AceWiki
- They created in total 186 sentences
  - 80% were correct and sensible
  - 61% were complex (using negation or implication)

Reasoning

AceWiki integrates the OWL reasoner Pellet that considers all OWL-compliant sentences of the wiki. It ensures that the ontology is always consistent:

- ▷ Every country is a part of exactly 1 continent.
- ▷ Every country that borders Switzerland is a part of Europe.
- ▷ Germany borders Switzerland.
- ▷ Germany is a part of Asia.

The reasoner infers class memberships and presents them in ACE:

- ▷ Switzerland is an area.
- ▷ Switzerland is a country.
- ▷ Switzerland is a landlocked-country.
- ▷ Switzerland is an object.

Also class hierarchies are inferred and presented in ACE:

Upward

- ▷ Every country is an area.
- ▷ Every country is an object.

Downward

- ▷ Every baltic-state is a country.
- ▷ Every landlocked-country is a country.

Furthermore, AceWiki supports inline-queries that are answered by the reasoner:

- ▷ Which cities are a part of Switzerland?
  - Berne
  - Geneva
  - Lucerne
  - Zurich

Next-Generation Wikis: What Users Expect; How RDF Helps

Axel Rauschmayer

Institut für Informatik, LMU München,
Oettingenstr. 67, 80538 München, Germany
`http://hypergraphs.de/`

Abstract. Even though wikis helped start the web 2.0 phenomenon, they currently run the risk of becoming outdated. In order to find out what aspects of wikis will survive and how wikis might need to evolve, the author held a survey among wiki users. This paper argues that adding RDF integration to wikis helps meet the requirements implicitly contained in the answers of that survey. Technical details are given by looking at the semantic wiki Hyena.

Key words: Next-generation wikis, Semantic Wiki, RDF, Semantic Web

1 Introduction

Wikis have been a popular web application for some time now. But the recent rise of Ajax [1] has changed the perception of web applications: Many wiki-like web sites have appeared, often specialized to do a single task well (where wikis are more universal). Examples include Google Calendar and Flickr.

This begs the question: What aspects of wikis will survive in the Web 2.0 age? What aspects are worth preserving? To answer this question, one has to find out what people use wikis for, what features they like and what features they are missing. A small survey on this topic helped the author do that. This paper interprets the survey results as requirements for the next generation of wikis and then argues that wikis that mix wiki pages and RDF data are perfectly equipped to meet those requirements. *How* they can meet those requirements is illustrated by taking a closer look at the semantic wiki Hyena [2].

2 The survey

The survey has intentionally been relatively simple. For example, it was probably not representative for all potential wiki users, because the participants (a total of 23) were chosen in an ad-hoc fashion by announcing the survey to the author's colleagues[1] and to a mailing list about semantic wikis. But the results are still interesting and point out several possible trends for wikis.

[1] None of them are experts in the semantic web or wikis.

The survey participants answered in percentages indicating how important a given fact was to them or how often they performed a given activity. The reported percentages are averages of these answers. Note that the answers do not necessarily apply to a single wiki; many participants use several wikis. The following sections present groups of questions and observations that the author derived from the answers.

2.1 What is a wiki used for? What is its content?

Collecting data or knowledge	91%
Coordination, planning, project management	56.75%
Web site, light-weight content-managment system	54.5%
Document creation (and later publishing)	48.75%
Discussions, forum	42%
Brainstorming, (possibly shared) whiteboard	39.75%
Weblog, relatively small journal-style entries	16%

"What is the purpose of your wiki? What do you use it for?"

The content of traditional wikis is just text. But what users care about is the data and knowledge contained in the text—as expressed by the 91% ranking of "collecting data or knowledge".

Text	68.25%
Data	55.75%
Knowledge	55.75%

"What makes up the content of the wiki?"

In the survey, "text" was explained as feeling more like a word document, possibly being a collection of notes. "Data" are lists, tables, forms, etc.—things one might keep in a spreadsheet or a database. "Knowledge" is similar to data, but the focus is on collecting facts (true statements) and on specifying these facts as precisely as possible. Thus, the numbers above confirm what all the tables and lists in traditional wikis already suggested: In addition to text (semi-structured data, if you will), structured data and knowledge play an important role when it comes to wiki content. Naturally, structured information could be more flexibly processed if it were explicitly stored and had dedicated editors. For example, spreadsheets handle tabular data well, so it would be nice if one could embed little spreadsheets inside a wiki page. Note that the survey results do not indicate that wikis should become pure databases.Rather, being able to mix text and data is what seems to make wikis attractive.

2.2 Who uses the wiki?

Several collaborators, all reading and writing	60.25%
Personal use, a single person	50%
Few editors, many readers	46.5%

At heart, wikis can be considered groupware. Still, having the wiki information available anywhere and the flexibility in structuring information, makes wikis good personal information managers: Survey participants attributed an average importance of 50% to this task.

2.3 Current and future wiki features

Information roaming: the wiki information is available online.	78.5%
Collaboration: share and jointly edit information.	77.25%
Linking: relate and collate pieces of information.	72.75%
Publishing: disseminate information.	67%

"What core aspects of (traditional) wikis are you interested in?"

Version control (editing history, who edited what, unlimited undo, etc.)	78.5%
File upload and management	69.25%
Wiki page meta-data (annotations and labels describing the content of the page)	58%
WYSIWYG text editor	56.75%
Generate a PDF file from a wiki page	52.25%
Diagrams (UML, mind maps, organizational charts, etc.)	48.75%
Finer-grained wiki pages	47.75%
Outliners (edit indented lists such as tables of contents)	46.5%
Live collaborative editing (all editors work on the same copy of the document, changes show up immediately)	46.5%
Spreadsheets (with calculation)	46.5%
Discussions (forums)	41%
Offline editing, synchronization	37.5%
Calendars	37.5%
Form-based data entry (similar to MS Access)	37.5%
Blogs	25%

"What (actual or hypothetical) features are important to you?"

The author thinks that while an offline mode has been ranked relatively low, it is still essential for next-generation wikis. Otherwise, information will not be truly available everywhere (1st table); especially for personal information management (Sect. 2.2), one will need to access it without online connectivity.

2.4 Wiki alternatives

The following is a list of web applications that the survey participants use as alternatives to wikis for some tasks: (1) BackPack, (2) Blogger, (3) del.icio.us, (4) Facebook, (5) Flickr, (6) Google Calendar, (7) Google Docs, (8) iusethis, (9) Online Contacts, (10) Trac, (11) Wordpress, (12) WikipediaReview.com.

Interestingly, the majority (all except 1, 4, 11) of these web applications is very task-specific. Accordingly, Sect. 2.3 indicates that users would like to see more task-specific editing support (including WYSIWYG text editors) in wikis. The difficulty is to do so without significantly raising the learning curve.

3 Hyena

Hyena is an RDF publishing and editing system that comes in two components: A desktop application (Eclipse plugin, Fig. 1) and a web application (Java Servlet, Fig. 2) for online editing.

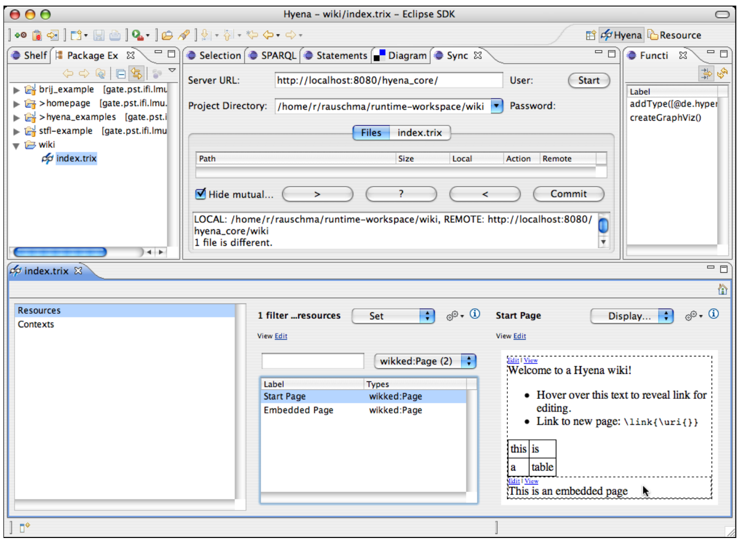
1. Storing and editing data: RDF is used as universal data storage. Hyena supports a variety of data encoded in RDF resources and has specialized graphical editors for them. Working with RDF generates *presentation data*: Lists of resources returned by a query, bookmarked locations, etc. This data can be edited in a similar fashion to RDF resources and saved (*manifested*) as RDF data.
2. Integrating pages and data: Wiki pages are also stored as RDF resources. They an link to external data or embed it. Similar to data-specific editors, embedding is supported by data-specific *embedders* (translators from the data to the abstract wiki syntax). All of a page's references (links, embeddings, etc.) to RDF resources are made explicit in RDF. This prevents stale links and allows one to track referers.
3. Meta-data for pages: Every page being an RDF resource, it can be annotated with RDF. This meta-data can be referenced in a query whose results can be manifested and embedded (as a table, as a sequence of embeddings, via templates). Thus, pages and data can be collated and presented in many ways.
4. Online and offline availability, collaborative editing: Both the desktop application and the web application manage web sites as *projects*, directory trees with files. This includes images, shared files and RDF data. One can synchronize projects between the desktop and the web. For RDF data, synchronization granularity is resources, otherwise it is files. Thus data can be published to a web server, but also edited offline. Furthermore, projects are easy to back up (which was one of the explicitly mentioned wishes in the survey).

The requirement of integrating text and data (Sect. 2.1) is fulfilled by items (1) and (2). The desired features "page meta-data" and "offline editing" (Sect. 2.3) are provided by items (3) and (4). Task specific editing (Sect. 2.4) is explained in item (1). For more detailed information on Hyena, consult [2]

Acknowledgments Thanks to Malte Kiesel, Andreas Schroeder, Philip Mayer, and Hubert Baumeister for their feedback on the survey questions.

References

1. J. J. Garrett. Ajax: A new approach to web applications. `http://www.adaptivepath.com/publications/essays/archives/000385.php`, 2005.
2. A. Rauschmayer. Wikifying an RDF editor. 2007. Submitted for publication.



Server URL:	http://localhost:8080/hyena_core/	User:
Project Directory:	/home/r/rauschma/runtime-workspace/wiki	Password:

```
LOCAL: /home/r/rauschma/runtime-workspace/wiki, REMOTE: http://localhost:8080/
hyena_core/wiki
1 file is different.
```

Label	Types
Start Page	wikked:Page
Embedded Page	wikked:Page

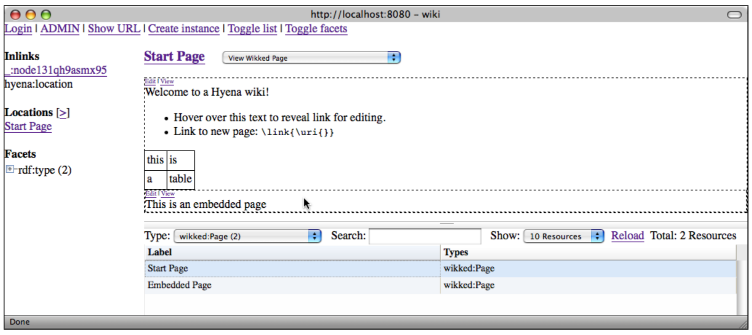
Welcome to a Hyena wiki!

- Hover over this text to reveal link for editing.
- Link to new page: `\link{\uri{}}`

this	is
a	table

This is an embedded page

Fig. 1. The Eclipse frontend for Hyena. The bottom half shows three editing panes that have been populated from left to right and thus make the history of editing visible.The top left is one of the standard Eclipse file explorers. One can see that the file `index.trix` that is being edited below belongs to project `wiki`. The top center hosts a variety of views that supplement the editor at the bottom. Currently visible is a view for synchronizing Eclipse projects with a web server. The top right contains a view that shows what commands can be used in the current context.



Welcome to a Hyena wiki!

- Hover over this text to reveal link for editing.
- Link to new page: `\link{\uri{}}`

this	is
a	table

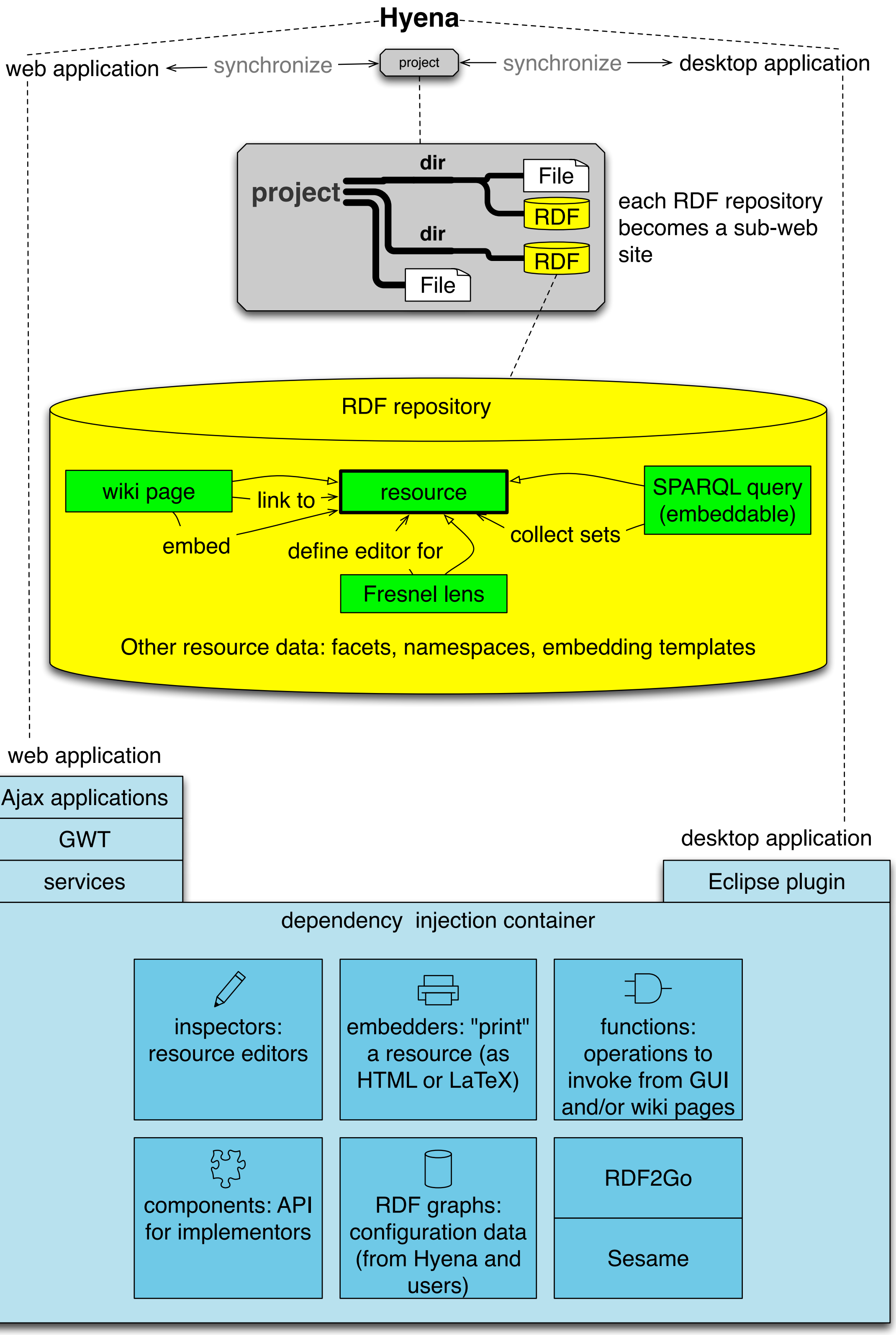
This is an embedded page

Label	Types
Start Page	wikked:Page
Embedded Page	wikked:Page

Fig. 2. The same project `wiki` that has been edited with Eclipse in Fig. 1 is displayed here as a web application in Firefox. The bottom right half shows the *resource list*, a subset of all available resources that is determined by filter criteria (such as "show only wiki pages", which is currently active). Resource `Start Page` has been selected and is visible in the top right. To the left, there is a toolbar that shows inlinks (resources that point to the currently selected resource); a list of special locations (the first of which is shown by default when starting the web application); and *facets*, a subset of all property key-value pairs among the resources in the resource list. Facets are grouped by key. One can hide the facets and the resource list so that Hyena/Web feels more like a web site.

RDF Editor Hyena

hypergraphs.de


Hyena
web application
synchronize
project
synchronize
desktop application
project
dir
File
RDF
dir
RDF
File
each RDF repository becomes a sub-web site
RDF repository
wiki page
link to
resource
SPARQL query (embeddable)
embed
define editor for
collect sets
Fresnel lens
Other resource data: facets, namespaces, embedding templates
web application
Ajax applications
GWT
services
desktop application
Eclipse plugin
dependency injection container
inspectors: resource editors
embedders: "print" a resource (as HTML or LaTeX)
functions: operations to invoke from GUI and/or wiki pages
components: API for implementors
RDF graphs: configuration data (from Hyena and users)
RDF2Go
Sesame

Data organized as projects

Application constructs encoded as resources

Architecture

Editing scenarios Tagged: BibTeX, bookmarks, list of favorite films, notes, Java code references

Other: compound document, list of friends (via FOAF), travel checklist (outline)

Axel Rauschmayer, LMU München (Germany)
Axel.Rauschmayer@ifi.lmu.de
www.pst.ifi.lmu.de/~rauschma

Integrating a Wiki in an Ontology Driven Web Site: Approach, Architecture and Application in the Archaeological Domain

Andrea Bonomi, Alessandro Mosca, Matteo Palmonari, Giuseppe Vizzari

Department of Computer Science, Systems and Communication (DISCo)
University of Milan - Bicocca
{andrea.bonomi, ale.m, matteo.palmonari, giuseppe.vizzari}@disco.unimib.it

Abstract. This paper describes an approach to the design and implementation of ontology driven dynamic web sites combining ontologies and wiki technologies. The core of the architectural solution proposed is completely based on ontologies rather than on more traditional forms of persistent data storage facilities, such as relational databases. This approach provides a flexible support to the design and implementation of web portals in which navigation schemes are not entirely predetermined but are instead influenced by actual relationships among the contents of the ontology, that are used to generate web pages as well as hyperlinks. A wiki technology is integrated with this approach in order to create pages not directly derived by elements of the ontology, but also to enrich the textual contents with suitable formatting, images and hyperlinks. The application of this approach to the realization of a web portal is also described; the portal is devoted to enable a number of scientific communities to share archaeological knowledge about the Silk Road domain.

1 Introduction

The introduction of technologies enabling the development of data driven web sites represented an extremely important step in the process that lead the initial version of the Web to its current state of pervasive diffusion and impressive growth rate. Data driven web sites are able to generate web contents and pages according to the information persistently stored in a suitable module, such as a relational Data Base Management System. This represents the last tier of an architecture encompassing a middle tier that is able to interpret suitable templates for web pages that are instantiated according to actual data stored in the data tier, and that are eventually passed to the first tier responsible of the visualization of the page (i.e. the client tier, a common web browser). Traditionally these templates have been realized by integrating modules developed in common programming languages with a web application (e.g. Common Gateway Interface or Java Servlet technologies), or embedding "programming languages–like" control structures and abstractions in traditional web pages through scripting languages (e.g. PHP, ASP, JSP). Some of the most relevant abstractions related to a data driven web site, and determining its organization and function are thus implemented and expressed in terms of a database logical schema and specific programming language control structures embedded in page templates.

Even if this approach surely represents a huge improvement with respect to static web sites, that are essentially repositories of HTML documents and embedded multimedia contents, a set of different research efforts have been carried out in an attempt to supply higher level abstractions to support the design and implementation of web based advanced information systems and applications. Some of these approaches, for instance [1, 2], are based on traditional data conceptual models for site contents and extend the scope of the modelling activity to aspects of relevance in the web context, such as navigation and presentation. In this vein, this paper presents an approach that provides instead the adoption of an *ontological* rather that data tier, building on experiences and results of research in the Semantic Web [3] area to provide new abstractions and instruments for the structuring and management of information supporting dynamic web pages composition. In particular, the basic idea is to exploit an explicit and formal conceptualization of concepts related to the web site domain, as well as specific aspects related to web sites in general, to structure data and information required to generate contents, to specify the navigation among them and to generate an effective presentation. The ontological approach provides a uniform and expressive framework for the representation and management of these different aspects. In particular, the ontology can encompass documents, and one particular type of document can be represented by a *wikipage*. This allows endow this ontology driven approach to the definition, design and realization of web sites with a more traditional and established form of definition of contents of web pages.

The following section will elaborate the research context in which this work is set, briefly introducing relevant related works, while Section 3 introduces the architecture and the various functionalities offered by NaVEditOW [4], the framework on which this approach is based. A case study in which the approach has been applied for the development of a portal organizing archaeological information and documents will then be introduced. Conclusions and future developments will end the paper.

2 Ontology Driven Web Sites and Wikis

In his proposal for a global hypertext [5], Tim Berners-Lee proposed a “gateway program” to generate hypertext view of existing data source. He has imagined a simply generic gateway with a limited, perhaps read-only, access on a database that allow to display it as a hypertext and navigate through the data.

Within a short period, the Internet and World Wide Web have become ubiquitous and today Internet is full of data-driven Web sites: yellow pages, e-Commerce sites, digital libraries are only the most common examples. But also, forum, blog, wiki, video and photo sharing website, hotel booking, online auction website, online community, Web-based email client and Web mapping service are all data-driven Web sites.

A data-driven Web site is much easier to maintain than a static Web site: most content changes require no change to the pages. Instead, changes are made to the data source and this source could be enterprise or organization database. Sharing a common data source, the Web application can be easily integrated within information system.

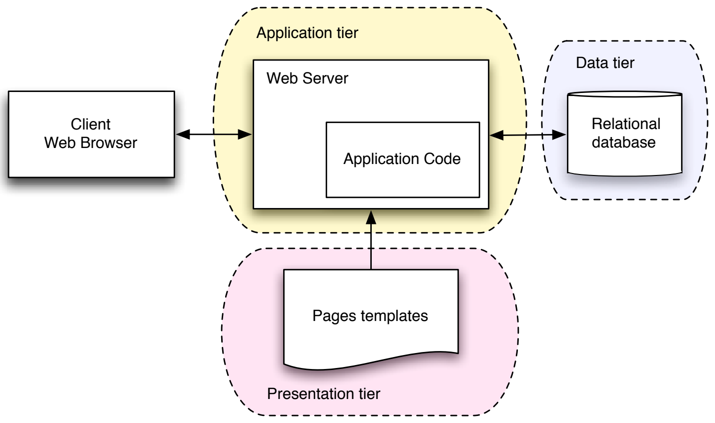


Fig. 1. On the top schema, the architecture of a traditional data-driven Web site.

So, for example, in a manufacture company when a new product is added to the enterprise database, it could be automatically displayed on the corporate Web site products catalog.

One of the most common architecture used in developing of dynamic data driven web sites, is shown in Figure 1. In particular, the persistent data storage is generally delegated to a relational database management system. The web server generally hosts dynamic pages including server side scripts necessary to query the data tier and collect the information required to compose pages related to the contents of the database. These dynamic pages represent a sort of *template* for the web pages that must be generated according to the stored data, specifying different aspects ranging from the queries that must be submitted to the database to retrieve them, to the kind of links that must be created to support the navigation inside the data.

An evolution of this approach is the adoption of the the MVC (Model View Controller) [6] architectural pattern. In a web based MVC application, the model is the domain-specific representation of the application information, the view layer is responsible to renders the model into user interface element (HTML pages) and the Controller processes and responds to the users actions and can invoke data changes.

Generally, the data is stored in a relational database and the model is represented by a database schema. In our opinion, it is difficult to represent all the relations that could be present in a complex domain (such the archaeology domain) with a database schema. In many cases, is difficult to translate some aspects of a conceptual schema (such the generalization) into the relational model. Some kind of extra-relational constraint are not representable within the database schema and requests database store procedure or external application code. There are also problems to manage the database schema, for example, the is no way to check the schema consistency.

The proposed approach provides the adoption of an ontology [7], rather than a relational database, as data layer (a schema of such architecture is shown in Figure 1) and to use a Wiki engine in order to simplified content authoring and management func-

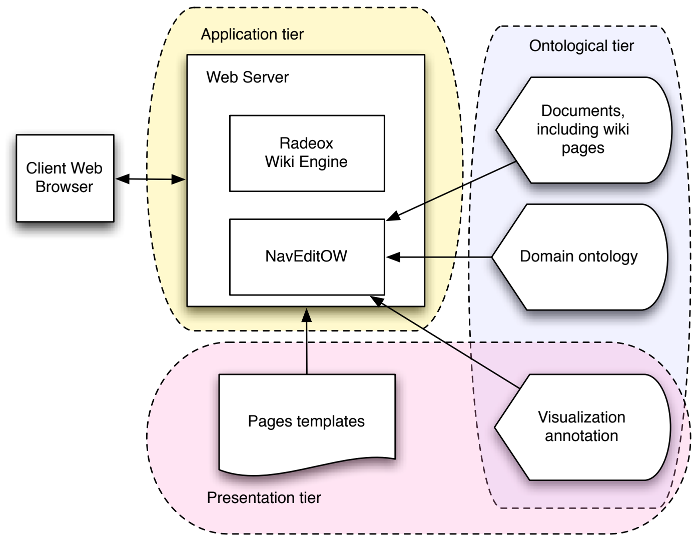


Fig. 2. The overall architecture of the proposed system, integrating a wiki engine for rendering specific documents stored in the ontological tier.

tionality. In particular, we employed NaVEditOW [4], a system for navigating, editing and querying ontologies through the web, as a means to access and manage the ontology, and we integrate the Wiki technology through the adoption of Radeox [8], that is an Open Source Wiki engine written in the Java language. An interesting feature of this engine is that it could be easy extended with custom macros that can support, for instance, the inclusion in a wiki page of a table of all instances of a given class. The overall resulting architecture is shown in Figure 2.

The main motivations of this architectural modification are related to a more *comprehensive exploitation* of the explicit relationships among the concepts described in the ontology, and a *simpler integration* of this kind of architecture with instruments and systems developed in the Semantic Web context.

Other approaches of ontology-driven Web site architectures can be found in the literature: in [9], it is described an application, OntoWeaver, that uses an ontology to provide a comprehensive support for the design and management of data-driven Web sites. The described approach uses site ontologies to enable a declarative representation of all the aspects of a Web sites: in particular, a domain ontology is used as an abstract of the back-end data sources and a user ontology models information about the Web site users. In [10] it is described a similar approach in which data from various sources are converted into RDF-annotated format (based on a domain ontology), composed together and used to generate a browsable Web site.

A large number of approaches for combining Semantic Web and Wiki technologies are currently under development; some relevant examples are Makna [11], IkeWiki [12], Rhizome [13]) and Semantic MediaWiki [14]. The above cited approaches and the related systems are all examples of Wikis expanded to encapsulate and exploit a well–defined semantics to enhance their functionalities. It is not the aim of this section to introduce a structured and comprehensive discussion of these approaches and systems (an interesting discussion on this line of work can be found in [15]), but it is important to note that the described approach follows a totally different line of work: we are more focused on supporting a collaborative effort towards the definition of domain ontologies and their exploitation in web based systems than in supporting the enhancement of wikis through the usage of semantic web technologies. The NavEditOW system and this effort can be intended as an attempt to bring part of the spirit of wiki technologies and the social aspects of the Web 2.0 in the semantic web context. The remainder of the paper will introduce the NavEditOW system and its application to the realization of a web portal supporting a community of archaeologists working on Cultural Heritage of Central Asia.

3 NavEditOW

In this section we present a system for web based navigation, querying and updating of ontological KBs. The presented software allows exploring the concepts and their relational dependencies as well as the instances by means of hyper-links; moreover, it provides a front-end to query the repository with the SPARQL[1] query language.

NavEditOW is an environment for navigating, querying and A-Box[2] editing of OWL[3] (Web Ontology Language) ontologies through a web-based interface.

With respect to ontology *navigation*, since individuals play a fundamental source of knowledge for people accessing an ontology, A-Box navigation should be supported. From this perspective, it is important to support not only navigation of concept hierarchies defined by *isA* relations, but also other forms of ordering on the individuals domain. As a first example, locations can be linked through a *partOf* relation and it should be possible to group locations under the location of which they are all subparts (e.g. browsing all countries of which Europe is composed of starting from Europe); as a second example consider a number of historical periods ordered according to a relations such as *followedBy*: it should be possible to exploit this relation to sort such individuals from the first to the last one.

With respect to *editing*, although T-box[4] maintenance require a certain knowledge about ontological formalisms, A-Box editing should be supported taking into account: (i) cardinality and range restrictions defined in the T-Box need to be respected; (ii) ranges of properties and individuals stored in the ontologies can be also exploited to drive and suggest instance update. Moreover, contextual editing, that is, the editing of the A-box while browsing the ontology, should be supported.

[1] SPARQL (SPARQL Protocol and RDF Query Language) is an RDF query language standardized by RDF Data Access Working Group of the World Wide Web Consortium. For more information, see http://www.w3.org/TR/rdf-sparql-query/

[2] The A-Box is the “assertion component” of a knowledge base.

[3] http://www.w3.org/TR/owl-features/

[4] The T-Box is the “terminological component” of a knowledge base.

Although end-users may not be familiar with *query* languages, the possibility to perform expressive queries should be supported. From the one hand a language as much as similar to well-known query languages for relational databases language should be preferred. On the other hand, interfaces enabling non expert users querying the ontology should be developed (e.g. query forms).

In the following paragraphs, we presents more details about each of these tree basic functionalities and the application architecture.

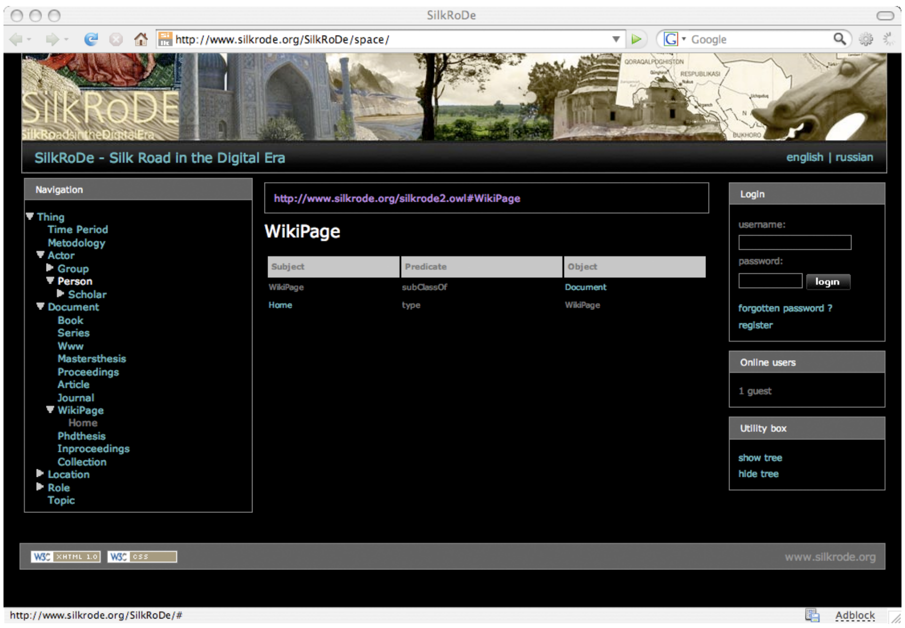
3.1 Navigation

With the ontology navigation interface the users can view ontology individuals and their properties and browse properties via hyperlinks. Browsing the ontology is essential for the user in order to explore the available information and it also helps non-expert users to refine their search requirements, when they start with no specific requirement in mind [16]. The hierarchical organization of the different concepts and individuals of the ontology is graphically represented as a dynamic tree. The aim of the navigation tree is to explore the ontology, view classes and instances, discover the relation between them. The tree does not represent only the a hierarchy of classes connected with *isA* binary relations (like the navigation tree of Protégé), but represents also also tree-like connections of individuals for domain dependent classes of properties (e.g. *partOf*, *isLocatedIn*, and so on). From a formal point of view, ontological relations supporting tree-like visualization (tree-like properties) are those represented as properties not symmetric and whose inverse is functional (therefore identifying directed acyclic graphs). These properties link directly an individual with its "father" and are particularly relevant with respect to mereological relations (e.g. *partOf*, *composedOf*), and to relations defining hierarchical spatial and temporal structures (e.g. representing the unfolding of historical periods). Another kind of relations exploited for the visualization are relations defining total orders on individuals (e.g. *isFollowedBy*).

The root of the navigation tree is the OWL class *Thing*, and the rest of the tree is organized as follows: under the root node, there are the top-level classes (i.e. direct subclasses of Thing); each class can be expanded to show its subclass hierarchy and its individual members; individual-to-individual tree connections are defined according to a number of selected tree-like properties (e.g. *partOf*); finally if total order relations are selected, they are exploited to order individuals within a given level of the tree. In order to distinguish between classes and individuals, the former are represented in petrol blue while the latter are shown in shocking pink. An example of a navigation tree is presented in the Figure 3. In this example *Geographical Region* is a class and *Cental Asia*, *Uzbekistan*, etc... are its instances. This individuals, in turn, are connected each other by the *partOf_directly* property.

3.2 Editing

The application allows the users to create, edit and remove individuals of the ontology, their properties and, in particular, their labels. In fact, To ensure multi-languages support, it's possible to define several labels in different languages for every individual.



http://www.silkrode.org/silkrode2.owl#WikiPage

WikiPage

Subject	Predicate	Object
WikiPage	subClassOf	Document
Home	type	WikiPage

Fig. 3. A screenshot of the NavEditOW system in the archaeological case study.

The properties of each classes are defined in the T-Box. Two types of properties are distinguished: *object property* is a binary relation between two individuals and *datatype property* is a binary relation between an individual and a literal (a primitive type, like string or number). Cardinality and range restrictions for properties are used to support users while editing. For example, in an archeological ontology, the class *TypologyOfArchaeologicalObject* has the property *builtOf*. This property has no cardinality restriction (so it can have zero, one ore more values) but *Material* is specified as range (co-domain). For instance, *Sword* is an instance of *TypologyOfArchaeologicalObject* and has the property *builtOf Metal*, where *Metal* is an instance of *Material*.

There are a datatype and an object editor in the framework: the datatype editor allows editing a literal values, displayed as a text input box, object allows defining the property values presenting the user a tree for selecting the values; the individuals displayed in the tree are only those that are valid for the property range.

Literal values can be not only plains strings but also Wiki text. This allows the users to insert long formatted text with link in the generated pages. For example, we can have the 'Uzbekistan' individual in our ontology. 'Uzbekistan', as an instance of Country, has some properties such population, capital city, part-of, currency, history, foreign relations, etc. Same (e.g. population) are plain literals (numbers, dates, strings), others are references to other ontology instances (e.g. the filer of the property 'capital city' is Tashkent which is an instance of the class City). History of foreign relations are Wiki Text, so they can be formatted, contains images an hyperlinks. Wiki Text can also contains 'macro' that allows to include in the text information from the ontology or results of SPARQL queries.

NavEditOW adopts an editing policy. Knowledge engineers interacting with domain experts, are supposed to create the ontology and edit the Tbox with standard design-time editing tools such as Protégé [17]. They are supposed to test its quality with reasoners such as Racer [18] or Fact++ [19] in order to check if the Tbox (i) is consistent and (ii) it captures the intended meaning (by concept hierarchy inference). End-users can edit the Abox; they can insert new instances, asserting that they are member of a concept of the ontology, and assert relation statements; this means, that they can edit the textual descriptions contained in the ontology. The assumption behind this editing policy is that end users are not familiar with formal semantics. First, learning to use concept constructor and ontology axioms is difficult for end users, if one goes beyond simple Tbox updates, such as the insertion of a new concept as subconcept of a concept in the ontology. Second, every axiom has logical implications that are difficult to control, especially for people non expert in formal semantics.

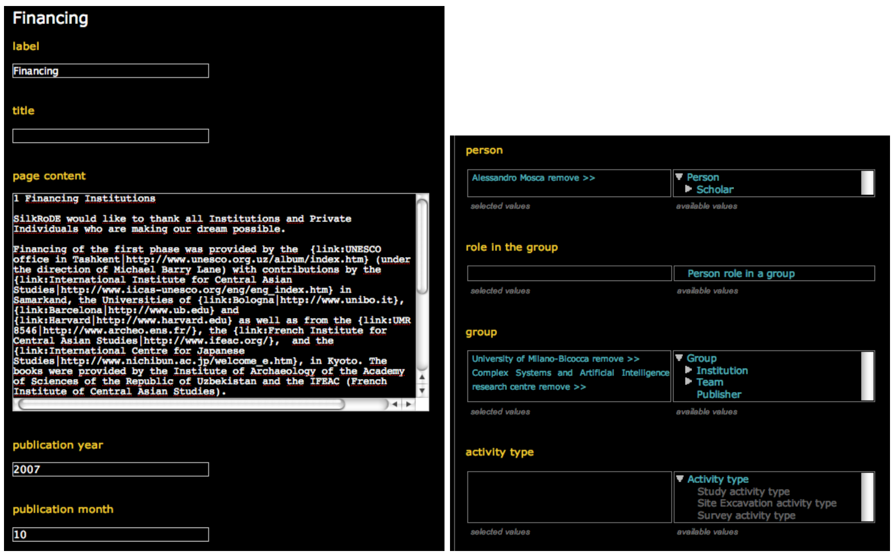
3.3 Querying

The first implemented query interface is the SPARQL query form in which users can write query in the SPARQL language, display results in paginated tabular form and navigate through results via hyperlinks. This interfaces is very flexible because the users can write arbitrary queries but is not suitable for end users. Another kind of query interfaces is based on a predefined set of queries. Every predefined queries is composed of a description in natural language, a SPARQL query with eventually free parameters and a list of parameters. Every parameters have a label, a type and eventually a restriction on the valid values (e.g. a parameter can be filled only with instances of a specific class). For this interface, users can select a query by its description, fill the query parameters and execute it. The results are presented as the results of the other query form. The queries can be inserted in the Wiki Text through a macro, for example the following macro shows in the resulting page all the provinces of the Uzbekistan:

```
{sparql | SELECT ?x
WHERE { ?x silkrode:partOf silkrode:Uzbekistan .
?x rdf:type silkrode:Province }}
```

3.4 Application architecture

From an architectural point of view, the functionalities (ontology editing, navigation and querying) of the user interfaces are based on the Application API, as shown in Figure 5. The main purpose of this API is to support the manipulation and querying of the ontology through the standard SPARQL query language and through a set of specific adapters, shielding the user from the underlying semantic framework. A plug-in interface, in fact, makes the application independent from the adopted specific semantic framework (as long as SPARQL is supported). A different adapter for every semantic


Financing

label

Financing

title

page content

```
1 Financing Institutions

SilkRoDE would like to thank all Institutions and Private
Individuals who are making our dream possible.

Financing of the first phase was provided by the  {link:UNESCO
office in Tashkent|http://www.unesco.org.uz/album/index.htm} (under
the direction of Michael Barry Lane) with contributions by the
{link:International Institute for Central Asian
Studies|http://www.iicas-unesco.org/eng/eng_index.htm} in
Samarkand, the Universities of {link:Bologna|http://www.unibo.it},
{link:Barcelona|http://www.ub.edu} and
{link:Harvard|http://www.harvard.edu} as well as from the {link:UMR
8546|http://www.archeo.ens.fr/}, the {link:French Institute for
Central Asian Studies|http://www.ifeac.org/},  and the
{link:International Centre for Japanese
Studies|http://www.nichibun.ac.jp/welcome_e.htm}, in Kyoto. The
books were provided by the Institute of Archaeology of the Academy
of Sciences of the Republic of Uzbekistan and the IFEAC (French
Institute of Central Asian Studies).
```

publication year

2007

publication month

10

person

selected values	available values
Alessandro Mosca remove >>	▼ Person ▶ Scholar

role in the group

selected values	available values
	Person role in a group

group

selected values	available values
University of Milano-Bicocca remove >> Complex Systems and Artificial Intelligence research centre remove >>	▼ Group ▶ Institution ▶ Team Publisher

activity type

selected values	available values
	▼ Activity type Study activity type Site Excavation activity type Survey activity type

Fig. 4. Example of datatype properties editing with Wiki Text (on the left) and object properties editing (on the right).

framework is needed because SPARQL is only a query language and does not offers any data manipulation statements (e.g. INSERT, UPDATE, DELETE).

The adapter API supports the manipulation and query of the RDF graphs in two different ways: *frame-centric* and *statement-centric*. The former view is similar to the object-oriented paradigm: every resource is viewed as an object and properties as attribute. This view is used for ontology navigation and resource manipulation. *Statement-centric* is a lower level view in which the graph is represented as a set of triples. Each triple contains three components: subject, predicate and object.

Currently two semantic framework adapters have been implemented: the first one is a wrapper for the Jena Semantic Web Toolkit, the other for the Sesame Framework. The former is an open-source Semantic Web Toolkit[5] aimed at supporting the development of applications that use the Semantic Web information models an languages [20]. We have initially adopted this framework since it matched our requirements, it is widely used within the Semantic Web research community and well documented. This first adapter implementation works well with small ontologies, but fails with larger ones since Jena is not suitable to manage an huge amount of data (a performance evaluation of several frameworks suitable for large OWL ontologies is presented in [21]). For this reason, we choose to develop a new adapter for the Sesame Framework[6] [22]. Sesame provides a number of functionalities for handling (querying and manipulating) RDF graphs. It also supports various types of storage facilities and inference mechanisms.

[5] http://jena.sourceforge.net/

[6] http://www.openrdf.org

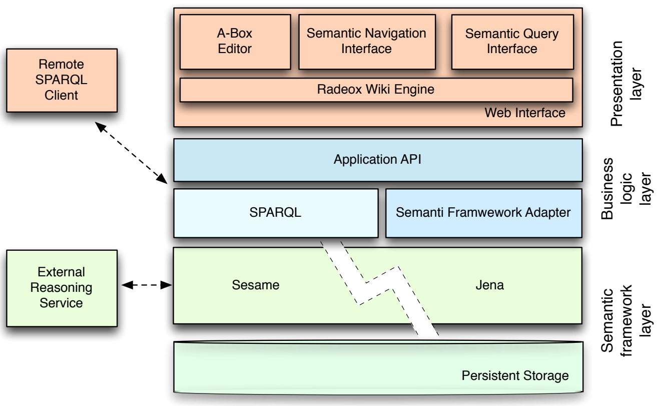


Fig. 5. The NavEditOW architecture

The default implementation supports inferencing and querying on RDF Schema but lacks a specific support for OWL.

4 The SilkRoDE Case Study

SilkRoDE (Silk Roads in the Digital Era) is a project that aims to collect, structure and diffuse all knowledge concerning the Cultural Heritage of Central Asia, including not only archaeological material, sites and historical monuments but also data from fields such as geography, sociology and ethnography. It is an open and evolutive project, which functions as an intelligent network linking all interested institutions, research groups and scholars, that worked and working in Central Asia.

As a first step, SilkRoDE aims to create a Wiki, used on a daily basis by all specialists of Central Asia, interested members of the general public and those involved in Cultural Management. The creation of this Wiki will be possible thanks to a collaboration between specialists from the Humanities and from Computer Sciences in a very close multidisciplinary context. One of the keys to the success of the SilkRoDE project is the decision of all participating scholars, institutions and research groups to work together as Equal Partners. This does not mean that all resources are simply pooled together but that each resource is clearly associated to the authors and funding agencies that enabled its creation. SilkRoDE thus aims to be a Network rather than a new institution. In this perspective, the NavEditOW system represents the general platform adopted by the Project to organize and manage the various different contents of SilkRoDE Wiki. The main aim is to share information and knowledge, and all partners should thus be enabled to employ the system to publish, connect personal data and information, and to identify other groups or Institutions working or interested in same themes. This need for a collaborative and collective participation in data and information collection phases lead us to choose a Wiki–based approach to develop the SilkRoDE portal.

The introduced approach and the NavEditOW system were adopted to represent and organize basic information about institution, research groups and scholars having active researches in Central Asia, as well as information about relevant bibliography and important cultural and archaeological sites. The platform will be integrated with webGIS: the possible connection between specific entities of the SilkRoDE ontology and georeferred entities stored in a GIS will support the visualization of spatial position and distribution of various entities in dynamically generated maps.

The ontological approach provided the required expressiveness and flexibility even in these starting phases of the project, in particular in order to support rich forms of navigation among stored contents. In particular, this approach provided the possibility of representing and managing relationships like "is-a" and "part-of" without flattening the related entities in a single table or splitting them in different tables, as would have been necessary adopting a traditional relational database. For instance institutions, research groups and individual scholars are all actors of the SilkRoDE ontology, in other words, they are individuals belonging to classes that are related to the *Actor* class by an "is-a" relation. It is now possible to define a generic relationship binding an archaeological site to an instance of the *Actor* class or one of its subclasses, without the need to define different relationships. This flexibility in defining and establishing relationships among individuals will support further types of analysis aimed, for instance, at identifying possible connections among actors that are working on similar or related research issues, or geographic areas, or adopting similar methodologies.

5 Conclusions and Future Developments

The paper has described an ontology driven approach to the modeling, design and implementation of dynamic web sites. In particular, we aimed at simplifying the realization of web based systems that exploit and give access to a shared ontology, but these systems should also look like traditional web sites and support simple forms of navigation. To this aim, we endowed the system with a wiki engine and we included documents, and in particular wiki pages, in the ontological tier, to support a simple form of editing of pages that give the site an ordinary structure and appearance, that can be enhanced by means of the exploitation of the underlying domain ontology.

The motivations of this effort, as well as related work and the research context were introduced, and the NavEditOW framework was described, in terms of provided functionalities and architecture. A case study providing the application of the introduced approach and framework was also presented.

Future works are mainly aimed at extending the range of the represented and managed concepts, with particular reference to the various topics that can be used to characterize relevant scientific publications, in an effort similar to the one described in [23]. Moreover, in the medium term, the project will consider the possibility to realize specific wrappers able to to export contents complying to the CIDOC Conceptual Reference Model [24], so as to achieve a high level of interoperability with this relevant standard for cultural heritage information organization.

With respect to the editing policy, it will be tested the possibility to enable end-users to insert new concepts and the respective subclass-relations. Moreover the more extended use of reasoning capabilities will be investigated.

References

1. Atzeni, P., Nostro, P.D.: T-Araneus: Management of Temporal Data-Intensive Web Sites. In Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., Ferrari, E., eds.: EDBT. Volume 2992 of Lecture Notes in Computer Science., Springer (2004) 862–864
2. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): a Modeling Language for Designing Web Sites. Computer Networks **33**(1-6) (2000) 137–157
3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American **284**(5) (2005) 34–43
4. Bonomi, A., Mosca, A., Palmonari, M., Vizzari, G.: NavEditOW - a System for Navigating, Editing and Querying Ontologies through the Web. In Apolloni, B., Howlett, R.J., Jain, L.C., eds.: KES (3). Volume 4694 of Lecture Notes in Computer Science., Springer (2007) 686–694
5. Berners-Lee, T.: Information Management: a Proposal (1989)
6. Leff, A., Rayfield, J.T.: Web-Application Development Using the Model/View/Controller Design Pattern. In: EDOC, IEEE Computer Society (2001) 118–127
7. Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human-Computer Studies **43**(5-6) (1995) 907–928
8. Jugel, M.L., Schmidt, S.J.: The Radeox Wiki Render Engine. In Riehle, D., Noble, J., eds.: Int. Sym. Wikis, ACM (2006) 33–36
9. Lei, Y., Motta, E., Domingue, J.: Modelling Data-Intensive Web Sites with Ontoweaver. In Grundspenkis, J., Kirikova, M., eds.: CAiSE Workshops (1), Faculty of Computer Science and Information Technology, Riga Technical University, Riga, Latvia (2004) 106–121
10. Jin, Y., Decker, S., Wiederhold, G.: Ontowebber: Model-Driven Ontology-Based Web Site Management. In Cruz, I.F., Decker, S., Euzenat, J., McGuinness, D.L., eds.: SWWS. (2001) 529–547
11. Dello, K., Simperl, E.P.B., Tolksdorf, R.: Creating and Using Semantic Web Information with Makna. In Völkel, M., Schaffert, S., eds.: SemWiki. Volume 206 of CEUR Workshop Proceedings., CEUR-WS.org (2006)
12. Schaffert, S.: IkeWiki: A Semantic Wiki for Collaborative Knowledge Management. In: WETICE, IEEE Computer Society (2006) 388–396
13. Souzis, A.: Building a Semantic Wiki. IEEE Intelligent Systems **20**(5) (2005) 87–91
14. Krötzsch, M., Vrandecic, D., Völkel, M.: Semantic MediaWiki. In Cruz, I.F., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L., eds.: International Semantic Web Conference. Volume 4273 of Lecture Notes in Computer Science., Springer (2006) 935–942
15. Krötzsch, M., Schaffert, S., Vrandecic, D.: Reasoning in Semantic Wikis. In Antoniou, G., Aßmann, U., Baroglio, C., Decker, S., Henze, N., Patranjan, P.L., Tolksdorf, R., eds.: Reasoning Web. Volume 4636 of Lecture Notes in Computer Science., Springer (2007) 310–329
16. Ram, S., Shankaranarayanan, G.: Modeling and Navigation of Large Information Spaces: a Semantics Based Approach. In: 32nd Annual Hawaii International Conference on System Sciences (HICSS-32), 5-8 January, 1999, Maui, Hawaii, Track 6: Modeling Technologies and Intelligent Systems., IEEE Computer Society (1999)

17. Gennari, J.H., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F., Tu, S.W.: The Evolution of Protégé: an Environment for Knowledge-Based Systems Development. Int. J. Hum.-Comput. Stud. **58**(1) (2003) 89–123
18. Haarslev, V., Möller, R.: Racer: a Core Inference Engine for the Semantic Web. In Sure, Y., Corcho, Ó., eds.: EON. Volume 87 of CEUR Workshop Proceedings., CEUR-WS.org (2003)
19. Tsarkov, D., Horrocks, I.: Fact++ Description Logic Reasoner: System Description. In Furbach, U., Shankar, N., eds.: IJCAR. Volume 4130 of Lecture Notes in Computer Science., Springer (2006) 292–297
20. McBride, B.: Jena: a Semantic Web Toolkit. IEEE Internet Computing **6**(6) (2002) 55–59
21. Guo, Y., Pan, Z., Heflin, J.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. In McIlraith, S.A., Plexousakis, D., van Harmelen, F., eds.: International Semantic Web Conference. Volume 3298 of Lecture Notes in Computer Science., Springer (2004) 274–288
22. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: an Architecture for Storing and Querying RDF Data and Schema Information. In Fensel, D., Hendler, J.A., Lieberman, H., Wahlster, W., eds.: Spinning the Semantic Web, MIT Press (2003) 197–222
23. Bonomi, A., Mantegari, G., Vizzari, G.: A Framework for Ontological Description of Archaeological Scientific Publications. In Tumarello, G., Bouquet, P., Signore, O., eds.: Semantic Web Applications and Perspectives 2006, Proceedings of SWAP 2006, the 3rd Italian Semantic Web Workshop, Pisa, Italy. Volume 201 of CEUR Workshop Proceedings. (2006)
24. Doerr, M.: The CIDOC CRM, an Ontological Approach to Schema Heterogeneity. In Kalfoglou, Y., Schorlemmer, W.M., Sheth, A.P., Staab, S., Uschold, M., eds.: Semantic Interoperability and Integration. Volume 04391 of Dagstuhl Seminar Proceedings., IBFI, Schloss Dagstuhl, Germany (2005)

Extending the Makna Semantic Wiki to support Workflows

Karsten Dello, Lyndon Nixon, and Robert Tolksdorf

Freie Universität Berlin, Institut für Informatik,
AG Netzbasierte Informationssysteme, Takustr. 9, D-14195 Berlin, Germany
{dello,nixon,tolk}@inf.fu-berlin.de,
http://www.ag-nbi.de/

Abstract. Semantic wikis combine the advantages introduced by the wiki principle with the potential of Semantic Web technologies. However, there is still a very limited support for coordination, collaboration and integration in current semantic wikis. In this paper, we present a solution for this through the integration of our Makna semantic wiki with a workflow system. The resulting implementation is presented and an example given how this integration leads to better coordination, collaboration and integration support.

1 Introduction

Wikis have become popular tools for collaborative information management rooted in the principles of editability and collaboration : each member of the community can provide his knowledge, revise incorrect information and of course benefit from access to everyone else's knowledge. Semantic wikis are a natural extension of this, allowing for knowledge to be provided in some formal manner, e.g. using RDF, to improve the search for and retrieval of knowledge in the wiki system.

In this paper, we respond to a significant shortcoming in current semantic wiki systems in that they provide limited support for coordination, collaboration and integration despite the presence of semantics which can be used to improve such aspects. Particularly for corporate semantic wikis, it can be important to follow specific workflows in the collaborative editing of some wiki article, e.g. that a certain department must provide its input before another department takes over working in the wiki.

We present our proposal for the limited coordination, collaboration and integration of semantic wikis: combining a semantic wiki and a workflow management system. We have designed a model integrating the semantic wiki's data model with the WfMC process definition reference model (*http://www.wfmc.org/standards/referencemodel.htm*). Following an introduction to the semantic wiki used in this work, Makna, we describe its integration with the workflow engine jBPM. We evaluate the implementation towards a workflow example and conclude by considering the contribution of this work.

2 Integrating a semantic wiki with a workflow engine

Makna was conceived as a Wiki-based tool for distributed knowledge engineering (*http://makna.ag-nbi.de*). It extends an existing Wiki engine with generic, easy-to-use ontology-driven components for collaboratively authoring, querying and browsing Semantic Web information. The architecture of Makna consists of the Wiki engine JSP-Wiki (*http://www.jspwiki.org/*), extended with several components for the manipulation of semantic data, and the underlying persistent storage mechanisms. A more detailed description of Makna and the user interfaces it supports can be found in [DPT06].

For this work we integrated the semantic wiki Makna and the workflow engine jBPM . The architecture of the distributed system with jBPM and Makna consists of a J2EE server, J2SE server(s), and a workstation and database servers that serve as persistence stores. In this paper we briefly consider some of the aspects of this integration which were implemented.

2.1 Semantic Workflow Annotation

It is necessary to reflect the workflow instances (e.g. tasks and processes) in the semantic model in order to support search for and enhanced presentation of tasks and processes. The insertion of workflow concepts into the semantic model at runtime has two prerequisites: first, a mechanism must be provided which assigns URIs to workflow instances; second, these URIs must be made accessible in jPDL process definitions.

With these prerequisites met, semantic decoration of workflow execution can be performed with standard jBPM procedures. While traversing a process graph the engine fires events – e.g. *process-start*, *process-end*, *task-start*, *task-end* and *task-assign* – which can be associated with custom actions. By associating these events with actions that perform SPARUL update queries based on URIs for workflow instances, the progress of a workflow can be reflected in the semantic model.

2.2 Semantic Flow Conditions

Facilitating semantically enhanced flow conditions in the process execution phase is desirable, because inference allows for transitions and control flow, based on the description of a workflow in the semantic model itself.

jPDL supports the association of transitions with boolean expressions which are based on workflow relevant data. The process execution continues via the first transition whose associated expression resolves to true, or via the default transition if none of the expressions resolves to true, though this behavior can be customized. Anyway, a straightforward approach is to query the semantic model with a SPARQL-ASK-Query, and store the result in a process variable which determines an expression that is associated with a transition. This procedure facilitates simple control of the process based on semantic model state.

2.3 Semantic Assignments

Strategies for the assignment of tasks to wiki users that build on semantic user and task descriptions should be supported by the system. Semantic user descriptions can be based on formalizations such as DOAC. Task descriptions are a bit more complicated because not only the general description of a task, which is valid for all task instances, but also details of the current execution of a particular instance might be relevant for assignments. The description of users and tasks can use common concepts, e.g. by sharing domain ontologies. The actual assignment of tasks to users is realized through a jBPM *AssignmentHandler* implementation which can be configured with a SPARQL query that references these concepts.

2.4 Semantic Search and Presentation of Workflow Individuals

The structured presentation of semantic resources has been a lacking feature in Makna. To deal with this problem we have added support for formating SPARQL XML responses with XSLT. This functionality is encapsulated as another JSPWiki-Plugin, which additionally has limited support for expressions that are resolved at rendering time (e.g. logged in user and page URL). Via an endpoint parameter remote triplestores are also supported. The plugin enables the structured presentation of workflow individuals such as tasks and process instances in the wiki. It can be called from the JSPWiki template level (invoked from a JSP) or from the wiki page level (invoked from wiki syntax). Another application of this plugin in the generation of lists such as semantically enhanced task lists. The tasks that have been assigned to the logged in user can be arranged by domain specific structures, thus enabling semantically enhanced task lists.

3 Evaluation

In this section we present an example workflow to illustrate improvement of coordination, collaboration and integration support in our system.

3.1 Example Workflow

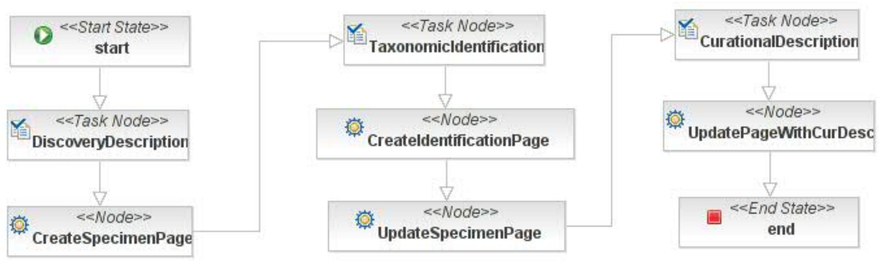


Fig. 1. Sample workflow for the processing of a new specimen. The three columns are associated with the process swimlanes *FieldWorkParticipant*, *Taxonomist* und *Curator*.

We have derived a couple of example workflows from a business model within the EDIT (European Distributed Institute of Taxonomy) project (*http://www.e-taxonomy.eu/*). Figure 1 shows a typical workflow: the processing of a specimen. It involves three actors: a field trip participant who describes the finding of the specimen, a taxonomist who performs the taxonomical identification and a curator who is responsible for the curational administration.

First, somebody in the role of a *field work participant* triggers the start of the process [1] and describes the discovery of a new specimen through an associated task form in the userconsole, whereby a new wiki page is created from a template. This template contains enhanced wiki syntax with placeholders for subjects of statements, which are replaced with corresponding user input. User input consists of literals (e.g. from textarea and input fields) as well as URIs of concepts and instances (e.g. from JSF *SelectItem* elements). Predicates used in the template are taken from existing ontologies. [2]

Notification by email is sent to somebody in the *taxonomist* swimlane to inform him/her that he/she is due to perform a taxonomic identification of this specimen. Again, the results are committed to the workflow engine through a task form. Upon completion a new page with the results of the taxonomic identification is created, and a typed link to it is inserted on the discovery page. A last action must be taken by somebody in the role of a *curator*. The selection of the actor is formalized as a SPARQL query and executed upon creation of the task instance. It is based on semantic description of curators' responsibilities and the results of the taxonomist's identification.

3.2 Improved Coordination, Collaboration and Integration

Even though this short sample workflow makes only use of one workflow pattern – the sequence – it can be used to show that coordination, collaboration and integration support has been improved. The combination with a workflow system has enabled the coordination of interactions within a wiki system. The creation of a new wiki page in the node *CreateIdentificationPage* has been combined with the modification of another page in the node *UpdateSpecimenPage* which adds a link to the newly created page, thus realizing a coordinated modification of two wiki resources.

The collaboration of the participating actors is described in the workflow, thus enabling a controlled yet versatile modification of a semantic wiki resource. The chronological dependencies between the activities of the actors in the process roles *FieldWorkParticipant*, *Taxonomist* und *Curator* is reflected in the sequential flow of the process. Through task assignment notifications the taxonomist and the curator are informed by emails about their tasks which arise from other user's previous activities. Further on, the workflow assures that information which is required in a later step is present before the execution continues, thus facilitating an efficient collaboration. In this example this has been realized at user interface level by the use of JSF validators in the associated task forms, though it could also has been realized at process definition level by means of a jBPM *task controller*.

Because the execution of processes and tasks is reflected in the wiki's semantic model it is also possible to use hypertext navigation facilities between task, process

[1] Workflow related functionality (e.g. listing and starting of processes) is provided by JSPWiki plugins in Makna which use RMI to interact with the jBPM engine.

[2] In our example we reuse the FungalWeb ontology [SNBHB05] for mycological classifications and TDWG's LSID ontologies for taxonomic data.

and user resources. Further on, Makna's click-searches which are provided with every resource in the wiki can also be helpful for collaboration. Examples include a list of all users with tasks in a certain process and a list of all specimen that have been identified by a certain taxonomist.

Another aspect of the improvement of collaboration support is the selection of an actor in the curator swimlane based on the results of the taxonomic identification and the semantic responsibilities descriptions in the wiki. In this example the taxonomic identification requires the selection of a *FungalWeb* concept and responsibility descriptions of actors refer to the same ontology. Because of the hierarchical structure of the *FungalWeb* ontology it is possible to infer the responsible actor, thus realizing a dynamic assignment strategy.

4 Conclusion

We have considered in this paper how the limited coordination, collaboration and integration support of semantic wikis could be improved and have presented a solution based on integration with a workflow system. Initial evaluation of our implementation has demonstrated improvements in these aspects which, in the authors' view, can prove to be of importance to the future uptake of semantic wiki systems, particularly in the corporate environment.

We plan to continue the development of Makna, and particularly to explore in a practical manner the deployability of semantic wikis in enterprise environments through our Corporate Semantic Web research group (*http://www.corporate-semantic-web.de*), which has as one of its goals research in corporate semantic collaboration. Makna and its workflow system integration is a first product of our work on corporate semantic wikis, and through industrial partnerships we plan to test the usability of Makna in real business scenarios where the described coordination, collaboration and integration support is of great importance.

5 Acknowledgments

This work has been partially supported by the "InnoProfile-Corporate Semantic Web" project funded by the German Federal Ministry of Education and Research (BMBF) and the BMBF Innovation Initiative for the New German Länder - Unternehmen Region.

References

[DPT06] Karsten Dello, Elena Bontas Simperl Paslaru, and Robert Tolksdorf. Creating and using semantic web information with makna. In Max Völkel and Sebastian Schaffert, editors, *Proceedings of the First Workshop on Semantic Wikis - From Wiki To Semantics*, volume 206 of *Workshop on Semantic Wikis*, pages S.43–57, Budva, Montenegro, June 2006. ESWC2006.

[SNBHB05] Arash Shaban-Nejad, Christopher Baker, Volker Haarslev, and Greg Butler. The fungalweb ontology: Semantic web challenges in bioinformatics and genomics. In *Proceedings of the International Semantic Web Conference 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 1063–1066. Springer, 2005.

SWOOKI: A Peer-to-peer Semantic Wiki

Charbel Rahhal, Hala Skaf-Molli, and Pascal Molli

LORIA – INRIA Nancy-Grand Est, Nancy Université, France
{Charbel.Rahal, skaf, molli}@loria.fr

Abstract. In this paper, we propose to combine the advantages of semantic wikis and P2P wikis in order to design a peer-to-peer semantic wiki. The main challenge is how to merge wiki pages that embed semantic annotations. Merging algorithms used in P2P wiki systems have been designed for linear text and not for semantic data. In this paper, we evaluate two optimistic replication algorithms to build a P2P semantic wiki.

1 Introduction

Nowadays, Wikis are the most popular web-based collaborative writing tools. In spite of their popularity, wikis suffer from the difficulty of navigation and information retrieval. To overcome these problems, some traditional wiki systems turned into semantic wikis. Semantic Wiki is a wiki engine with technologies from semantic web[1] to embed formalized knowledge, content, structures and links in wiki pages. Popular semantic wikis are based on the client-server architecture. All wiki pages reside on a single server that controls operations of distributed users. Consequently, scalability, performance, fault-tolerance and load balancing are major challenges for current semantic wikis. In addition, centralized architecture suffers from censorship problem and does not support off-line work. An approach to solve these problems is to shift from centralized architecture to full distributed (peer to peer) one. In this paper, we address the challenge of transforming a P2P wiki system into a P2P semantic wiki system. In fact, merging algorithms used in P2P wiki systems have been designed for linear text and not for semantic data. In this paper, we show how we can use the existent optimistic replication algorithms to build a P2P semantic wiki.

2 SWOOKI Approach

SWooki is the first attempt to build a peer to peer semantic wiki. SWooki is based on Wooki [1] a peer-to-peer wiki system. SWooki integrates the semantic web technology by following the philosophy of Semantic Media Wiki[2]. The semantic annotations are embedded in the wiki text via a wiki markup e.g. typed links. It follows the *use of wikis for ontologies* approach. A formal ontology emerges

[1] www.w3.org/2001/sw

during the edition of the wiki pages. SWooki provides the same functionalities of any server-based semantic wiki. In addition, SWooki allows the following three interesting use cases for P2P wikis: (1) a *massive collaboration*, (2) the *off-line editing*, and (3) an *ad-hoc collaboration* [3]. In order to combine P2P wiki system with semantic wiki systems, it is very important to know how semantic wikis represent their semantic data and how they combine textual parts with semantic parts. The main issue that we address is how to merge wiki pages that contain semantic annotations and if this combination changes the behavior of the Semantic Media Wiki. In this paper, we investigate how we can combine the Wooki with the Semantic Media wiki. We called this combination SWooki.

We adopt Wooki [1] because it supports all use cases for a P2P wiki system previously cited. In Wooki, wiki pages are replicated over all members of the p2p overlay network. A wiki page is considered as a sequence of lines. Each server hosts a copy of pages and can autonomously offer the wiki service. Page copies at each site are maintained by an optimistic replication mechanism called Woot [4] that disseminates changes and ensures consistency. Woot ensures the CSCW principles of convergence and user intentions. The only two available merging algorithms for peer-to-peer wikis are *the Thomas rule* [5] and *Woot* strategy. These algorithms handle linear text. The merge using these algorithms is done by the server which differs from the merge done by the users during concurrent editing in centralized semantic wikis. In case of Thomas rule, only the modifications of the user that lastly saved are kept. In case of Woot, the result includes the modifications of all users. However, the result in Woot is produced by the server and must be reviewed by a human in order to verify its accuracy. The Woot algorithm preserves users intentions, all concurrent effects are visible in the final version of the wiki page. If the user wants to change the result of the merge, he can do that easily. In conclusion, Swooki adopts the Woot algorithm because it provides the best solution for the merge by keeping all concurrent changes made by the users without any lost of updates. The SWooki approach allows to build a P2P semantic wiki very easily by integrating semantic annotations into a P2P wiki. It allows also to balance the load of queries. It provides a cheap way to have many replicas of the same semantic wiki. This total replication of semantic data can be used to distribute semantic queries on different replicas.

References

1. Weiss, S., Urso, P., Molli, P.: Wooki: a p2p wiki-based collaborative writing tool. In: Web Information Systems Engineering, Nancy, France, Springer (2007)
2. Völkel, M., Krötzsch, M., Vrandecic, D., Haller, H., Studer, R.: Semantic Wikipedia. Proceedings of the 15th international conference on World Wide Web (2006)
3. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Computing Surveys (2004)
4. Oster, G., Urso, P., Molli, P., Imine, A.: Data consistency for P2P collaborative editing. Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work (2006) 259–268
5. Johnson, P., Thomas, R.: RFC0677: Maintenance of duplicate databases. Internet RFCs (1975)

A Generic Corporate Ontology Lifecycle

Markus Luczak-Rösch and Ralf Heese

Freie Universität Berlin, Berlin 14195, Germany,
{luczak,heese}@inf.fu-berlin.de

Abstract. Weaving the Semantic Web the research community is working on publishing publicly available data sources as RDF data on the Web. To facilitate the adoption of Semantic Web technologies in corporate environments some issues on ontology engineering have to be addressed, e.g., support unexperienced employees to work collaboratively on ontologies. Although, existing methodologies structure well the process of ontology engineering, we miss an adequate tool support. We describe the Lekapidia case study and derive requirements for ontology engineering in a corporate environment. Furthermore, we present an extended ontology lifecycle integrating ontology engineering and ontology usage.

1 Introduction

Within the past years the Semantic Web community has developed a comprehensive set of standards and data formats to annotate semantically all kinds of resources, e.g., documents and images. Currently, a main focus lies on integrating publicly available data sources and publishing them as RDF on the Web, e.g., linking open data [1]. In contrast, many corporate IT areas are just starting to engage in Semantic Web technologies. Early adopters are in the areas of enterprise information integration, content management, life sciences and government [2]. Applying Semantic Web technologies to corporate content is known as *Corporate Semantic Web*.

To facilitate the adoption of Semantic Web technologies in a corporate environment some issues have to be addressed. Although ontology engineering methodologies might be well thought out, we miss an adequate tool support for unexperienced participants and the economic-driven needs of companies.

In Section 2 we present the results of the Lekapidia case study. A team of six people modeled collaboratively an ontology on desert recipes. Afterwards, this process was examined under the theoretic foundations of the DILIGENT methodology and simulated using a wiki-based tool for ontology engineering. We analyze the results of the case study in Section 3 and derive requirements on tools for modeling ontologies in a corporate environment. As a result we present an innovative two parts ontology lifecycle. Furthermore, we describe a corporate Semantic Web scenario, developed in cooperation with the Projektron GmbH: a semantic ticket system.

2 Lekapidia Case Study

In this section we describe the Lekapidia case study which we use to evaluate wiki-based ontology engineering empirically and to derive requirements for ontology engineering in a corporate environment. First we outline the setting of the case study and describe how the DILIGENT methodology was applied to this scenario. Afterwards, we present our conclusions drawn from the case study which refers to the special needs of corporate environments. In the Lekapidia case study a team of six students were asked to develop collaboratively a semantic wiki for desert recipes including a desert recipes ontology, while other four people were working as the software engineers. The teams were free in choosing the tools to build the ontology, to develop the application, to control the collaborative work, and to produce a documentation. They used Protégé for modeling tasks and a conventional MediaWiki for discussions.We used the case study for a valuable proof-of-concept of the DILIGENT ontology engineering methodology [3]. DILIGENT [4] assumes that ontology engineering scenarios are characterized by unexperienced and unequally skilled participants working in a distributed environment having individual needs on the ontologies. It permits local adoptions of ontologies and also defines a structured and iterative process for user argumentation to discuss changes of the central ontology. After reaching a consensus a central board decides on the integration of these adoptions into the central ontology. We used a wiki-based tool, coefficientMakna, as an integrative support to facilitate DILIGENT. For that reason a semantic wiki system was extended to support two semantic models. One model for statements about normal wiki pages and on model for discussion pages and pages which are marked as development issues or ideas. The structured argumentations follow the DILIGENT argumentation ontology. Thus, it is possible to hold discussions related to design issues as well as ontology primitives. When a decision is made the ontology consensus is build automaticly by processing the arguments. It is possible to build multiple ontologies for multiple groups in that way.

The participants of the Lekapidia project had no experience in ontology engineering. Examining the activities of the working groups we discovered a lack of communication. Considering DILIGENT, we discovered that the argumentation-based approach has enabled the unexperienced users to discuss their design decisions in an intuitive way. However, it does not support application-dependent or scenario-oriented ontology engineering. Empirical studies such as [5] state that the adoption of wikis in enterprises fails due to missing participation and underestimated entrance barriers of wikis.

Lekapidia does not allow any proposition about a long-running ontology engineering lifecycle, because the developed ontology was not deployed in a productive system. Even though, the structure of the project is close to real-world ontology development processes. The simulation with coefficientMakna allows to draw conclusions from a concrete methodological approach. We come to the conclusion that wiki-based approaches do not adequately support ontology engineering tasks. We identify a strong gap between the currently accepted ontology engineering approaches, e.g. DILIGENT, which suggest the applicability of wiki-based ontology engineering, and the needs of ontology engineering in corporate contexts. Ontologies are commonly seen as an artifact without any application-dependence. We suggest it as the outcome of a process which is concurrent while the ontology is in use and which is not finished after a decisive iteration step. Thus, appropriate methodologies and tools are needed, which respect this perspective.

3 Requirements of Corporate Ontology Engineering

The Lekapidia case study results a lack of adequate methodologies and tools respecting the agile character of ontology engineering. In the following we present new requirements for ontology lifecycles in a corporate environment. A key feature of our lifecycle is that it includes a cycle feeding back requirements on the ontology derived from its usage.

Corporate Ontology Lifecycle

Corporate Semantic Web refers to the usage of Semantic Web technology in a corporate environment. In a project of the same name we focus besides others on ontology engineering in a collaborative environment to increase the effectiveness of this process. A main advantage over realizing the Semantic Web is the controlled environment in a company. That allows us to name the boundaries of the setting as follows:

- Central allowance and control of the conceptualization
- Existing rules and workflows for employees
- Limited domain complexity
- Trust in semantic annotations

A main part of ontology engineering is the evolution of an ontology over life time. Figure 1 depicts our approach towards a corporate ontology lifecycle. The outer circle describes the engineering process by ontology engineers and domain experts while the inner one describes the adaption of the ontologies driven by usage requirements.

The ontology engineering process (outer circle) starts with the *creation/selection* phase by collecting and model knowledge fragments which results in a prototype ontology. This ontology is *validated* against the objectives. At the intersection point between the engineering and the usage cycles the engineers decide if the ontology reached a state to be used (*populated*) in the production system. If it does not meet the requirements or change requests arise from its usage the ontology engineers have to *evaluate* the current ontology. The *evolution/forward engineering* phase describes the task of changing the ontology to meet the new requirements. If an ontology has been populated to the production system then instances of concepts are generated by processing data and documents. The ontology is *deployed*. The *feedback tracking* phase is essential for adapting the

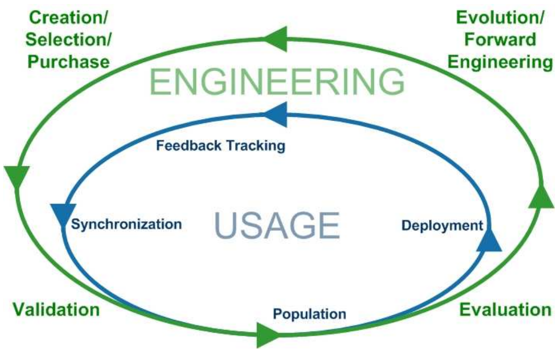


Fig. 1. The Corporate Ontology Lifecycle

ontologies to new requirements arising from its usage. A new requirement arises if a user gives explicit feedback, e.g., by arguing about concepts and relationships, or a system monitor generates conclusions by tracking the user behavior. The collected feedback and requirements are analyzed (*synchronization*) and if inconsistencies are recognized between the user's viewpoint and the ontology then ontology engineers start to adapt the ontology entering the outer cycle.

Use Case Semantic Ticket System

We transfer our lifecycle model into practice to support a feasible evaluation. In cooperation with the Projektron GmbH we evaluate the proposed ontology lifecycle in a real-world scenario: a semantic ticket system. Projektron uses and sells a ticket system which can be used to collect requests of customers, e.g., bug reports for a software. Although tickets are annotated with keywords and categories, similar tickets cannot be detected automatically. A main reason is the usage of synonym terms, e.g., differences in the terminology of the customer and the operating company of the ticket system. The difference in terminology may originate from the adaption of a software product to the terminology of the customer. For example, the customer uses "job" or "issue" instead of "task". Having the information about similar tickets a software engineer could solve these tickets in a single run and, thus, save time.

Establishing an ontology lifecycle as described above helps to keep track of customer-specific changes in the terminology of a software product. Furthermore the ontology has to be adapted to the terminology of ticket submitters to be able to detect similar tickets in the system.

We name the following requirements for a semantic ticket system aiming at an integrative support for our lifecycle:

1. Expert design tools enable the operating company to develop valid and consistent ontologies.

2. (Semi-)Automatic knowledge acquisition performed by machine learning algorithms, amongst others, lessens the effort for the ontology engineers to develop valuable ontology prototypes, e.g., based on the common terminology of a customer.
3. (Semi-)Automatic knowledge retrieval lessens the additional work for the user to annotate relevant data at the run-time.
4. Lightweight extended communication platforms, e.g., forums or feedback forms, and the automatic recovery of user behavior feature the adaption of new requirements arising from ontology usage.
5. Alternative intuitive visualization, e.g., graph visualizer, provide an intuitive navigation for users of any level of experience and enable easy detection of similar concepts.
6. Interfaces for applications are necessary to allow a number of applications to integrate as much consistent ontologies as needed.
7. Ontology storage and versioning enable centrally administration and configuration of the interdependence, matching and alignment of the various coexisting ontologies.

We will extend and use these requirements in progress towards an architecture for a holistic corporate semantic web and implement a practical proof of concept which respects them. Thus, we do not just transfer Semantic Web technologies from web-scale to corporate-scale, but improve the foundations by innovative research results which start from another point of view, e.g. ontology engineering as a usage-oriented lifecycle.

4 Related Work

Current lifecycle models for ontologies [6–8] consider only one cycle consisting of the phases design, validation, population, deployment, maintenance, and evolution. To our best knowledge there exists only one approach dividing the process of ontology engineering into two orthogonal cycles [9]. In contrast, our approach assumes a spiral model. The NeOn project [10] also researches the development of ontologies and focuses on standardizing the interchange of knowledge between world-wide operating enterprises. We assume a corporate environment, e.g., the ontologies are developed to process data and documents in a company effectively.

5 Conclusion

In order to find an applicable set of functional requirements for an integrative tool-support for ontology engineering in corporate environments, we used the results of the Lekapidia case study to discard wiki-based tools for this task. Based on ideas of the DILIGENT methodology and assumed characteristics of corporate settings, we constructed an innovative ontology lifecycle. The semantic ticketing use-case provides the basis for functional requirements which comply with the lifecycle integrative.

We expect the corporate ontology lifecycle to evolve towards a generic model, which enables companies to estimate the complexity and the chances of a transition from conventional information systems to ontology-based information systems. The approach will suite intra-corporate as well as inter-corporate settings.

Acknowledgement: This work has been partially supported by the "Inno-Profile-Corporate Semantic Web" project funded by the German Federal Ministry of Education and Research (BMBF).

References

1. W3C SWEO Community Project: Linking open data. http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData (2008)
2. Gartner, Inc.: Hype cycle for emerging technologies. http://www.gartner.com/it/page.jsp?id=495475 (2006)
3. Luczak, M.: Design and implementation of a wiki-based tool for collaborative ontology engineering. Master's thesis, Freie Universität Berlin (2007)
4. Pinto, S., Tempich, C., Staab, S., Sure, Y.: Distributed Engineering of Ontologies (DILIGENT). In: Semantic Web and Peer-to-Peer. Springer Verlag (2006) 301–320
5. Department of Personnel Economics and Human Resource Management of the University of Cologne: Wikis in enterprises. http://wikipedistik.de/survey/results.html (2008)
6. Gruninger, M., J., L.: Introduction. Commun. ACM **45**(2) (2002) 39–41
7. Novacek, V., Handschuh, S., Maynard, D., Laera, L., Kruk, S., Voelkel, M., Groza, T., Tamma, V.: Report and prototype of dynamics in the ontology lifecycle. Technical report, Galway, Ireland : Knowledge Web (2006)
8. Buitelaar, P.: NLP in the ontology life-cycle. http://www.lt4el.eu/content/files/ws_prague/eLearning-Prague.final.pdf (2007) Invited talk at the international workshop of the LT4eL project.
9. Staab, S., Studer, R., Schnurr, H.P., Sure, Y.: Knowledge processes and ontologies. IEEE Intelligent Systems **16**(1) (2001) 26–34
10. Tran, D.T., Haase, P., Lewen, H., Munoz-Garcia, O., Gómez-Pérez, A., Studer, R.: Lifecycle-support in architectures for ontology-based information systems. In: Proc. of the 6th Int. Semantic Web Conference (ISWC'07). (2007) 508–522

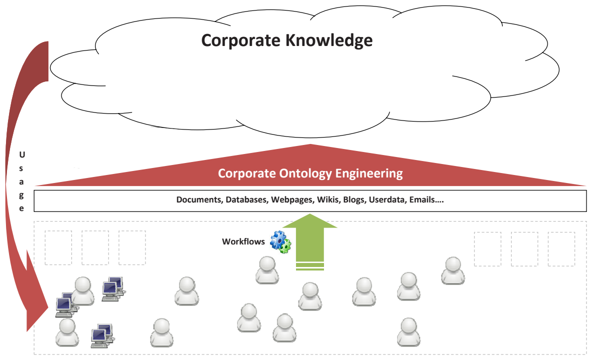
A Generic Corporate Ontology Lifecycle

Markus Luczak-Rösch and Ralf Heese

Freie Universität Berlin
Institute for Computer Science
Networked Information Systems
Königin-Luise-Straße 24/26
D-14195 Berlin
{luczak|heese}@inf.fu-berlin.de

Motivation

Although recent ontology engineering methodologies might be well thought out, we miss an adequate support for the economic-driven needs of companies and a context-dependent point of view. Corporate settings are characterized by a complex IT infrastructure, which provides various contents, such as documents or databases (domain context). Ontologies, as knowledge representation artifact in this setting, should respect the agility of the evolving knowledge of the whole context.


Corporate Knowledge
Usage
Corporate Ontology Engineering
Documents, Databases, Webpages, Wikis, Blogs, Userdata, Emails....
Workflows

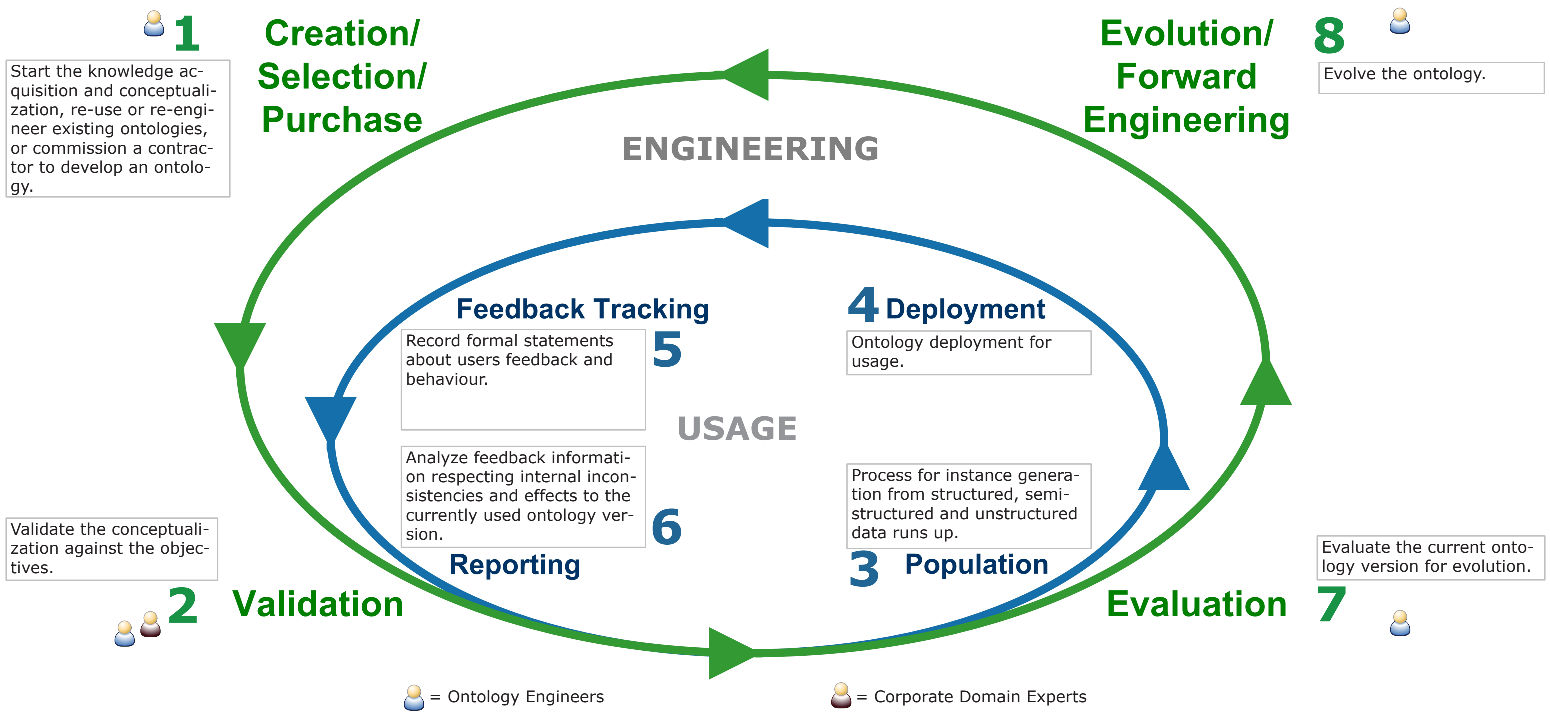
Approach

We present an innovative two parts ontology lifecycle, which has been concluded from requirements that we derived from the Lekapidia case study. This lifecycle separates engineering tasks from ontology usage and aspires a minimal effort of corporate human resources and a reduction of engineering tasks. The integrity of the rapidly released ontology prototypes is reached by an innovative tracking mechanism, which enables an automatic, implicit improvement of the conceptualization.

Corporate Ontology Engineering Settings

- Central allowance and control of the conceptualization
- Existing rules and workflows for employees
- Limited availability of domain experts for engineering tasks
- Trust in semantic annotations

The Corporate Ontology Lifecycle



1 Creation/ Selection/ Purchase
Start the knowledge acquisition and conceptualization, re-use or re-engineer existing ontologies, or commission a contractor to develop an ontology.

2 Validation
Validate the conceptualization against the objectives.

3 Population
Process for instance generation from structured, semi-structured and unstructured data runs up.

4 Deployment
Ontology deployment for usage.

5 Feedback Tracking
Record formal statements about users feedback and behaviour.

6 Reporting
Analyze feedback information respecting internal inconsistencies and effects to the currently used ontology version.

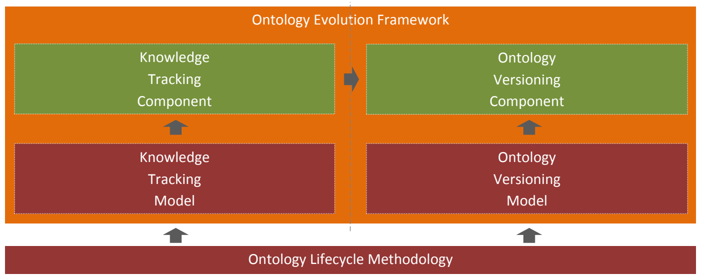
7 Evaluation
Evaluate the current ontology version for evolution.

8 Evolution/ Forward Engineering
Evolve the ontology.

= Ontology Engineers
= Corporate Domain Experts

Vision and Outlook

We aim at an extension of this approach towards an innovative architecture for ontology lifecycle management in corporate contexts. This architecture is carried by an ontology versioning mechanism, which makes use of an innovative knowledge tracking model to facilitate cost-effective, agile knowledge evolution. The lifecycle is designed to allow a cost-benefit-estimation in the forefront of each engineering iteration. A cost-benefit-estimation model for this purpose will be developed in the future.


Ontology Evolution Framework
Knowledge Tracking Component
Ontology Versioning Component
Knowledge Tracking Model
Ontology Versioning Model
Ontology Lifecycle Methodology

www.corporate-semantic-web.de

GEFÖRDERT VOM Bundesministerium für Bildung und Forschung
INNOPROFILE UNTERNEHMEN REGION Die BMBF-Innovationsinitiative Neue Länder
STI · BERLIN
Freie Universität Berlin

Descriptive Schema: Semantics-based Query Answering

S. D. Lee, Patrick Yee, Thomas Lee, David W. Cheung, Wenjun Yuan

Department of Computer Science, The University of Hong Kong.
{sdlee,kcyee,ytlee,dcheung,wjyuan}@cs.hku.hk

Abstract. We propose the novel concept of “descriptive schema” (DS). Unlike ordinary database schemas, a DS does not restrict the structure of the underlying database. Rather, it is just a probabilistic description of the structure. When answering keyword queries, DS can be used to improve semantics-based query answering and result ranking.

1 Schema: To have or not to have?

Wikipedia is a rich repository of information. However, facilities to exploit the information are still limited. Although typical search WWW search engines such as Google[1] allow users to look for information using keywords, they lack a schema for formulating the queries precisely.

Besides hyperlinks among the Wikipedia pages, many pages have Category tags as well as Infoboxes, which can be exploited to perform more sophisticated searches. For example, the DBpedia community makes use of these tags to build a database of RDF triplets, allowing more expressive and precise queries in the form of SPARQL to be used to retrieve useful information [2].

The above are two extremes of search and query. In the former case, the user can perform a search easily using relevant keywords, without having to learn the schema's lexicon beforehand. In the latter case, a schema can be used to help specify the query more precisely, but it has a non-trivial learning curve. In this paper, we propose the approach of “descriptive schema” to address these shortcomings. We attempt to strike a balance between the ease of use of a schema-less approach and the high accuracy that a schema-based system can bring us.

2 Descriptive Schema

In this paper, we propose a new concept called “Descriptive Schema” (DS). Unlike XSD (XML Schema Definition), DS is not meant to *prescriptively* mandate a structure on the underlying data. We want to retain the flexibility of free format for the pages. Rather, DS, as its name implies, is *descriptive.* It is only a summary of the structure exhibited by the underlying database. It does not define the structure. The data may occasionally violate the DS.

This tolerance to violations marks our biggest innovation, contrasting with existing approaches. Existing approaches to data modelling use “Prescriptive

Schema", which mandates a rigid structure on the underlying data, with little (if any) tolerance to violations.

We model a DS by a set of rules on the underlying data. There are many possible ways to formulate the rules. One example rule is: "90% of the time, a page of class 'Countries' has value for the field 'capital' in the infobox (infobox for countries)". Note that the rules defined in this way are probabilistic, because they are not satisfied all the time. A DS may thus be considered a summary of the patterns occurring in a database, instead of policies imposed on the data.

The task of discovering a DS from a database is a mining task, which is the problem of finding all rules satisfying a the specified syntax and support thresholds, thus following the data mining model in [3].

3 Applications

Since a DS captures semantical information about the underlying data, it enables a semantics-based approach to answering search queries. We can, for instance, use the DS to help us disambiguate the query, enrich the query with semantical information, as well as using the semantical information to rank the search results. Applications of DS include, but are not limited to, the following:

- Keyword Disambiguation
- Query Augmentation
- Result Ranking
- Data Cleansing
- Guidelines for Authors
- Guided Query Building

4 Conclusions

We have proposed the concept of "descriptive schemas", which is a set of rules obeyed by most of the underlying data, with tolerance for violations. Although the primary goal of devising this novel concept was to help answering keyword queries with an accuracy comparable to databases with prescriptive schemas, we have realized that DS can also be useful for other applications. Future works include exploring further potentials of DS, developing a formalism for it, devising efficient algorithms for mining DS, as well as more in-depth studies of the applications mentioned in this paper.

References

1. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks **30**(1-7) (1998) 107–117
2. Auer, S., Lehmann, J.: What have Innsbruck and Leipzig in common? extracting semantics from Wiki content. In Franconi, E., Kifer, M., May, W., eds.: ESWC. Volume 4519 of Lecture Notes in Computer Science., Springer (2007) 503–517
3. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Min. Knowl. Discov. **1**(3) (1997) 241–258

Descriptive Schema: Semantics-based Query Answering

S. D. Lee, Patrick Yee, Thomas Lee, David W. Cheung, Wenjun Yuan

Department of Computer Science, The University of Hong Kong.

`{sdlee,kcyee,ytlee,dcheung,wjyuan}@cs.hku.hk`

Abstract

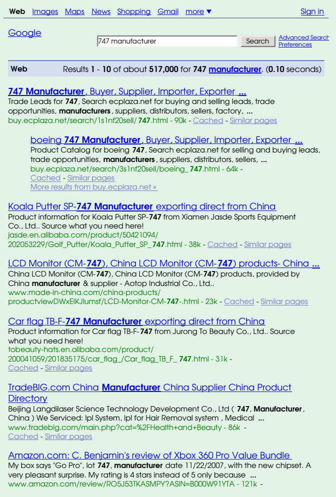
We propose the novel concept of "descriptive schema" (DS). Unlike ordinary database schemas, a DS does not restrict the structure of the underlying database. Rather, it is just a probabilistic description of the structure. When answering keyword queries, DS can be used to improve semantics-based query answering and result ranking.

1. Schema: To have or not to have?

- Wikipedia is a rich repository of information.
- But: not easy to extract information precisely.

Keyword Search: Search engines such as Google

- Easy to use: only need to enter keywords
- But: no schema for formulating precise queries.



Web Images Maps News Shopping Gmail more ▾ Sign in

Google [747 manufacturer] Search Advanced Search Preferences

Web Results **1** - **10** of about **517,000** for **747** **manufacturer**. (**0.10** seconds)

747 Manufacturer, Buyer, Supplier, Importer, Exporter ...
Trade Leads for **747**, Search ecplaza.net for buying and selling leads, trade opportunities, **manufacturers**, suppliers, distributors, sellers, factory, ...
buy.ecplaza.net/search/1s1nf20sell/ **747**.html - 90k - Cached - Similar pages

boeing **747 Manufacturer**, Buyer, Supplier, Importer, Exporter ...
Product Catalog for boeing **747**, Search ecplaza.net for selling and buying leads, trade opportunities, **manufacturers**, suppliers, distributors, sellers, ...
buy.ecplaza.net/search/3s1nf20sell/boeing_ **747**.html - 64k -
Cached - Similar pages
More results from buy.ecplaza.net »

Koala Putter SP-**747 Manufacturer** exporting direct from China
Product information for Koala Putter SP-**747** from Xiamen Jasde Sports Equipment Co., Ltd.. Source what you need here!
jasde.en.alibaba.com/product/50421094/
202053229/Golf_Putter/Koala_Putter_SP_**747**.html - 38k - Cached - Similar pages

LCD Monitor (CM-**747**), China LCD Monitor (CM-**747**) products- China ...
China LCD Monitor (CM-**747**), China LCD Monitor (CM-**747**) products, provided by China **manufacturer** & supplier - Aotop Industrial Co., Ltd..
www.made-in-china.com/china-products/
productviewDWxElKJlumsf/LCD-Monitor-CM-**747**-.html - 23k - Cached - Similar pages

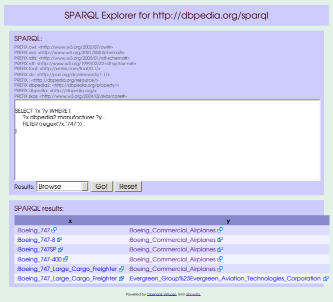
Car flag TB-F-**747 Manufacturer** exporting direct from China
Product information for Car flag TB-F-**747** from Jurong To Beauty Co., Ltd.. Source what you need here!
tobeauty-hats.en.alibaba.com/product/
200041059/201835175/car_flag_/Car_flag_TB_F_ **747**.html - 31k -
Cached - Similar pages

TradeBIG.com China **Manufacturer** China Supplier China Product Directory
Beijing Langdilaser Science Technology Development Co., Ltd (**747**, **Manufacturer**, China) We Serviced: Ipl System, Ipl for Hair Removal system , Medical ...
www.tradebig.com/main.php?cat=%2FHealth+and+Beauty - 86k -
Cached - Similar pages

Amazon.com: C. Benjamin's review of Xbox 360 Pro Value Bundle
My box says "Go Pro", lot **747**, **manufacturer** date 11/22/2007, with the new chipset. A very pleasant surprise. My rating is 4 stars instead of 5 only because ...
www.amazon.com/review/RO5J53TKASMPY?ASIN=B000W91YTA - 121k -

Schema-oriented Querying: à la DBpedia

- An RDF triple database retrieved from Wikipedia.
- Captures information from Category and Infobox tags.
- Richer in structure and semantics.
- Allows more precise SPARQL queries.
- But: need to learn the schema (lexicon + structure) of the data before posing useful queries.



SPARQL Explorer for http://dbpedia.org/sparql

SPARQL:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbpedia2: <http://dbpedia.org/property/>
PREFIX dbpedia: <http://dbpedia.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>

SELECT ?x ?y WHERE {
    ?x dbpedia2:manufacturer ?y .
    FILTER (regex(?x,"747")) .
}
```

Results: Browse Go! Reset

SPARQL results:

x	y
:Boeing_747	:Boeing_Commercial_Airplanes
:Boeing_747-8	:Boeing_Commercial_Airplanes
:Boeing_747SP	:Boeing_Commercial_Airplanes
:Boeing_747-400	:Boeing_Commercial_Airplanes
:Boeing_747_Large_Cargo_Freighter	:Boeing_Commercial_Airplanes
:Boeing_747_Large_Cargo_Freighter	:Evergreen_Group%23Evergreen_Aviation_Technologies_Corporation

Powered by OpenLink Virtuoso and dbpedia

A middle-ground : Descriptive Schema (DS)

- Ease of use: to search using keywords
- Precision of query: approaching the precision of schema-oriented queries
- Idea: Using the DS and the search keyword, guess and formulate a relatively precise query to the RDF triples.

2. Descriptive Schema

- We propose a new concept called "Descriptive Schema" (DS).
  - Unlike ordinary database schemas (e.g. XSD), DS is not meant to *prescriptively* mandate a structure on the underlying data.
  - DS is meant to retain the flexibility of free format for Wiki pages.
  - DS is *descriptive*: It is only a summary of the structure exhibited by the underlying data.
  - The data may occasionally violate the DS.
- We model a DS by a set of probabilistic rules, e.g. 90% of the time, a page of class 'Countries' has value for the field 'capital' in the infobox (infobox for countries).
- The task of discovering a DS from a database is a mining task.

3. Applications

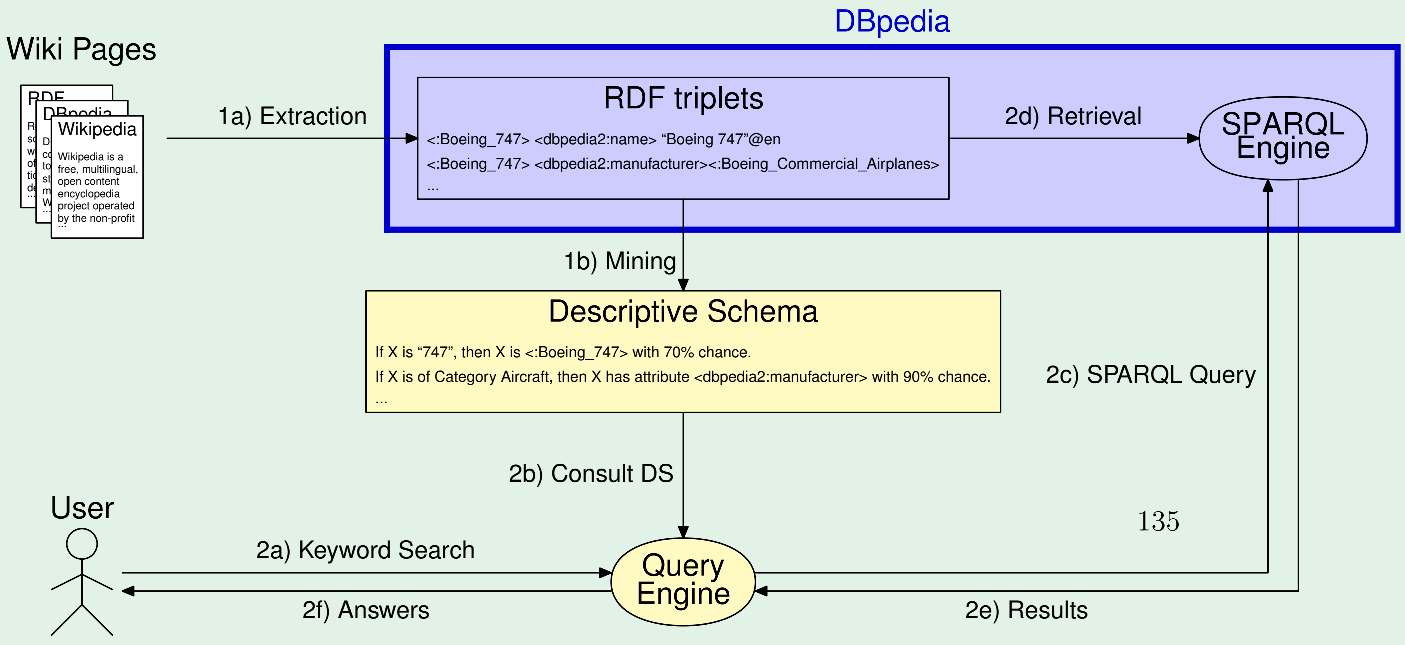
Applications of DS include, but are not limited to:

- Keyword Disambiguation
- Query Augmentation
- Result Ranking
- Data Cleansing
- Guidelines for Authors
- Guided Query Building

4. Conclusions

We have proposed the concept of "descriptive schemas":

- a set of rules obeyed by *most* of the underlying data with tolerance for violations.
- meant to help answering keyword queries with an accuracy comparable to databases with prescriptive schemas.
- DS may also be useful for other applications.
- Future works:
  - exploring further potentials of DS
  - developing a formalism for DS
  - devising efficient algorithms for mining DS



<:Boeing_747> <dbpedia2:name> "Boeing 747"@en
<:Boeing_747> <dbpedia2:manufacturer><:Boeing_Commercial_Airplanes>
...

If X is "747", then X is <:Boeing_747> with 70% chance.
If X is of Category Aircraft, then X has attribute <dbpedia2:manufacturer> with 90% chance.
...

Property Clustering in Semantic MediaWiki

Define Your Own Classes and Relationships

Dr. Gero Scholz

IVU Traffic Technologies AG, Berlin, Germany

Abstract. Semantic MediaWiki (SMW) currently has an atomic understanding of properties: they are seen as annotation marks which can be arbitrarily attached to articles. As a next step towards an object oriented representation of knowledge we introduce a concept of *property clustering*. This makes it possible to define a formal meta model for a knowledge domain. We support class inheritance and typed relations between objects. As a proof of concept we provide an implementation which is based on a set of templates and a few existing MediaWiki extensions. A graph of the meta model can be generated automatically. We offer different models for entering information based on templates and forms. A demo website (`http://semeb.com/dpldemo/SMWpc`) is available.

Keywords: *Semantic MediaWiki, Semantic Forms, Class, Relation, Inheritance, Meta Model, Proof of Concept, Demo Implementation*

1 Introduction

Currently in SMW every possible combination of properties can be assigned to every article. It is possible to assign multiple values for the same property to the same article. The difference between *relations* and *values* which was part of the SMW concept in older versions has been dropped in favor of more generalized *properties* in the latest SMW release. All this leads to a fairly universal, generic concept. In short, SMW offers a concept of *weak typing* expressed by arbitrary bundles of properties taken from an ocean of all possible attributes which might be useful for annotation.

But people do not primarily perceive objects as conglomerates of attributes. Instead they classify objects and use well defined names for these classifications. Classes in essence are *named clusters of properties*. Consequently, this article introduces a concept of *strong typing* which we call SMWpc. The 'pc' might translate to 'property clustering' or to 'personal classes'. The latter interpretation would emphasize that the design of classes always depends on the perspective of authors and readers.

SMWpc is a proof of concept which is already usable for small wikis. It is based on SMW, a few other MediaWiki extensions and some tricky MW templates. To improve performance and robustness a more professional implementation should be made by extending the current php source code of SMW.

2 Idea and Concept

The graph in Fig.1 describes the general idea of SMWpc.

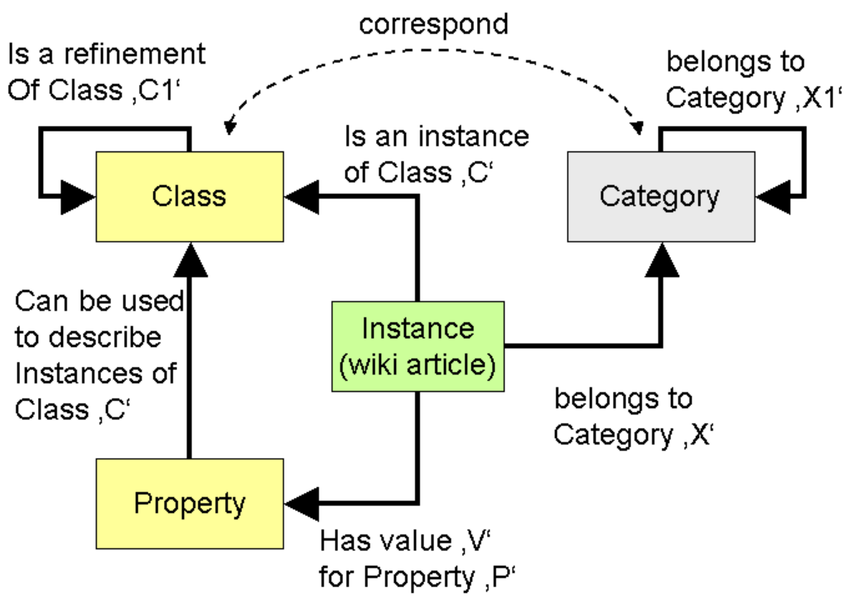

correspond
Is a refinement Of Class ‚C1'
belongs to Category ‚X1'
Is an instance of Class ‚C'
Class
Category
Can be used to describe Instances of Class ‚C'
Instance (wiki article)
belongs to Category ‚X'
Property
Has value ‚V' for Property ‚P'

Fig. 1. Meta Model of SMWpc

In SMWpc MediaWiki articles are seen as *instances of a classes (objects)*. A class is formally described in a meta model using special *meta properties*. Each class in SMWpc corresponds to a traditional MediaWiki category which is named after the class. There are no freely floating properties in SMWpc. Instead properties are always tied to classes. A special meta property is used to describe class inheritance.

3 Meta Model

A full version of the meta model can be found on the website. The most important meta property is *.obj is a*. It states that an article describes an object of a certain class. The meta property *.prop describes* ties a property to its class. Note that one property can be tied to many classes. *.class extends* is used to define (single) inheritance. It is a good design principle to use templates for the assignment of property values. Via *.prop assigned by* we establish a reference between a property and its associated 'assignment template'. Sometimes the value of a property can be algorithmically derived from the values of one or more other properties. We use *.prop derived from* to express this. The meta property *.prop refers to* allows to express that a property of a class is to be understood as a reference to an object of another class. *.prop reverse* offers a second name for the same relation if used in the opposite direction. The properties *.prop unique* and *.prop mandatory* express the cardinality of properties, i.e. they state if zero, one or many values will be allowed for a certain property.

Apart from these essential features there are other meta properties which can help you to attach color schemes or icons to classes and properties. There is also a meta property that links an edit form to a class. And last not least there are class-specific templates which produce a nice common layout for all objects belonging to the same class.

As you may have noted all SMWpc meta properties start with a prefix like *.obj, .class, .prop, ..* to make clear that they do not belong to the application domain of the wiki. It would be a good idea to use the same convention for SMWs existing meta properties like 'has type', 'allows value' etc. There should be a clear separation of namespaces between the meta model and the application domain of a wiki. Technically speaking all SMWpc meta properties are normal SMW properties. This allows to use the concept of reflection (introspection) in the implementation of SMWpc. SMW should consider to follow the same strategy. It would be of great value to operate on the *information model* of a wiki in the same query language that you use to operate on its *contents.*

4 Focus of SMWpc

The initial version of a wiki typically contains a small, weakly structured collection of articles which have some commonalities. Once a wiki grows the designer of the wiki can use SMWpc to create a formal meta model which supports queries and helps to enter information in a more structured way. It is important to closely monitor the ratio between the size of the 'information model' and the total amount of information in a wiki. Encyclopedic wikis will have a lower ratio than specialized wikis with closer scope and more elaborated relationships between the articles. Most often there will be a perceived lack of semantic structure in a wiki. But there is also a (small) danger of over-engineering when a small wiki is started with a very rigid structure.

The main focus of SMWpc is on small and medium-size wikis (less than 10.000 pages) which have a dedicated focus. Their user communities agree on a common scheme for classification of articles and they want better support for collecting highly structured information. An example could be a wiki in the area of molecular genetics but it could also be a wiki about pets where you have classes like *species, food, disease* etc. It is quite clear that a property named *symptom* belongs to class *disease* and not to *food* or *species.* With SMWpc there is a way to express this. While it may make a lot of sense to have multiple values for the symptoms of a disease, there should only be a single value for the property *maximum age* of class *species*. The property *likes* must contain a reference to an instance of *food* and not to a *disease*. With SMWpc you can express all this and much more.

5 Example

We set up an example which deals with students, their subjects of study and their hobbies (playing games and playing musical instruments). The example tries to demonstrate all features of SMWpc. So do not pay too much attention to the contents. The information model generated by SMWpc ios shown in Fig.2. For more information please go to the website (`http://semeb.com/dpldemo/ClassStudent`).

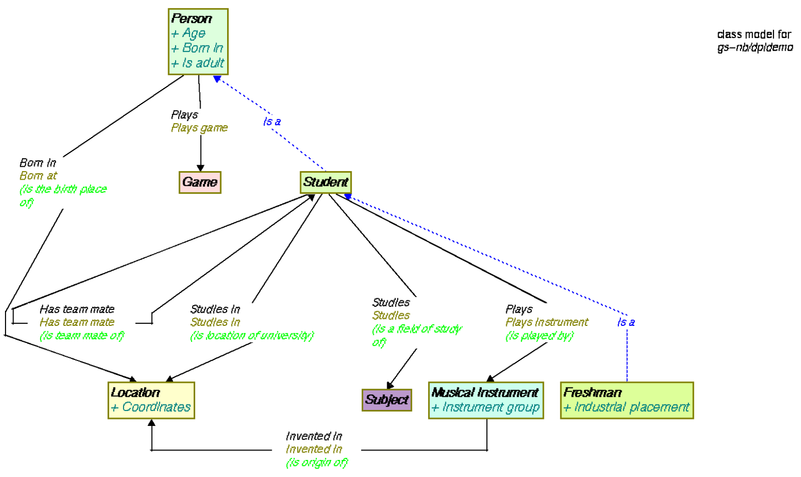


Fig. 2. Sample class model of an SMWpc application

6 Conclusion

SMWpc is a first step in the direction of true object oriented semantic modeling with MediaWiki. There are lots of features which can be improved and added in future. And there is much more functionality already available than could be shown and explained here. We hope that the idea of SMWpc will be adopted by the Semantic MediaWiki community. Integration of SMWpc concepts into SMW would create a more robust solution with better performance. Adding SMWpc concepts to SMW would enlarge the scope of SMW significantly. It would be a pure add-on, so no current functionality would be lost.

BOWiki: ontology-based semantic wiki with ABox reasoning

Joshua Bacher[1,2,3], Robert Hoehndorf[1,3,4] and Janet Kelso[3]

[1] Department of Computer Science, Faculty of Mathematics and Computer Science, University of Leipzig, Johannisgasse 26, 04103 Leipzig, Germany
[2] Institute for Logics and Philosophy of Science, Faculty of Social Science and Philosophy, University of Leipzig, Beethovenstrasse 15, 04107 Leipzig, Germany
[3] Department of Evolutionary Genetics, Max Planck Institute for Evolutionary Anthropology, Deutscher Platz 6, 04103 Leipzig, Germany
[4] Research Group Ontologies in Medicine (Onto-Med), Institute of Medical Informatics, Statistics and Epidemiology (IMISE), University of Leipzig, Härtelstrasse 16-18, 04107 Leipzig, Germany

Claim. This paper presents the semantic wiki BOWiki, that uses a ontology to verify the content of semantical dat added by the user. The BOWiki is a semantic Wiki, designed to eliminate the need for costly and time consuming manual expert database curation, while providing users with an automated reasoning system to verify the consistency of newly added content to the knowledgebase (KB). A semantic wiki built on an ontological foundation can provide users with information about particular types of entities and how they relate to one another. Automated reasoners can be adapted for use within an ontologically based semantic wiki, in order to verify whether newly submitted information is consistent with existing KB content, prior to incorporating the new information into the KB [5]. The reasoner is also useful for querying the data. The BOWiki combines an ontologically based semantic wiki with an automated Pellet reasoner to deliver users a collaboratively curated and consistent KB. Although originally targeted to serve the biological community, the BOWiki can be used in any domain.

1 Implementation and Usage

The BOWiki is an extension of the MediaWiki and comprised of 4 parts (a figure is accessible online[1]: (a) the BOWiki software extension, (b) the BOWikiserver, (c) the BOWiki database extension and (d) an OWL-DL ontology. The BOWiki extension to the MediaWiki is the main application component. It both displays data and interacts with the user. The BOWiki extension communicates with the BOWikiserver over a custom-designed protocol. The BOWikiserver classifies the content in the BOWiki's current KB and has the capacity to reason over the KB. For this purpose, the BOWikiserver uses the Jena 2 Semantic Web Framework [1] and currently employs the Pellet OWL Reasoner [4]. The database extension provides persistent storage of the BOWiki's KB, which enables revovery of

[1] See http://onto.eva.mpg.de/pub/eswc-misc/

the KB content in the event the BOWikiserver fails. During the BOWiki setup, when the BOWiki is initialized, an OWL-DL ontology must be imported into the BOWiki.
The BOWiki markup plays an important role in the BOWiki's operations. A translation between the BOWiki markup and OWL [3] is available online[1]. It illustrates how it is translated from BOWiki syntax into appropriate OWL Syntax. During installation, an OWL-DL ontology must be chosen for importing. The types and binary relations used in these extensions come from an OWL-DL ontology [3], which must be imported into the BOWiki during setup. In addition to this markup the BOWiki allows for inline queries[2]. When a wikipage is modified and one of the BOWiki markup extensions is used, the newly submitted data is immediately processed by the BOWikiserver and its consistency verified. Only consistent data is added to the BOWiki's KB. Inconsistent changes are rejected, and a notification with an explanation of the inconsistency is provided to the user. The BOWiki further includes several features intended to help users with basic functionality: special pages allow reviewing all relations and all OWL classes known to the BOWiki's reasoner; allow importing ontologies in the OBO flatfile format [2]; rebuilding the KB from data stored in the BOWiki database and exporting the content of the BOWiki's KB to OWL. An online tutorial[2] guides new users in using the BOWiki.

2 Conclusion

We designed the BOWiki, an extension to the MediaWiki, to enable biologists to develop a collaboratively curated KB that automatically verifies its ontological adequacy. As a semantic wiki built on an ontological foundation, the BOWiki provides its users not only with information about particular entities, but also tells users how these entities relate to one another. The automated Pellet reasoner verifies the consistency of newly submitted information to the KB, thereby avoiding the incorporation of inconsistent information that sometimes plagues user curated systems.

References

1. Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: Implementing the Semantic Web Recommendations. Technical Report HPL-2003-146, Hewlett Packard, Bristol, UK, 2003.
2. Christine Golbreich and Ian Horrocks. The OBO to OWL mapping, GO to OWL 1.1! In *Proc. of the Third OWL Experiences and Directions Workshop*, number 258 in CEUR (http://ceur-ws.org/), 2007.
3. Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. *W3C Recommendation*, 2004.
4. Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 2007.

[2] See http://bowiki.net/wiki/index.php/Tutorial

5. Denny Vrandecic and Markus Krötzsch. Reusing Ontological Background Knowledge in Semantic Wikis - From Wikis to Semantics. In *Proceedings of the First Workshop on Semantic Wikis*, 2006.