Project no. FP6-028038

Palette

Pedagogically sustained Adaptive LEarning Through the exploitation of Tacit and Explicit knowledge

Integrated Project
Technology-enhanced learning

D.KNO.06
KM services for CoPs

Due date of deliverable: January 31, 2008
Actual submission date: February 11, 2008

Start date of project: 1 February 2006
Duration: 36 months
Organization name of lead contractor for this deliverable: INRIA

Project co-funded by the European Commission within the Sixth Framework Program		
Dissemination Level		
R	PUBLIC	PU

Keywords: Knowledge Management, Services, Semantic Web, Ontologies, Annotations
Responsible partner: Rose Dieng-Kuntz (INRIA)

MODIFICATION CONTROL			
Version	**Date**	**Status**	**Modifications made by**
V0.1	12/12/2007		Amira Tifous (INRIA) : chapter on SweetWiki
V0 2	12/12/2007		Alain Vagner Jean-David Labails Yannick Naudet Marie-Laure Watrinet Benjamin Gateau (CRP HT) : chapter on Bayfac
V0.3	15/12/2007		Bassem Makni, Rose Dieng-Kuntz (INRIA): chapter on SemanticFAQ
V0.4	18/12/2007		Stéphane Rieppi, Étienne Vandeput, ULg : evaluation of SemanticFAQ
V0.5	18/12/2007		Adil El Ghali (INRIA) : chapter on KM LinkWidget
V0.6	18/12/2007	Draft	Olivier Corby (INRIA) : first integration
V0.7	18/01/2008	Draft	Adil El Ghali : complements on SweetWiki & second integration
V0.8	23/01/2008	Draft	Rose Dieng-Kuntz: corrections + introduction and conclusion
V0.9	23/01/2008	Draft	Olivier Corby : last integration
V1.0	23/01/2008	Sent to Reviewers	Rose Dieng-Kuntz
	28/01/2008	Feedback of Reviewer	Nikos Karacapilidis (CTI)
		Feedback of Reviewer	Aida Boukottaya (UNIFR)
	31/01/2008	Feedback of Reviewer	Manfred Kunzel (UNIFR)
		Corrections taking into account reviewers' reports	All
V1.9	05/02/2008	Integration of the corrections	Amira Tifous
V2.0	11/02/2008	Sent to SC	Rose Dieng-Kuntz

Deliverable manager: Olivier Corby (INRIA)

List of contributors:
Rose Dieng-Kuntz (INRIA)
Benjamin Gateau (CRP HT)
Adil El Ghali (INRIA)
Jean-David Labails (CRP HT)
Bassem Makni (INRIA)
Stéphane Rieppi (ULg)
Étienne Vandeput (ULg)
Amira Tifous (INRIA)
Alain Vagner (CRP HT)
Marie-Laure Watrinet (CRP HT)

List of evaluators:
Aida Boukottaya, EPFL
Nikos Karacapilidis, CTI
Manfred Kunzel, UNIFR

Acknowledgements
We thank very much Khaled Khelif and Hacène Cherfi for their support for SemanticFAQ.

Summary

This deliverable describes several KM services developed for CoPs. It details the current achievements on the developments of the semantic wiki SweetWiki offering collaborative knowledge creation services. It presents SemanticFAQ that offers a service for semi-automatic annotation of a corpus of e-mails and enables information retrieval from such e-mails. The results of a first user-centered evaluation of SemanticFAQ portal are also presented. Then we describe the principles and architecture of the KM LinkWidget, that is aimed at enabling CoP members to link resources stored in a repository with conversations in a discussion forum, by using semantic annotations of the resources and the discussions. Last, the deliverable describes the new functionalities of BayFac (offering a service of document classification and search), developed in order to take into account remarks of CoP members.

Contents

Chapter 1 Introduction 7
Chapter 2 Collaborative Knowledge Creation Service: SweetWiki 9
2.1. Current achievements 9
2.1.1 Bug correction 9
2.1.2 Interface improvements 10
2.1.3 URIs visibility and multiple ontologies 11
2.1.4 Password management 11
2.1.5 SPARQL query library 12
2.1.6 Restructuring the source tree 17
2.1.7 Awareness through RSS feeds subscription 19
2.2. Future development 20
2.2.1 First planned release 20
2.2.2 Second planned release 21
2.2.3 Perspectives 22
Chapter 3 Semi-automatic Annotation and Information Retrieval from Mails: SemanticFAQ 23
3.1. Introduction 23
3.2. Definition 23
3.3. Semantic annotation of mails 24
3.3.1 Metadata annotation 24
3.3.2 Content annotation 25
3.4. Algorithm for information retrieval from mails 26
3.5. Information Retrieval from Mails 27
3.5.1 Ontology based navigation 27
3.5.2 Semantic portal 27
3.6. User-centered analysis of SemanticFAQ 29
3.6.1 First Look 29
3.6.2 Ludicity 30
3.6.3 Ontology enrichment 31
3.6.4 Ergonomic analysis and suggestions for future releases 32
3.7. Related Work and Further Work 34
Chapter 4 KM LinkWidget 37
4.1. Introduction 37
4.2. Definitions 37
4.2.1 Repository and Semantic Repository 37
4.2.2 Forum 38
4.3. How it will work 39
4.3.1 Annotation of resources 39
4.3.2 Annotation of discussions 39
4.3.3 Collecting external annotations 40
4.3.4 Retrieving/Calculating Links 40
4.3.5 Displaying Links 40
4.4. Architecture 40
4.4.1 Used technologies 41
4.5. Scenario of use by CoPs 41
4.5.1 Did@cTIC 41
4.5.2 Learn-Nett 42
4.6. Conclusions 42
Chapter 5 Faceted Classification and Search Service: BayFaC 43

5.1. Functional update 43
5.1.1 Role management 43
5.1.2 Private document management 44
5.1.3 Facet management 44
5.1.4 Installation and configuration 47
5.1.5 Facet Ontology improvement 48
5.2. Interface improvement 49
5.2.1 Classification evaluation 49
5.2.2 Search evaluation 50
5.3. Bayesian Performance feedback and evaluation 50
5.4. Interoperability 51
5.4.1 REST Web Services 51
5.4.1.1. Context 52
5.4.1.2. BayFac Search 53
5.4.1.3. BayFac Classification 53
5.4.2 Data import 54
Chapter 6 Conclusions 57
Appendix A BayFac WADL 61
A.1. Main WADL file 61
A.2. defaultMessage.xsd 64
A.3. facet/index.xsd 64
A.4. facet/show.xsd 65
A.5. facetItem/index.xsd 65
A.6. facetItem/show.xsd 65
A.7. facetVector/create.xsd 65
A.8. facetVector/index.xsd 66
A.9. facetVector/show.xsd 66
A.10. facetVector/update.xsd 67
A.11. fs/index.xsd 67
A.12. fs/show.xsd 67
A.13. instance/create.xsd 68
A.14. instance/index.xsd 68
A.15. instance/show.xsd 68
A.16. instance/update.xsd 68

Chapter 1 Introduction

The deliverable D.KNO.04 [D.KNO.04] presented several basic KM services: (a) services based on Corese such as administration service, ontology management, annotation management, retrieval and export; (b) services based on Generis such as ontology edition, knowledge annotation and knowledge retrieval. It also introduced two complex KM services: SweetWiki that offers collaborative knowledge creation services and BayFac that offers service of document classification and search.

This deliverable details the new developments of SweetWiki and BayFac services and introduces new complex KM services:

- SemanticFAQ, a service for semi-automatic annotation of a corpus of e-mails and information retrieval from such e-mails. Such a service can be useful for CoPs exchanging through e-mails (e.g. @pretic CoP) or even discussion forums.
- KM LinkWidget, a service aimed at enabling CoP members to link resources stored in a repository with conversations in a discussion forum, by relying on semantic annotations of the resources and the discussions.

Chapter 2 will focus on SweetWiki current achievements (refactoring, bugs correction, interface improvements, SPARQL query library, password management and awareness through RSS feeds subscription).

Chapter 3 details SemanticFAQ, the techniques for semi-automatic annotation of a corpus of e-mails, the ontology-based navigation on such e-mails (relying on the @pretic ontology described in deliverable D.KNO.05) and the semantic portal developed, as well as its evaluation by @pretic CoP mediator.

Then chapter 4 presents a new service, KM LinkWidget, aimed at being used by Did@ctic and Learn-Nett CoPs. The deliverable describes the principles and architecture of this service aimed at enabling CoP members to annotate resources and discussions and at retrieving links between resources and discussions.

Chapter 5 describes the new functionalities of BayFac developed in order to take into account feedback of CoP members: role management, private document management, facet management, and improvements of classification and search interfaces as suggested by CoP members, Interoperability is offered through implementation of BayFac classification and search features as RESTful services.

After conclusions in chapter 6 summarizing the contributions of this deliverable and the further work planned on KM services in WP3, the appendix presents the BayFac WADL files.

Chapter 2 Collaborative Knowledge Creation Service: SweetWiki

In this chapter, we detail the current achievements on the development of the semantic wiki SweetWiki and give the outlines of the future development to be made. These improvements are mainly guided and motivated by the feedbacks gathered during the experiments that we led with Palette CoPs on the use of SweetWiki [Vandeput and Ledent, 2007] [D.KNO.04].

2.1. Current achievements

The first achievements concern the development having been done for the versions v1.5 and v1.7 (ongoing work). A major part of this development dealt with code refactoring and bug correction.

2.1.1 Bug correction

A few examples of such bugs:

- Search queries with multiple tags: the AND operation aiming at returning the pages and objects (images, videos and attached files) tagged with all the tags submitted in the query was not working well.
- Management of WikiWords: sometimes, the WikiWords were not detected.
- Management of tags and WikiWords with invalid characters: mechanisms for the processing of the non valid characters have been implemented. The accentuated tags and WikiWords are transformed to their closest non accentuated equivalents.
- Duplicated entries: for example in her HomePage, a user can specify the tags she's interested in, so as to see the pages annotated with these tags only by accessing her HomePage. The results being sorted according to the tags, it can happen that one page is displayed many times (if it is annotated with many tags of the user's list).

```
Results = { (P1,t1), (P2,t1), (P3,t2), (P1,t2), ... } ;
                    Pi pages, tj tags
```

We improved this functionality so as to see the pages.

```
Results = { (P1,(t1,t2)), (P2,t1), (P3,t2), ... }
```

- Pages with the same name, belonging to different Webs[1]: this situation led to have incomplete queries results, because these queries did not take into account the Webs which the pages belonged to. Therefore, only one of the pages was displayed, overwriting references to the others.
- All the links in "Advanced Search/See all created links" were wrong.
- In "Advanced Search/See All Tags" option, the tags, their parents and the number of pages and objects they are used to annotate are displayed. However, since the case of multiple inheritance has not been considered, the layout of the parents tags of each tag was not adapted.
- Office documents import: their management was not efficient enough since the non

[1] A Web is a sub-space of the Wiki. A Web contains WikiPages, whereas a WikiPage belongs to one Web.

local images were not displayed.

2.1.2 Interface improvements

- **Table of content**

We implemented for SweetWiki pages a widget allowing us to display the table of content of the current WikiPage, based on its structure. This widget allows the user to have a glimpse of the page content (particularly useful when the pages are long). It also enables to skip to the desired section of the page by clicking on its corresponding title in the Table of content.

- **Parameterized Look and feel**

We enhanced SweetWiki with an easy to use interface enabling the wiki administrator to parameterize the Look and Feel of each WikiWeb.

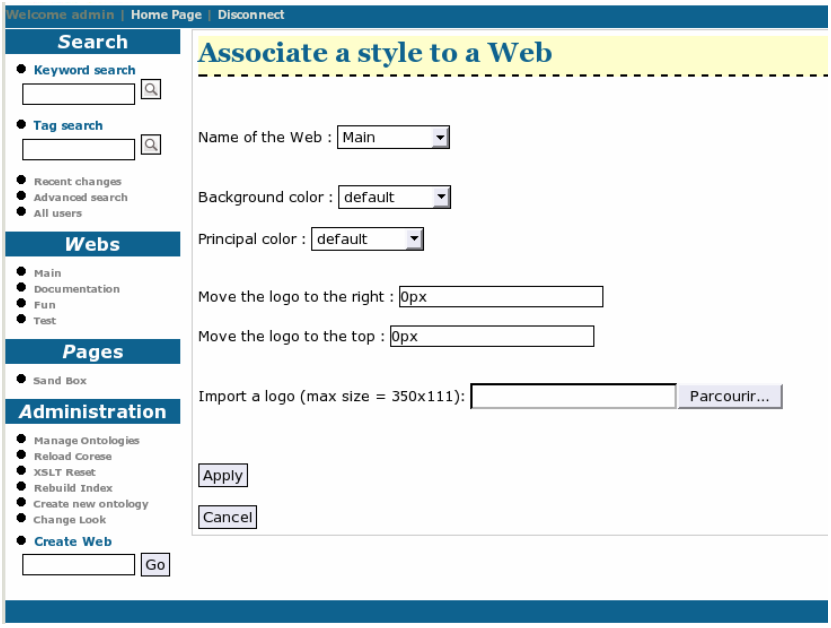


Figure 2.1: Interface for Changing the look of a Web

As illustrated by figure 2.1, the administrator can choose the Web to change and, for this Web, its background color, the color of the interface components and widgets. The administrator can also add a logo to the Web and adjust its position in the layout.

Each time one of these operations is executed, its result is real-time displayed on the current page, changing its layout, so as to enable the administrator to have a view of the changes to occur on all the pages that belong to the chosen Web.

Once the administrator validates the desired changes, the CSS that specifies the layout of the pages of the Web is modified.

- **Date/time layout**

SweetWiki offers the possibility to the users to search for pages by building and submitting advanced queries, based on multiple criteria, such as the modification date.

Considering that such information is very relevant, in a collaborative environment where a page can be modified many times a day, and where the old versions of a current page can also

be accessed, we made it more detailed than it was, by including the exact time of the modification of the pages.

This improvement involved the modification of SweetWiki ontology which describes the model of SweetWiki. In this ontology, the value type of the “modification” attribute of a WikiPage has been modified from:

`http://www.w3.org/2001/XMLSchema#date`

to

`http://www.w3.org/2001/XMLSchema#dateTime`

```
<rdf:Property rdf:ID="modification">
    <rdfs:label xml:lang="en">last update</rdfs:label>
    <rdfs:label xml:lang="fr">dernière modification</rdfs:label>
    <rdfs:comment xml:lang="en">Date of the last update of the
document</rdfs:comment>
    <rdfs:comment xml:lang="fr">Date à laquelle le document a été modifié pour la
dernière fois </rdfs:comment>
    <rdfs:domain rdf:resource="#WikiPage"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
</rdf:Property>
```

2.1.3 URIs visibility and multiple ontologies

In the first versions of SweetWiki, the URIs used in the system were not accessible from outside. They have been modified in order to be accessed, which makes it possible to exploit SweetWiki folksonomy and annotations by other Palette services. Here is the list of accessible resources:

The wiki model	http://sweetwiki.inria.fr/<instance>[2]/files/ontologie/wiki.rdfs http://ns.inria.fr/sweetwiki/2007/08/wiki#
The folksonomy	http://sweetwiki.inria.fr/<instance>/files/ontologie/folksonomy.rdfs
The queries library	http://sweetwiki.inria.fr/<instance>/files/queries/queries.xml
Annotations for the Page: PageName	http://sweetwiki.inria.fr/<instance>/files/rdf/Web_PageName.rdf

In addition, we make possible the use of multiple ontologies (other than the folksonomy) for tagging. The CoPs can now use the O'CoP ontology or other domain ontology to tag their pages. The semantic search will be improved thanks to the stronger organization of these ontologies compared to a folksonomy.

2.1.4 Password management

In this version, the wiki users have the possibility to receive their password by e-mail. The development of this functionality was `not planned in the development schedule of SweetWiki. However, it has been done, considering the high need expressed by the Palette

[2]<instance> stands for the name of the wiki instance we refer to. For example, for the wiki used by Palette members it is *swikipalette.*

users.

As a consequence, we had to modify the initial design of SweetWiki, in which this issue had not been considered. Indeed, initially, the login information stored consisted of the user's WikiName and password.

If a user wants to know her password automatically and autonomously, without appealing to the administrators of SweetWiki, this password has to be sent to her by e-mail. Therefore, passwords recovery functionality relies on an authentication based on the WikiName and e-mail address of the users.

Thus, to cope with this issue, we had to modify the structure of the file containing the users' login information.

This file is exploited by means of a servlet which is invoked when a user fills the form dedicated to authenticate herself with her WikiName and her e-mail address. This servlet checks the validity of the information provided by the user through the form, and if this information is right, sends the user an e-mail containing her password.

2.1.5 SPARQL query library

We analyzed in depth the code of SweetWiki so as to factorize all the SPARQL[3] queries. These queries were embedded into the JSP pages and Java classes used to provide search and layout functionalities (see the deliverable [D.KNO.04] for details). We gathered these queries into an XML file following the DTD below.

```
<!ELEMENT queries (#PCDATA | query)*>
<!ELEMENT query (#PCDATA | comment | prefixes | sparql | parameters | referenced-in)*>
<!ATTLIST query id CDATA #IMPLIED>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT prefixes (#PCDATA | prefix)*>
<!ELEMENT prefix (#PCDATA)>
<!ATTLIST prefix id CDATA #IMPLIED>
<!ELEMENT sparql (#PCDATA)>
<!ELEMENT parameters (#PCDATA | parametre)*>
<!ELEMENT parametre (#PCDATA)>
<!ATTLIST parametre id CDATA #IMPLIED>
<!ATTLIST parametre type CDATA #IMPLIED>
<!ELEMENT referenced-in (#PCDATA | param)*>
<!ATTLIST referenced-in file CDATA #IMPLIED>
<!ELEMENT param (#PCDATA)>
<!ATTLIST param id CDATA #IMPLIED>
```

Element	Description
queries	The root of the XML queries file
query	The element corresponding to each SPARQL query
comment	The natural language description of a query

[3] SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query

prefixes	Embeds the list of the prefixes used in the query
prefix	The element corresponding to each prefix used in the query
sparql	The SPARQL statement
parameters	Embeds the list of the parameters used in the query if it is parameterized
parameter	The element corresponding to each parameter used in the query
referenced-in	Corresponds to the list of the files where the query is called
param	The value of the parameter

Attribute	**Description**
query id	The identifier of the query, used to make a call to the query
prefix id	The identifier of each prefix
parameter id	The identifier of each parameter
parameter type	The type of each parameter
file	The path of the file where the query is called
param id	The identifier of the parameter used in the file where the query is called

Then, we developed a SAX parser enabling to explore this XML file and provide an easy way to reuse these queries.

· *By the developers:* it becomes easy to find all the queries and reuse the relevant ones for extending the functionalities of SweetWiki. Moreover, when a query is called many times in a page, it is better to have it loaded once and instantiated with different values for its parameters than to load it many times. Finally, thanks to this library, it is also easy to write new queries based on the existing ones for the non-SPARQL-expert developers.

· *By the users:* as a perspective, the descriptions of the queries might possibly be provided to the users through an interface, enabling them to easily select the relevant queries to embed into the pages they edit (so as to provide dynamic content to their pages), rather than to write SPARQL queries by themselves. This possibility would raise an issue concerning the layout of the queries results. Indeed, the results of the queries are processed and displayed by means of XSL stylesheets which have to comply with the schema of the results and the query itself (considering the information returned by the query and amongst these information which ones to display). Actually, the only way to include dynamic content into a WikiPage is to write, test and attach SPARQL queries to the pages through SweetWiki editor.

Example of a query from the SPARQL queries library

```
<query id="info_query">
<comment> This query gives information about a page: its last author, the web it
belongs to and the date of its last modification</comment>
<prefixes>
      <prefix id="wiki">http://sweetwiki.inria.fr/ontology#</prefix>
```

```
</prefixes>
<sparql> select distinct ?author ?modification where {
?page wiki:name '$param1' . ?page wiki:author ?author . ?page wiki:hasForWeb
'$param2' . ?page wiki:modification ?modification } </sparql>
<parameters>
       <parametre id="param1" type="pageName"/>
       <parametre id="param2" type="webName"/>
</parameters>
<referenced-in file="/jsp/header_wiki.jsp">
       <param id="param1">nomPage</param>
       <param id="param2">nomWeb</param>
</referenced-in>
</query>
```

This query is called in almost all SweetWiki pages so as to display automatically information about the name of the last modifier of the page and its modification date (see figure 2.2).

Figure 2.2: Use of a SPARQL query to fill widgets content

- **Query optimization**

During the first phase of SPARQL queries library building, we gathered all the queries disseminated in SweetWiki pages into the structured file described previously. The second phase consists of factorizing these queries and making their calls more optimized.

This leads to write generic parameterized queries, as shown in the following example, where we factorized three queries into one.

The SPARQL queries below are called when a user submits a tag-based query.

“ImagesTaggedWith”, “VideosTaggedWith” and “AttachFilesTaggedWith” respectively return the URIs of the images, videos and attached files uploaded on the wiki pages and annotated with the tags of the user's query.

```
<query id="ImagesTaggedWith">
<comment> Images tagged with one or more tags </comment>
<prefixes>
       <prefix id = "wiki">http://sweetwiki.inria.fr/ontology# </prefix>
</prefixes>
<sparql> select distinct ?doc ?page ?web group ?doc where {
?page wiki:includeDocument ?doc . ?page wiki:hasForWeb ?web . ?doc rdf:type
wiki:Image . ?doc wiki:hasForKeyWord '$param1' }</sparql>
<parameters>
```

```
        <parametre id="param1" type="tag"/>
</parameters>
<referenced-in file="/data/Tools/searchKeyword.jsp">
        <param id="param1"> tagurl </param>
</referenced-in>
</query>
```

```
<query id="VideosTaggedWith">
<comment> Videos tagged with one or more tags </comment>
<prefixes>
        <prefix id = "wiki"> http://sweetwiki.inria.fr/ontology# </prefix>
</prefixes>
<sparql> select distinct ?doc ?page ?web group ?doc where {
?page wiki:includeDocument ?doc . ?page wiki:hasForWeb ?web . ?doc rdf:type
wiki:Video . ?doc wiki:hasForKeyWord '$param1' }</sparql>
<parameters>
        <parametre id="param1" type="tag"/>
</parameters>
<referenced-in file="/data/Tools/searchKeyword.jsp">
        <param id="param1"> tagurl </param>
</referenced-in>
</query>
```

```
<query id="AttachedFilesTaggedWith">
<comment> Attached files other than images or videos tagged with one or more tags
</comment>
<prefixes>
        <prefix id = "wiki"> http://sweetwiki.inria.fr/ontology# </prefix>
</prefixes>
<sparql> select distinct ?doc ?page ?web group ?doc where {
?page wiki:includeDocument ?doc . ?page wiki:hasForWeb ?web . ?doc rdf:type
wiki:File . ?doc wiki:hasForKeyWord '$param1' }</sparql>
<parameters>
        <parametre id="param1" type="tag"/>
</parameters>
<referenced-in file="/data/Tools/searchKeyword.jsp">
        <param id="param1"> tagurl </param>
</referenced-in>
</query>
```

Factorized query

These three queries can be instantiated by the following generic one, in which the type of the tagged object becomes a parameter of the query.

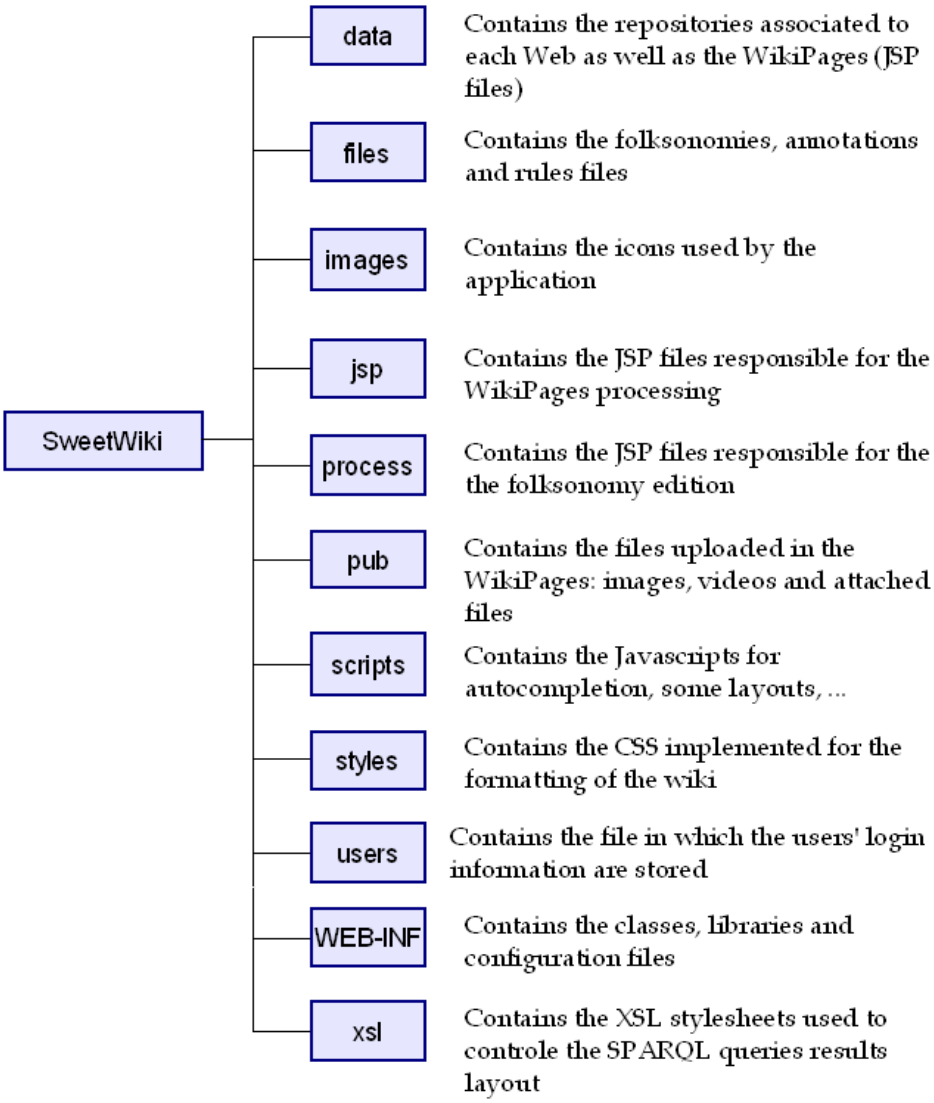
In this example, the three queries are called by the same JSP page. However, even if they were called by different pages, this would not be a barrier to optimizing them, given that the element “`referenced-in`” and its sub-hierarchy enable to define, for each file in which the query is called, the value for each one of its parameters.

```
<query id="ObjectsTaggedWith">
<comment> Objects (Images, Videos or attached Files) tagged with one or more tags
</comment>
<prefixes>
      <prefix id = "wiki"> http://sweetwiki.inria.fr/ontology# </prefix>
</prefixes>
<sparql> select ?doc ?page ?web group ?doc distinct where { ?page
wiki:includeDocument ?doc . ?page wiki:hasForWeb ?web . ?doc rdf:type
wiki:'$param2' . ?doc wiki:hasForKeyWord '$param1' } </sparql>
<parameters>
      <parametre id="param1" type="tag"/>
      <parametre id="param2" type="docType"/>
</parameters>
<referenced-in file="/data/Tools/SearchBy.jsp">
      <param id="param1"> tagurl </param>
      <param id="param2"> {Image, Video, File} </param>
</referenced-in>
</query>
```

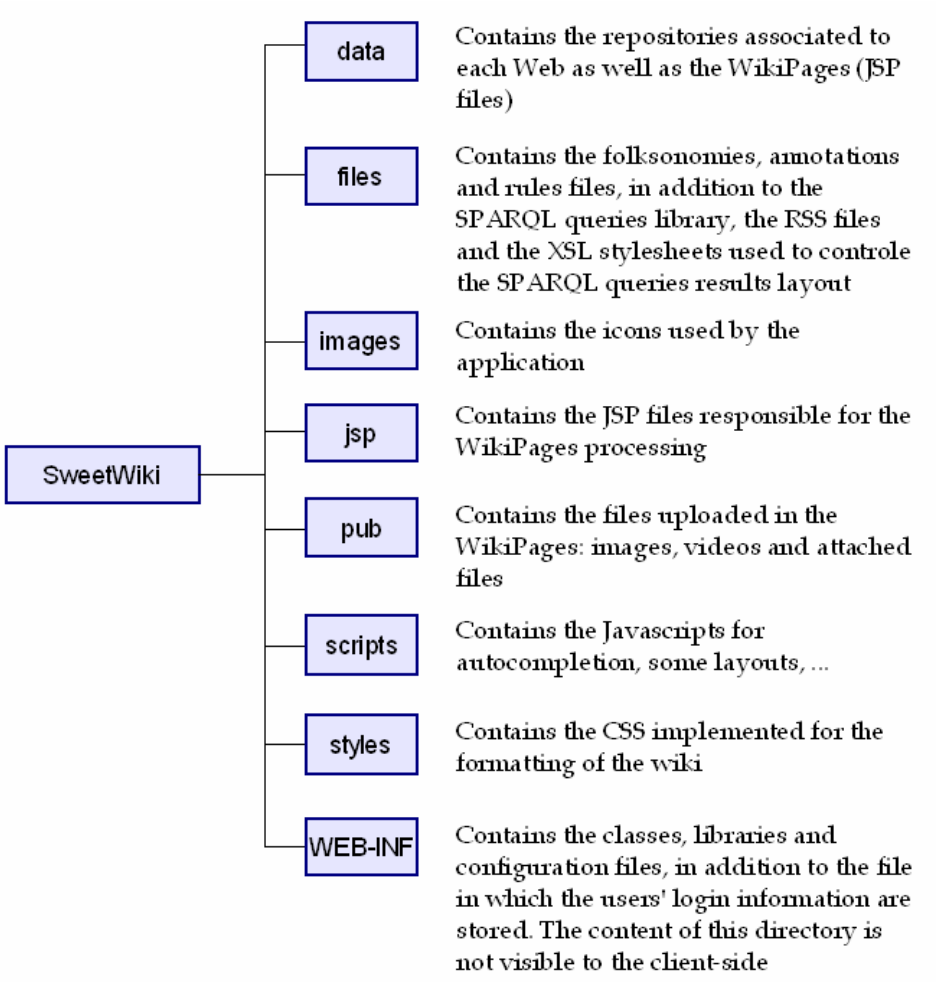
The layout of the results relies on the XSL stylesheets assigned to the query.

2.1.6 Restructuring the source tree

- **The initial structure of SweetWiki**

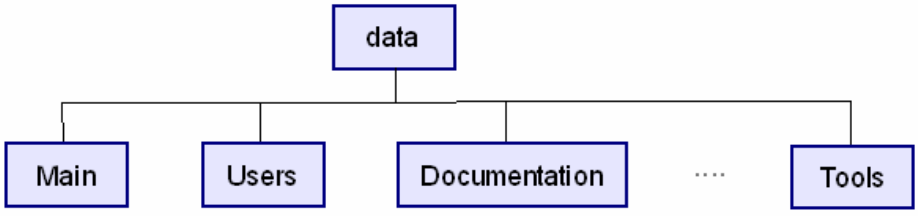


- **The new structure of SweetWiki**



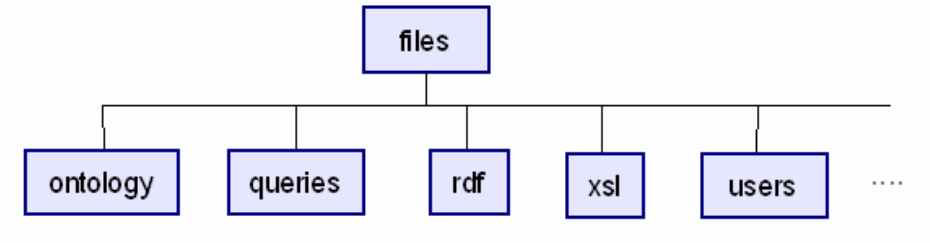
To summarize, the components of SweetWiki that are directly accessed and manipulated by the users are:

- the files stored in the “images” and “pub” repositories;
- the files stored in the “data” repository: when the users create, modify or access a WikiPage;



- the files stored in the “files” repository: the files contained in this repository can be accessed and exported. Those which are modified by the wiki users are those files belonging to the sub-directories “ontology”, “rdf”, “users”, they contain the folksonomy, the WikiPages annotations (each WikiPage is associated to an RDF file containing its annotations. These annotations include the users' tags as well as other meta-data like the page modification date, the identity of the user who last modified it, the pages that are pointed by this page, etc.) and annotations describing the wiki users.

The files that belong to the other sub-directories are used for processing. For example, there is the SPARQL queries file, which includes the SPARQL queries for tag-based search and those for widgets generation. To be displayed, the JSP WikiPages rely on XSL stylesheets for displaying the SPARQL queries results.



2.1.7 Awareness through RSS feeds subscription

The development of awareness functionalities through RSS feeds subscription is a work in progress. This work has been initiated to answer a recurrent request of the CoPs members, who want to follow more efficiently the changes occurring on the wiki content. They will be the main users of this functionality, but it will also be used by the CAKB[4] [D.IMP.04] in order to gather knowledge concerning the CoPs in SweetWiki.

It raises many issues about the choice of the queries that should be relevant for the users to subscribe to.

Indeed, any query can contain information that users consider as relevant.

$$\forall q \in SPARQLlib \mapsto r = results(q)$$

r can be constituted of:

- the set of the pages recently modified;
- the set of the pages annotated with the tags specified in *q*;
- the set of the pages containing objects (images, videos, attached files) annotated with the tags specified in *q*;
- the set of the tags that are the most used to annotate pages and objects.

Thus, a query *q* is made of the definition of the set of items I_n to be returned as results ($1 \leq n \leq nC + nP$; nC: number of classes of the queried ontology, nP: number of the properties of the queried ontology), added with various types of constraints C_m (filtering, counting, size, ...).

In addition, given that RSS feeds aim, by definition, at notifying the users of news or changes in the content of a Web site, then the queries that are used to provide these information include a temporal sorting parameter *S*.

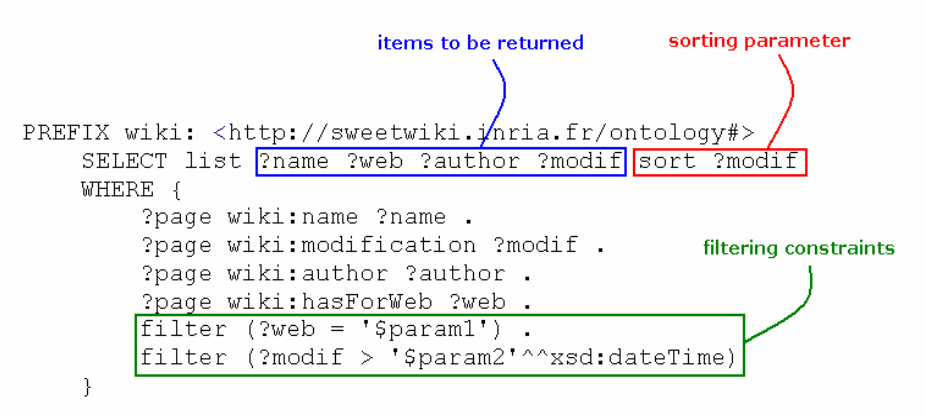
The queries that appear to be useful for the users are:

- For the users who would like to have an insight into the whole wiki content: the recent changes on SweetWiki pages;
- For the users who would like to have an insight into the content of a particular WikiWeb: the recent changes occurring on the WikiPages that belong to this particular Web;
- For the users who would like to be notified whenever a page has been annotated with a

[4] WP5, task 2.

tag that is relevant to them: the recent changes on the WikiPages that are annotated with the tags which a particular user is interested in. These tags are defined by the user through her HomePage.

Example of a SPARQL query returning the recent changes on a Web of SweetWiki



This query returns the list of the WikiPages that have been modified since the date specified with the value of `$param2` and that belong to the WikiWeb specified with the value of the parameter `$param1`.

- **From SPARQL queries to RSS feeds**

The processing of the SPARQL query, by the Corese engine [Corby et al., 2004] [D.KNO.03], generates an XML document containing the results of the query. The schema of this XML document complies with the W3C recommendation[5] for SPARQL queries results.

RSS files being XML documents, we will have to perform a transformation on the SPARQL queries results documents so as to make them compliant with the RSS format chosen. Amongst the seven existing formats for RSS files (versions 0.90 to 2.0), we choose to adopt the RSS1.0 specification[6] (RDF Site Summary), which conforms to the W3C RDF specification and is used for RDF-based applications. The transformation of the SPARQL queries results documents to RSS files will be realized by means to an XSL stylesheet.

2.2. Future development

2.2.1 First planned release

SweetWiki will be improved with:

1. Interface related functionalities:

- By making the interface of SweetWiki adapt to the user's preferred language, defined in her profile. Indeed, an annotation file will be associated to each user, and profile information, such as her preferred language will be formalized and exploited through these annotation files. These annotations will be modeled through SweetWiki structure ontology.
- Use of templates for the pages. These templates would be designed depending on the Web they belong to.

[5]SPARQL Query Results XML Format http://www.w3.org/TR/rdf-sparql-XMLres
[6]RDF Site Summary (RSS) 1.0 http://validator.w3.org/feed/docs/rss1.html

2. Improved tag-based search, enabling the submission of complex tag-based queries. The user can then submit parameterized queries which involve the attributes or relations between the tags (other than the specialization relations, already exploited in the current version of SweetWiki).

3. A better management of access rights, based on the users' annotations and modeled through SweetWiki structure ontology. This will imply to modify this ontology, and thus refactor SweetWiki structure.

4. Upgrade of SweetWiki to the latest versions of SeWeSe and Corese [Durville and Gandon, 2007].

2.2.2 Second planned release

In this release of SweetWiki, it will be possible to:

1. Perform pages refactoring, enabling the wiki administrator to rename a page, move it to another Web or delete it (only by moving it to a particular Web playing the role of a trash).

2. Edit the folksonomy in a more user-friendly way. To achieve this, we will pay attention to the editor ergonomics and provide it with drag and drop mechanisms for the folksonomy structuring (see Figure 2.3). These mechanisms will enable to completely structure the folksonomy by allowing multiple inheritance, for example, and detailed formalization (see Figure 2.4).

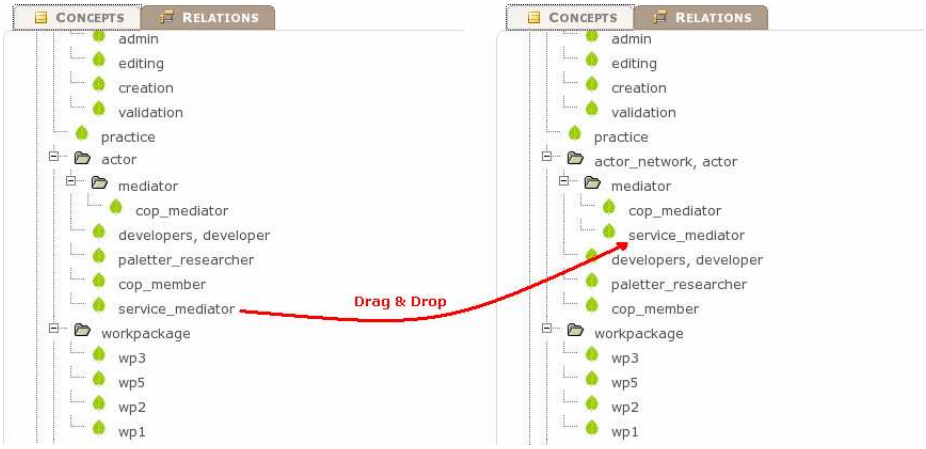


Figure 2.3: Mock-up of the ontology editor - Structuring interface

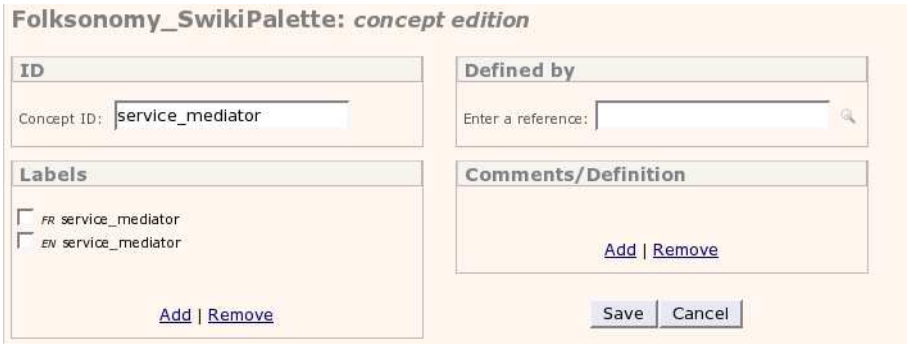


Figure 2.4: Mock-up of the ontology editor - Edition interface

3. Exploit tag versioning, that will allow us to perform temporal analysis on the use of tags.

This would help to infer knowledge on the ontology (a concept that is never used to tag the pages may be is not relevant and representative of the CoP), on the members of the CoPs (inferences can be made on the expertise of a member depending on the categories of tags he uses).

2.2.3 Perspectives

During the last months before the end of the Palette project, we will focus SweetWiki enhancement on:

1. Tagging improvement by enabling the authenticated users to tag the WikiPages without imposing them to activate the pages editor.

2. Exploring the issue of (Semi)-Automatic tags organizing. Based on the use of the tags in the wiki (webs and pages), or by some users (relying on their profiles). The issue concerning these criteria has to be deepened so as to provide an efficient way to suggest an organization of the tags that will enhance the search and thus, the learning through the use of SweetWiki.

Chapter 3 Semi-automatic Annotation and Information Retrieval from Mails: SemanticFAQ

3.1. Introduction

In this chapter, we present the SemanticFaq service developed for the @pretc CoP. It consists of a web portal offering the navigation in the @pretic discussions using semantic web technologies to ensure intelligent information retrieval from mails. We will describe our methodology for building semantic annotations from e-mails and we will present preliminary feedbacks of @pretic members.

@pretic is a community of teachers in Belgian secondary schools who are responsible for the management of computer labs (CCM) in their schools. They have been exchanging information for ten years through a mailing-list. The topics addressed in this mailing-list have a large focus. There are complaints about people and organizations, pedagogical talks and technical advice.

In order to boost @pretic's members interest in the CoP, the idea was formulated that some kind of FAQ based on the mailing-list archives would make its members more interested in the CoP. This idea came after the CoP's mediator tried to help them by giving them access to PALETTE's tools and services. The feedback informed him that @pretic needed a tool that is tailored to their needs more than general tools (such as web editors and wikis).

@pretic community is open to all teachers using Information and Communication Technologies (ICT) in Belgium during their interactions with students or for preparing their lessons. The members of this CoP communicate mainly by exchanging e-mails on a mailing list: some mails describe the ICT encountered problems while other mails suggest solutions for solving such problems.
During discussions between INRIA and the @pretic mediator for elaborating @pretic scenarios in the framework of the so-called team B, the need to facilitate navigation of @pretic members among past mails describing the ICT encountered problems or solutions to problems previously discussed was stressed by the @pretic mediator. Therefore, INRIA proposed an approach for creating semantic annotations on such mails. These annotations are based on the @pretic ontology described in [D.KNO.05] and that INRIA partly created from linguistic analysis of this corpus of mails.
The part of the @pretic ontology dedicated to ICT problems, comprises, among others, three modular ontologies, each dedicated to a specific task:

- ONTOPEDIA: an ontology describing computer components,
- OEmail: an ontology for describing mails, and
- Technical Problems Ontology: an ontology for describing computer problems.

3.2. Definition

An annotation is a graphical or textual information about a document, and placed in the document or outside in order to facilitate access, retrieval and use of this document. Semantic Annotations refer to the knowledge modeled by an ontology. These are formal annotations, as they are intended to be processed by software agents.
[Desmontils and Jacquin, 2002] divided the process of semantic annotation of documents by

an ontology in sub-processes to:
1. Identify: manual or automatic process which involves placing references to the ontology concepts in the document.
2. Instantiate: automatic or manual process to valuate the attributes of concepts using the information contained in the document.
3. Improve: manual process aimed at adding information through attributes of concepts which could not be valued in the previous phase.

3.3. Semantic annotation of mails

3.3.1 Metadata annotation

As described in D.KNO.05, the aim of Oemail ontology is the description of metadata on mails. The annotation process consists of two sub-processes [Makni et al, 2007, 2008]:

1. Mail parsing: this process consists of acquiring the mail data in its raw form and of extracting the headers and body of each mail in XML. For this purpose, we have developed a mail monitor which listens to a mail server (IMAP or POP3) and builds automatically an XML tree for each incoming mail. Our implementation is based on JavaMail API[7].
2. Mail annotation: this process involves mapping of XML elements detected in the previous phase with their corresponding concepts in Oemail and then exporting to RDF.

Table 3.1 illustrates an example of input and output of metadata annotation process[8]:

mail.xml	**mail.rdf**
<message lang="french"> <number>1212</number> <replyTo>2456</replyTo> <reply>1211</reply> <XMailer /> <Message_Id><B8B4C1CB.3750%p*****@wanadoo.be></Message_Id> <X_Sender /> <Mime_Version>1.0</Mime_Version> <Content_Type>text/plain; charset="ISO-8859-1"</Content_Type> <Content_Transfer_Encoding>quoted-printable</Content_Transfer_Encoding> <X_Priority /> <X_MSMail_Priority /> <X_MimeOLE /> <auteur>*************</auteur> <from>*********@wanadoo.be</from> <recept>Personnes Ressources Bxl</recept> <to>cefis_prbxl@lyris.det.fundp.ac.be</to> <date>Wed Mar 13 08:43:55 CET 2002</date> <subject>Re: [cefis_prbxl] La connexion ADSL concrètement</subject> <enc>ISO-8859-1</enc> <text>Merci pour toutes ces infos ! Le subnet mask est 255.255.255.0, je suppose ?</text> </message>	<rdf:Description rdf:about="http://reference.inria.fr/mails#1212"> <oem:Content-Transfer-Encoding>quoted-printable</oem:Content-Transfer-Encoding> <oem:From rdf:resource="http://reference.inria.fr/pers#*******"/> <oem:Content-Type>text/plain; charset="ISO-8859-1"</oem:Content-Type> <oem:Mime-Version>1.0</oem:Mime-Version> <oem:Message-Id><B8B4C1CB.3750%*****@wanadoo.be></oem:Message-Id> <oem:To>cefis_prbxl@lyris.det.fundp.ac.be</oem:To> <oem:ReplyTo rdf:resource="http://reference.inria.fr/mails#1211"/> <oem:Subject>Re: [cefis_prbxl] La connexion ADSL concrètement</oem:Subject> <oem:Date>Wed Mar 13 08:43:55 CET 2002</oem:Date> <rdf:type rdf:resource="http://reference.inria.fr/OEmail#ReplyMessage"/> </rdf:Description> <rdf:Description rdf:about="http://reference.inria.fr/pers#******"> <oem:E-Address>********@wanadoo.be</oem:E-Address> <oem:EAddress>*********@skynet.be</oem:E-Address> <oem:Nom>******</oem:Nom> <rdf:type rdf:resource="http://reference.inria.fr/OEmail#EmailAddress"/> </rdf:Description>

Table 3.1: Metadata annotation example

[7] http://java.sun.com/products/javamail/
[8] In table 1, we have made the person names anonymous.

3.3.2 Content annotation

For the annotation of content of a message (i.e. body of a mail), we used KIM (Knowledge & Information Management platform) [Popov et al., 2003, 2004]. KIM provides an infrastructure and services for automatic semantic annotation, indexing, and retrieval of unstructured and semi-structured content. By default KIM contains an upper level ontology called PROTON to describe high level concepts such as “entity” and “device”. To exploit KIM platform we enriched PROTON ontology by @pretic ontology notably the Component and Problem ontologies by specialization of “device” concept in “Computer science device”. We made some changes on our knowledge base in order to match the input format of KIM which uses specific properties such as "HasAlias" and "HasMainAlias" to support the multiple labels for an entity. Then we developed a semantic annotation module, which queries the KIM server through its API, gives the named entities detected for each text message and saves them in RDF.

Disambiguation

Word sense disambiguation consists of removing the ambiguity of words that have multiple meanings and giving them a unique sense. The natural language is rich with examples of ambiguities; for example, the term "cold" can refer to a natural phenomenon or a sensation of temperature depending on the context of use.

Our ontology contains such ambiguities because some terms can be attached to more than one concept. The term "Intranet" for example can be an instance of the concepts Business computing, Business tools, the Internet and Networks, while the term "Upgrade" can be instance of "Hardware" and "Software" since an upgrade can be hardware or software.

The literature describes many disambiguation algorithms, which mostly rely on machine learning techniques. We have developed our own disambiguation algorithm based on semantic vectors.

Our idea is to keep among the ambiguous concepts the closest to the context. The latter is determined by the named entities in the adjacent text.

Our algorithm generates a vector of all concepts found in the message, calculates a matrix of semantic distances, and selects the concept which has the smallest semantic distance from its neighbors.

Table 3.2 shows an example of calculation of the matrix of semantic distance with the decision. This example shows that the terms "URL" and "RNIS" are ambiguous since they are attached respectively to concepts ("World Wide Web", "Domain Name System") and ("Data Transfer Rates","ISDN"). But the context formed by the terms URL, RNIS and Modem detected in the message, enables to calculate the most likely concept. In this example, the term "URL" was annotated by "Domain Name System" and the term "RNIS" by "ISDN", the semantic global distances of which are the smallest in comparison with the concepts of context.

Remark: we use the Corese semantic distance in our algorithm [Corby et al., 2004, 2006].

Term	**Concept**	**World Wide Web**	**Domain Name System**	**Data Transfer Rates**	**ISDN**	**Modems**	**Global Distance**	**Decision**
URL	World Wide Web	0	0.76	0.76	0.77	0.76	3.0	Domain Name System
		0.76	0	0.79	0.63	0.79	**2.98**	
	Domain Name System							

RNIS	Data Transfer Rates	0.76	0.79	0	0.59	0.65	2.8	ISDN
		0.77	0.63	0.59	0	0.39	**2.3**	
	ISDN							
Modems	Modems	0.76	0.79	0.65	0.39	0	2.6	Modems

Table 3.2: Disambiguation algorithm execution

Annotator architecture

The architecture of our annotation system is shown in figure 3.1 which illustrates metadata and content annotation.

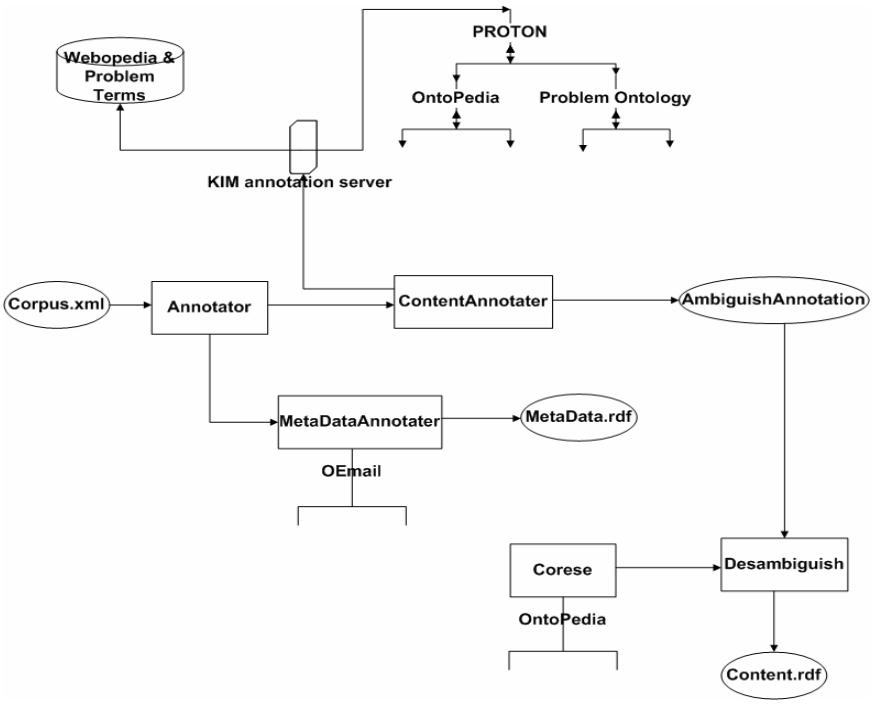


Figure 3.1: Annotator architecture

3.4. Algorithm for information retrieval from mails

The originality of our algorithm that we have designed for information retrieval from mails is that it retrieves the whole discussion feed annotated by a given concept through a unique query to the semantic search engine Corese.

The idea behind this algorithm is the use of Corese graphs to build the discussion feed. Corese implements SPARQL query language by graph homomorphism [Corby and Faron, 2007].

Our algorithm is based on five modules which are:

1. *Global_Discussion_Feed:* it builds all the paths guided by the "Oemail" property "ReplyTo" which indicates that a mail is a response to another mail.
2. *Query*: it takes a concept as input and retrieves the mails annotated by that concept.
3. Paths_restriction: having a set of mails and the Global Discussion feed, this module maintains the paths that cross mails of the set.
4. *Paths_to_Tree*: this recursive algorithm builds a tree from a given list of paths.
5. *Tree_to_XML:* it exports the tree in XML format.

At starting time of web service, the Global_Discussion_Feed module builds a list of paths containing the ReplyTo property. This is done by the extended SPARQL query with path:

```
prefix oem:<http://ns.inria.fr/palette/2007/12/OEmail#>
select $path
where {?x direct::oem:ReplyTo::$path ?y}
ORDER BY desc(pathLength($path))
```

The list of paths (Replies) is maintained in application scope as it will be used for each request.
When the user selects a concept (Concept) the algorithm executes as follows:

```
for each chosen concept       {
  List <-- Query(Concept)
  Restricted_List <-- Paths_restriction(List,Replies)
  Tree <-- Paths_to_Tree(Restricted_List)
  XML <-- Tree_to_XML(Tree)
  return XML
}
```

The string containing the XML data is then parsed in client side and printed in tree form.

3.5. Information Retrieval from Mails

3.5.1 Ontology based navigation

The @pretic ontology aims at guiding semantic search among the frequently asked questions to solve the problems faced by the CoP members. We have thus developed a Web interface enabling ontology-based navigation in the description of problems and their answers. We adapted the hyperbolic navigation [Munzner and Burchard, 1995] originally used in the navigation of websites to ontology navigation.
Hyperbolic navigation has the advantage of giving an overall view well suited to a member of the CoP that does not know the hierarchy of problems. The choice of a concept is followed by a query to the semantic search engine Corese [Corby et al., 2004] to get the messages annotated by the problem, the answers then displayed in the form of discussion feed.
In order to reduce the execution time of our Web application, the metadata are calculated and displayed when the user flies over a message.

3.5.2 Semantic portal

We have designed a web portal application to encapsulate the hyperbolic navigation through @pretic ontology. The main aim of choosing the portal architecture is to enable user profile awareness and securing access to the CoP knowledge resources.
We have plugged some semantic functionalities in the @pretic portal, for example the registration process checks whether the user is a member of the CoP by querying the metadata annotations through Corese (see figure 3.2).

Figure 3.2: Semantic management of registration

Information retrieval use case

We describe the search process by a scenario in which the CoP member seeks for printer problems. After choosing the appropriate concept (Figure 3.3 section 1), the system queries the semantic search engine Corese to get the discussion feeds annotated by the chosen concept and prints it in a tree form (Figure 3.3 section 2). Finally the solution provided by the mail is retrieved (Figure 3.3 section 3).
The @pretic portal allows the semantic navigation through *Problem* ontology and *Component* ontology described in [D.KNO.05].

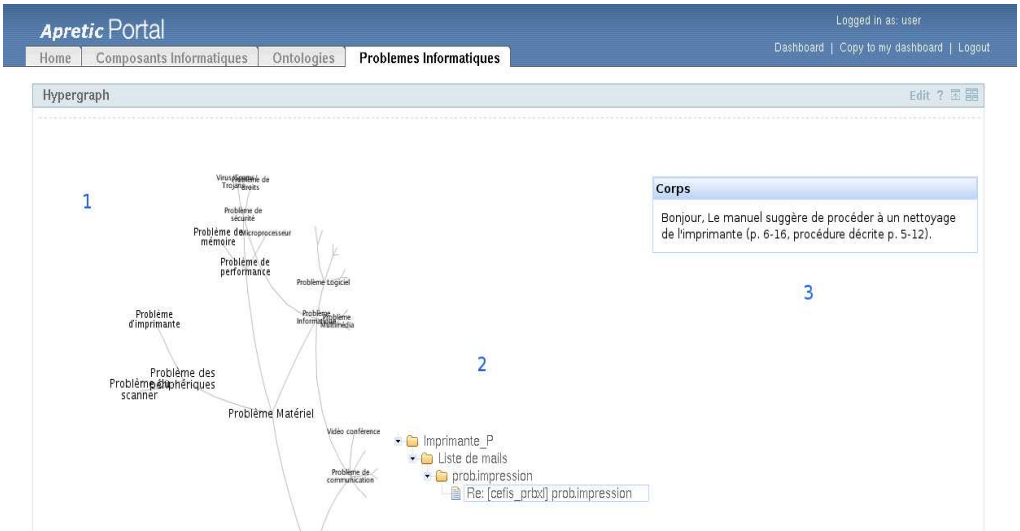


Figure 3.3: Hyperbolic navigation through the Problem ontology

Portal architecture

The portal architecture is illustrated by figure 3.4.

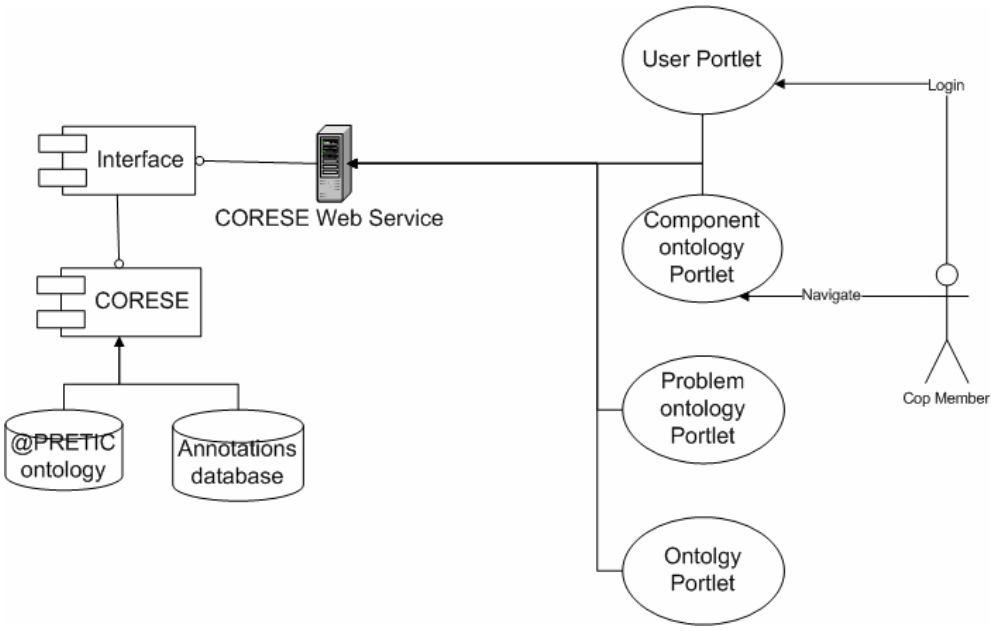


Figure 3.4: Portal architecture

We have deployed the Corese search engine into a web service which will be queried by each portlet. At startup time, the web service loads ontologies and annotations. The interface defines function calls of web service such as getting mails annotated by a given component concept or a problem concept.
The user portlet manages logging and registration actions.
The Component and Problem ontology portlets, allows the information retrieval guided respectively by the Component and the Problem ontologies.
The ontology portlet gives a tree view to both ontologies.
The member could customize his/her vision by defining a dashboard. The administrator has a more extended portlet for content management system and role management.

Used technologies

- *Portal framework:* We have chosen the JBOSS portal framework as it is an open source portal in which we are plugging semantic functionalities.
- *JSF components:* The RichFaces components which also come from the JBOSS matrix were the most suitable for the portal.
- *AJAX libraries:* To make the portal AJAX enabled for better performance by reducing server call charge, we have used the AJAX4JSF libraries which add AJAX behavior to the RichFaces components.
- *Hyperbolic navigation:* For hyperbolic navigation through ontologies, we have adapted HyperGraph applet by transforming an RDFS file containing ontology description in XML file containing nodes (Concepts) and edges (subClassOf properties).

3.6. User-centered analysis of SemanticFAQ

3.6.1 First Look

A first look at @pretic's semanticFAQ shows a refreshingly new interface concept, far from the usual, conventional looking and hard to use mailing-list archives.

@pretic's semanticFAQ, on the other hand, invites the user to walk through the messages and to learn quickly from a message to another related message.

Other services have a tendency to bore users, in a way that they often do not even consider the idea of using the service. In the best cases, they use the tool out of necessity, find the information they search after a while and immediately quit the service, without trying to dig further.

Not trying to dig further is often a mistake, since reading peripheral messages give the users a more complete and precise view of a problem and allows for a certain degree of mastery in a topic.

Figure 3.5 @pretic's mailing-list archive, a conventional mailing-list archive system – austere and of low efficiency

3.6.2 Ludicity

This brings us to a very important point about a missing feature in most of Palette's tools and services: the ludicity aspect, i.e. the fun factor of the service. Most so-called 'Web 2.0 applications' have understood it and rely on it for their users to often visit their service.

The case of a mailing-list archive, of course, is different. The stakes are not the same since it is out of the commercial realm. Nevertheless, the goal remains the same: make users actually use the service. The more they will use it, the more they will find it useful. Not only will they find the right information quickly, but a regular practice of the system will make them more efficient in their use of the tool. On the other hand, there is little incentive to use a traditional mailing-list archive regularly.

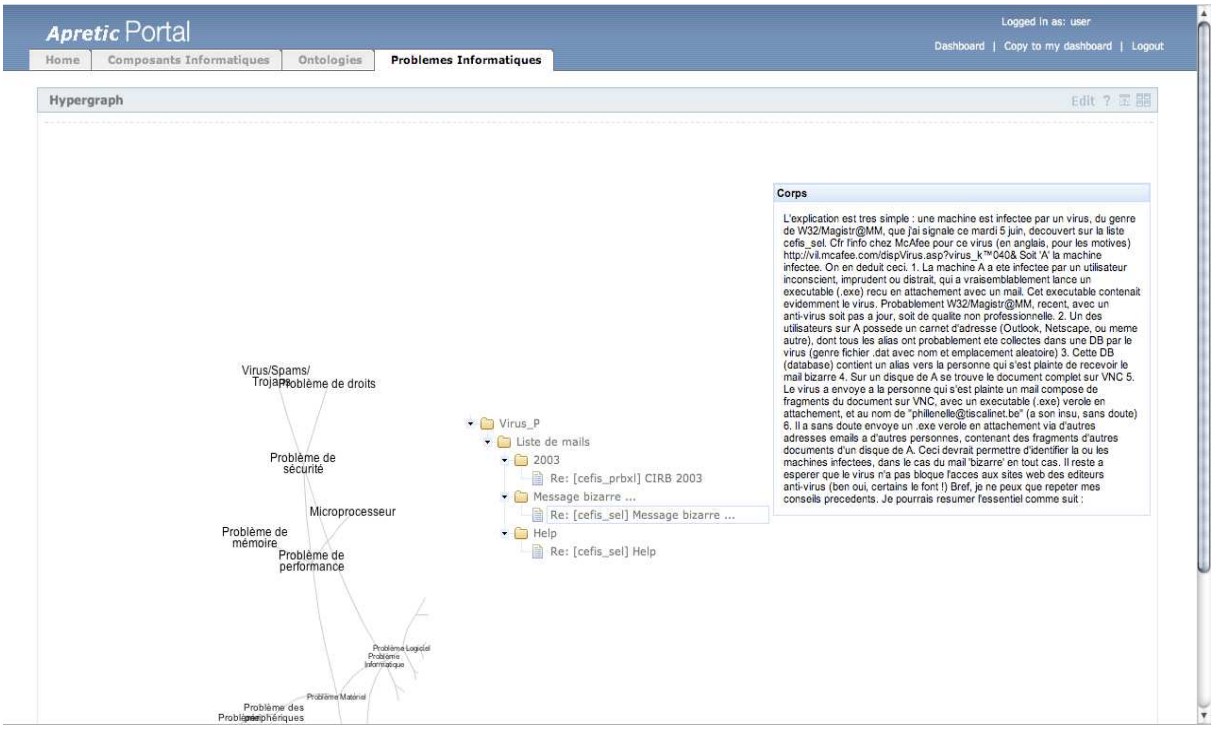


Figure 3.6 @pretic SemanticFAQ, a highly innovative way to browse mailing-list archives

We believe that a good part of the added value of this tool is its ludicity. At this point, it is important to understand that a tool or service being somewhat entertaining means that the tool or service cannot be serious. Ludicity, as we consider it in this chapter, has no pejorative meaning: the @pretic semanticFAQ is fun and entertaining to use. This entertaining aspect is a mean to attain a goal, i.e. enhancing the frequency of use of the tool and spread knowledge in a more efficient way than what other similarly-aimed services allow.

3.6.3 Ontology enrichment

The ontology used for the creation of this prototype was the semi-automatic @pretic ontology as it was extracted by the INRIA using the method described in the [D.KNO.05].

The CRIFA–ULg team worked out these shortcomings by suggesting three separate ontologies (which could be merged into a more encompassing one):

- Problems related to persons and groups (cf. the ‘Human Problem’ ontolgoy was created by CRIFA-Ulg) ;
- Learning and teaching (this ontology was also developed by CRIFA-Ulg);
- Technical problems (usage of computers) (this ontology was created by INRIA).

In the SemanticFAQ prototype developed by INRIA and tested, by CRIFA-ULg we had access to three tabs, 'Composants Informatiques', 'Ontologies' and 'Problèmes Informatiques'. Were the CRIFA ontologies used, it would have given a far more fine-grained access to the archives, because these ontologies were created specifically for the CoP. A few core @pretic members helped the CRIFA researchers to make it as accurate and precise as possible.

Please note that at this early stage of prototyping, it is perfectly acceptable not to have those ontologies already integrated in the system. It made enough sense for us to conduct our initial tests.

3.6.4 Ergonomic analysis and suggestions for future releases

The CRIFA team has developed a good experience in ergonomic analysis in the context of WP1 task 7 (usability issues) and several PALETTE tools and services have been analyzed at the time of writing this document (Amaya, SweetWiki, CoPe_It!, and eLogbook – this last one is still work in progress).

The CRIFA team and a few core @pretic members had the chance to have a brief access to a prototype of SemanticFAQ, in the short period of a few days. Therefore, the first part of this section title, 'ergonomic analysis', is maybe a bit far-fetched: we were not able to apply our established methodology in full, but used it as a guideline for a brainstorming session, which gave results which were, in our opinion, sufficient given the fact that the service we had access to was a prototype. We did not analyze it as thoroughly as other services, because some of its UI problems were due to its early stage of development, not to a bad approach at UI basics and because of time constraints. A more complete and methodological approach will follow as the service matures.

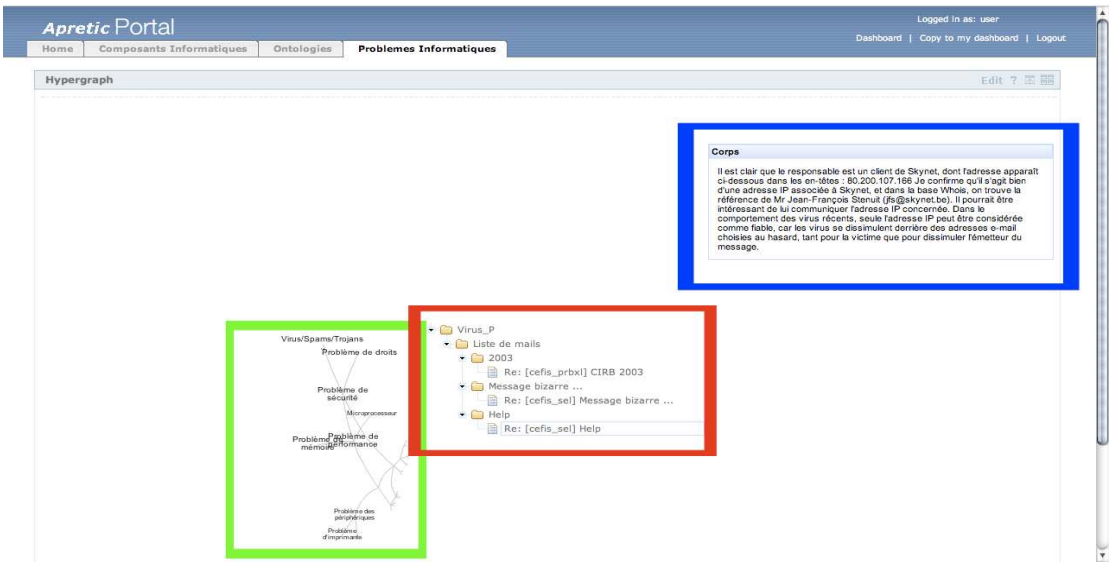


Figure 3.7 @PRETIC SemanticFAQ, preliminary ergonomic feedback

Nevertheless, here are a few things we noticed that could be improved. Not only will we notice these, but we will also suggest functionalities that the CRIFA and @pretic members thought would be useful in the future releases of the SemanticFAQ service. This section has been written from the point of view of both CRIFA researchers and CoPs members.

Screen Layout

As seen in figure 3.7, there are three zones on the main screen. They seem to 'float' in an awkward way on the screen. A better use of the screen estate would help.

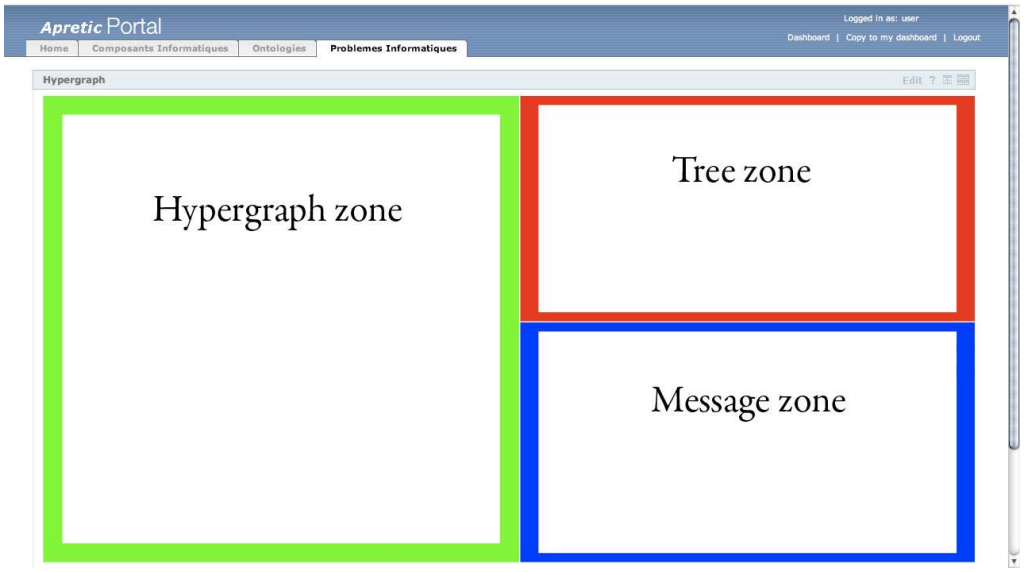


Figure 3.8 Suggested layout

We suggest to maximize the zone allocated to the Hypergraph navigator (emphasized in green on figure 3.7 and figure 3.8) at the left of the screen (roughly 50% of the zone real estate), which would allow for rich ontologies to be displayed and browsed with a greater ease of use. At the top-right, the tree zone (red) would show the threaded view of the currently browsed topic, while the selected message (blue) would be displayed in the bottom-right section of the screen.

A clear separation of the three zones, through unequivocal graphical clues would greatly enhance the user experience. Please note that our red/green/blue rectangles are not intended to suggest these visual clues but only to help the reader to understand our notices and advice. By comparing figures 3.6 and 3.7, you can notice the 'floating' feeling we discussed before: the three zones look like they are never anchored at the same place. A clear parting of the zones, and a better anchoring of them at fixed zones would lessen the cognitive load of the service.

The threaded view

The threaded view, as it stands now, is a bit confusing: some messages have a 'folder' icon, other have a 'page' icon. There is a logic to that: the messages with a 'page' icon are messages that have not answers (end-of-thread messages). It is nevertheless confusing because it is generally accepted that a folder icon does not contain text itself. Moreover, it is redundant, because the tree metaphor already provides the needed visual clues, in a more satisfying way.

On a more general note, it seemed that all the messages were not present in the database. A lot of messages seemed to be orphaned. Threading a mailing-list archive a posteriori is not always easy and we believe that at this stage of the process, the 're-threading' process was not finished.

The message view

The message view is very important since it displays the actual searched content (the other zones are just means to find the message). It would be helpful to:

- show the title of the message in the title bar (instead of 'Corps');
- show the date and time of the message in the title bar (the date can sometimes be very useful in determining the relevance of a message);
- allow to print the message and possibly the whole thread;
- restore the messages' formatting (no carriage returns make the messages hard to read);
- let the user magnify the font size, which is a bit small by default;
- emphasize the clicked/searched (see below) term, for instance by making its background color slightly yellow;

Various suggestions

- Provide a keyword search: in some cases, the user may know exactly what he searches in the archive. In these cases, a classical free entry keyword search would help, especially if the result would not only affect the message zone, but also the Hypergraph zone, placing the message in its context in the ontology.
- Allow a two-terms (or more) ontology search: instead of just being able to search according to one term taken from the ontology, for instance, 'ADSL', let the user chose a second term, for instance, 'server'. In this example, only the messages tagged with 'ADSL' and 'server' would be displayed, automatically discarding a lot of messages discussing ADSL connections but having nothing to do with servers.
- Add a rating system to messages: every identified user could be allowed to rate messages (from 0 to 5 stars, for instance), so that in future searches, a particularly relevant message would have more weight that an irrelevant one and would be displayed first.
- Add sorting options: sorting messages by date is often a good idea. Sometimes, sorting them by author is even more useful (if a user knows that another user is an expert in a given topic, for instance). Sorting them by relevance (thanks to the rating system we just discussed) is also interesting.

3.7. Related Work and Further Work

In this chapter, we presented an approach for semi-automatic annotation on the mails exchanged through a mailing list of the @pretic CoP.
Related work on annotation of texts offers numerous annotation tools detailed in the synthesis offered by [Uren et al., 2006]. Such annotation tools can be compared according to:

- The nature of suggestions of the annotation tool: instances of concepts, values of their attributes, instances of relations, metadata on named entities, etc.
- The storage of annotations: in the document source or in an annotation server.
- The language for representing annotations: XML for Cafetiere [Black et al., 2005], RDF for Annotea project [Koivunen, 2005], SAMOVAR [Golebiowska et al., 2001], OntoWatch [Cao et al., 2004], MeatAnnot [Khelif et al., 2005], or QBLS [Dehors et al., 2005].
- The language for handling the reference ontologies used for creating the annotations.
- The type of processing: manual process vs. automatic vs. semi-automatic process.
- The automation techniques used: Natural Language Processing (NLP) using linguistic techniques vs. machine learning.
- The fact that resources are internal or external.
- The nature of annotated resources: static Web pages, dynamic Web pages, multimedia resources (images, video, audio) as Vannotea or M-OntoMat-Annotizer ….
- The evolution of annotation according to resource evolution.

In the current state of the art, we distinguish the following approaches:

- Purely manual editors (such as OntoMat Annotizer [Bloehdorn et al., 2005], the Mangrove system [McDowell et al., 2003] or Cohse annotator [Bechhofer et al., 2003]).
- Manual, collaborative editors such as or Annotea [Koivunen, 2005] the W3C project for collaborative annotation that proposes Amaya [Quint et al., 1997] and Vannotea.
- Use of manually written rules or wrappers in order to capture known patterns for the annotations: e.g. Melita [Ciravegna et al., 2002] enables the user to write rules based on regular expressions; Cafetiere [Black et al., 2005] is a rule-based system for generating XML annotations.
- Supervised systems learning from sample annotations marked up by the user: e.g. tools such as Ontomat, MnM [Vargas-Vera et al., 2002] and Melita include Amilcare [Ciravegna et al., 2003] that learns how to annotate the documents by generalizing the user's annotations.
- Unsupervised systems using various strategies to learn how to annotate without user's supervision: e.g. Armadillo [Chapman et al., 2004] or Ontomat-Pankow relying on the Pankow approach [Cimiano et al., 2004].
- Annotation service providers, offering such services on demand for users browsing non-annotated resources: e.g. Magpie [Dzbor et al., 2003].

In [Azouaou, 2006], the author distinguishes cognitive annotations (intended to humans) and computational annotations (intended to programs).
Here are our previous annotation systems:

- Samovar [Golebiowska et al., 2002], a system that uses Natural Language Processing (NLP) techniques for semi-automatic creation or enrichment of an ontology from textual comments of a database and for generation of RDF annotations of these textual comments with the concepts thus created.
- MeatAnnot [Khelif et al., 2005], a system that uses and implements NLP techniques to identify instances of concepts and relations of a reference ontology, in textual scientific articles in biomedical domain in order to generate RDF annotations.
- QBLS [Dehors et al., 2005], an e-Learning system based on semi-automatic generation of RDF annotations of learning resources after association of concepts of ontologies (pedagogical ontology, domain ontology) to styles used in text processing systems such as MS Word or Open Office.
- OntoWatch [Cao et al., 2004], a system dedicated to watch, relying on several algorithms using an ontology in order to complete a user's query sent to Google and offering automatic generation of RDF annotations after the analysis of the structure of the Google results.

In comparison with this related work, the originality of our work of semantic annotation and information retrieval from mails comes from the use of mails as a source of knowledge [Makni et al, 2007, 2008]. The specificity of the work presented in this chapter is the fact that we work on a degraded corpus of mails, the poor linguistic quality of which makes harder the use of NLP tools.
[Zhong et al., 2002] and [Sakurai and Suyama, 2005] offer mail mining but their main objective is email classification in order to filter spams for example as [Jongwan et al., 2007]

As a conclusion, mail mining seems potentially very useful in the framework of CoPs for several reasons: (a) mails can be a main knowledge source in some CoPs such as @pretic, (b) if the number of already existing mails of a CoP is huge, they cannot be handled manually and

an automatic approach as offered by mail mining is necessary. So we have tried to offer a generic approach so as to ensure its reusability for other CoPs communicating through mails or discussion forums. We will extend this work in order to propose a semantic FAQ exploiting the benefits of ontologies, semantic annotations and reasoning offered by our semantic search engine Corese.

Chapter 4 KM LinkWidget

4.1. Introduction

The knowledge space of CoP contains the resources produced and used by the CoP in its activities, the content of this knowledge space is composed of resources of different natures: many kinds of documents, discussion, ... These resources were modeled in the Resources section of the O'CoP ontology [D.KNO.02]. From a computational point of view, these resources are produced, manipulated and stored using different Palette services. When the number of these resources grow, it becomes difficult for CoP members to find the relevant resources for their activities, if they are localized or associated to services other than the one they use. A service allowing the CoP members to easily find/access their resources across the services they use will be quite useful when the number of the resources become important.
In the current work, we focus on a particular case of link between resources, namely the link between:

(i) the "static" resources of the CoP stored in a repository and
(ii) the "dynamic" resources constituted by the discussion between CoP members in a forum.

The KM LinkWidget will enable CoP members to link resources stored in a repository with conversations in a discussion forum, by using semantic annotations of the resources and the discussions. It will use the basic KM services described in [D.KNO.04] to achieve the basic tasks of KM (annotation, retrieval, ontology management).

The widget will be implemented as navigator extension[9], and will manage two kinds of annotation bases, the first is local and will contain the personal annotations of the user, the second is common to the CoP and will contain the public annotations of the CoP members.

In this chapter, we will first define some of the notions used to describe KM LinkWidget, we will then propose an architecture of the service and expose some implementation choices, and finally we will present a scenario of use of the service by some Palette CoPs.

4.2. Definitions

4.2.1 Repository and Semantic Repository

Repository

A *repository* is a central place where resources are stored, archived and maintained in an organized way. The resources can be of any type: texts, multimedia documents, ... In Wenger's classification of useful tools for a CoP [Wenger, 2005], repositories can be either generic or dedicated to a specific task (e.g. a LMS is a repository dedicated to learning resources). In this work, we consider only *web-repositories* which are repositories with a web interface.

[9] We choose to implement KM LinkWidget as a Firefox (http://www.mozilla.org/) .

The most common organization of resources in a repository reproduces the structure of a file system: hierarchical organization with directories containing sub-directories and files. For a web-repository we define the *access schema* as the static schema describing this organization. Figure 4.1 show an example of such a schema.

```
<!ELEMENT kmlw:repository (kmlw:folder*)>
<!ATTLIST kmlw:repository
   xmlns:kmlw CDATA #FIXED "http://www.inria.fr/kmlw/2007/11/repository"
>

<!ELEMENT kmlw:folder (kmlw:file*)>
<!ATTLIST kmlw:folder
   name CDATA #REQUIRED
   url CDATA #REQUIRED
>

<!ELEMENT kmlw:file (kmlw:attribute*)>
<!ATTLIST kmlw:file
   name CDATA #REQUIRED
   size CDATA #REQUIRED
   type CDATA #REQUIRED
   url CDATA #REQUIRED
>

<!ELEMENT kmlw:attribute EMPTY>
<!ATTLIST kmlw:attribute
   name CDATA #REQUIRED
   value CDATA #REQUIRED
>
```

Figure 4.1 Example of access schema

Semantic Repository

A *semantic repository* is a repository where the organization of resources is entirely or partially achieved by the use of semantic annotations of resources. In this case, the access schema of the repository is composed by the ontology used to annotate the resources and the static schema describing the non semantic structure of the repository if there is one.

SweetWiki (see. Chapter 2) can be seen as semantic repository of resources (the WikiPages and the attached files), the static organization of resources consists of the system of WikiWebs (that can be viewed as directories) and the semantic organization is achieved by tagging the resources.

4.2.2 Forum

A forum is a service that hosts the *interactions* between users, these interactions are organized into *discussions* and each discussion is composed of posts. Each user can read other users posts and add his/her own contribution.
The forum is defined by its schema; Figure 4.2 shows an example of such schema:

```
<!ELEMENT forumPage (discussionEntry*) >
<!ATTLIST forumPage subject CDATA #REQUIRED>

<!ELEMENT postEntry (label,user,date,body?,attach?,postEntry*) >
<!ATTLIST postEntry id ID #REQUIRED >label,user,date,body?,attach?
```

Figure 4.2 Example of a forum schema

A *Semantic forum* is forum where the discussions and posts are semantically annotated w.r.t. an ontology.

4.3. How it will work

KM LinkWidget will allow users to manage the annotations associated to the resources in the repository and the forum, and will use these annotations to establish links between these two kinds of resources. The annotations can be:

(i) private, when the user annotate a resource or a discussion for his own usage, and do not share it with the other members of the CoP.
(ii) public, when the annotation is shared with the other CoP members.

The service manage two kinds of annotation bases, a *personal annotation base* for private annotations, and a *CoP annotation base* which contains the public annotations of all the CoP members. The CoP annotation base is stored in the repository, and the personal annotation is stored locally for each user.

The KM LinkWidget is a service integrated to a web browser, when the user is browsing the associated web-repository and web-forum, the widget uses the annotation bases to enrich the displayed pages with the deduced associations between resources and discussions. It also allow users to add semantic annotations for a set of predefined operations, for example, when uploading a document in the repository or when posting a new contribution in a discussion, ...

Let us see how the different components will work:

4.3.1 Annotation of resources

When the user is browsing the repository, s/he can annotate any resource; two cases are thought out:

- *operation-associated annotation*: when the user adds/modifies a resource to the repository the KM LinkWidget will propose an interface to annotate this resource, some of the annotations (e.g. Ownership of the resource, Version, Date, ...) are calculated automatically. In addition, user can add his/her own annotations (e.g. keywords, subject, ...)
- *free annotation*: the user can also update/modify the existing annotations of a resource.

4.3.2 Annotation of discussions

When browsing the forum, the user can annotate the discussions or the posts, the annotation can either be associated to an operation in the forum or free:

- operation-associated annotation: when starting a new discussion or reply in an existing one, the KM LinkWidget display an annotation interface to annotate the discussion or the post. Some of the annotations are calculated automatically (e.g. Ownership, Date, Subject, ...). The user can also add his own annotations.
- free annotation: the user can update/modify the existing annotations of a discussion or a post.

4.3.3 Collecting external annotations

If the browsed site (repository or forum) contains embedded annotations (e.g. Microformats[10], RDFa[11], ...) KM LinkWidget extract them and add/use them as annotations for the resource or the discussion.

4.3.4 Retrieving/Calculating Links

The main operation of KM LinkWidget is the retrieval of the associations or links between the resources and the discussions. This calculation is based on a set of queries, these queries being defined w.r.t. the CoP ontology, and can be parameterized for each user. Here are some examples of queries that can be used:

- retrieve the resources which have the same subject as a discussion;
- retrieve the resources owned by the user in the repository and having the same subject as the discussion containing a post;
- retrieve the discussions related to a resource.
- ...

4.3.5 Displaying Links

The calculated links are displayed directly in the viewed page on the browser, or in a sidebar during the navigation. For example, a list of tags associated to a discussion can be directly displayed in the viewed page, while some other information, like the list of the resources owned by the user related to the current discussions can be viewed in the sidebar. The user has the possibility to configure the displayed queries, and also how s/he wants the results to be displayed.

4.4. Architecture

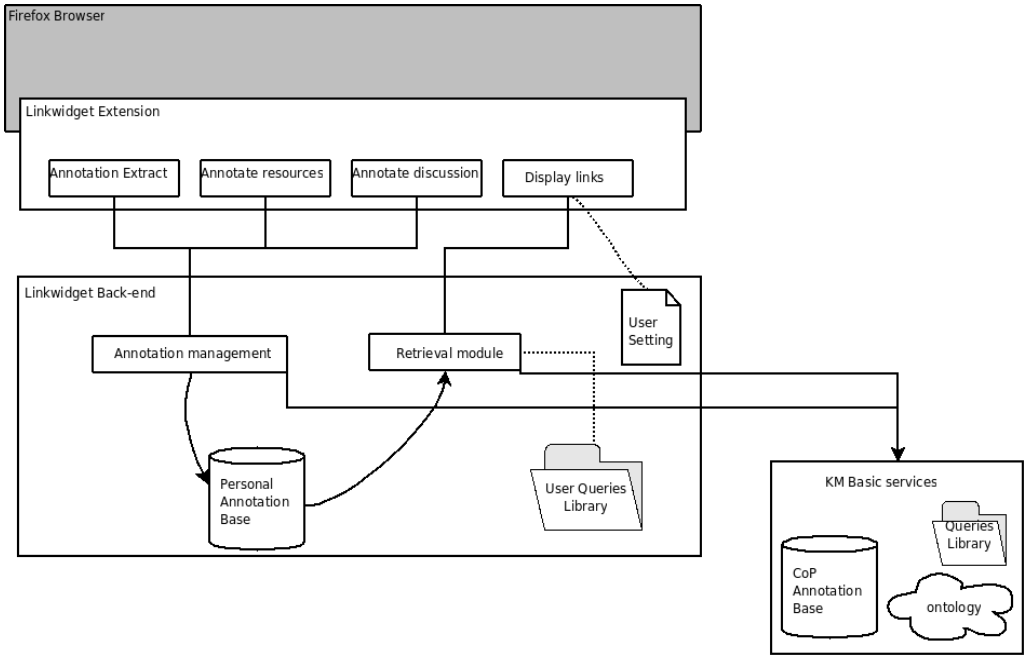


Figure 4.3 KM LinkWidget architecture

[10] http://microformats.org/wiki/
[11] http://www.w3.org/TR/xhtml-rdfa-primer/

Figure 4.3 shows the architecture of the KM LinkWidget. The visible part for the user is composed of a set of Javascript components used to gather information from the user in order to annotate resources and discussions, or directly from the displayed page to extract embedded annotations, and a component to display the retrieved links/associations into the current page or in the sidebar.

The back-end of LinkWidget is composed of two modules:
- an annotation module: to add/remove annotations from the user annotation base, and publishing public annotations added by the user to the CoP annotation base.
- a retrieval module: to calculate the links/associations using user annotation base and CoP annotation base, and the query libraries.

These two modules use the basic KM services based on Corese/SeWeSe described in [D.KNO.04].

4.4.1 Used technologies

The technologies that will be used in the implementation will be:
- **Firefox**: the application platform
- **XUL**: the extension's user interface language
- **Javascript**: the front-end and extension's scripting language
- **Java**: the back-end programming language
- **Jetty**: the embedded web server
- **SemServices**: for the basic KM operations
- **Grrdl**: for the extraction of RDF annotations

4.5. Scenario of use by CoPs

4.5.1 Did@cTIC

The Did@cTIC CoP has three main motivations for using the KM LinkWidget:

1. to facilitate the capitalization of pedagogical resources which are created during small discussion groups exchanging about the teaching practices and experiences;
2. to facilitate the enhancement of both "explicit" and "tacit" knowledge appearing during the interactions about different teaching subjects;
3. to help the production of new documents (with DocReuse) that will be used in particular by the CoP members or during other teaching courses given by the Did@cTIC trainers.

LinkWidget can help CoP members to answer these needs as follows:

1. During an interaction about the teaching practices and experiences, the user can be provided with all the resources related to the discussed subject that are available in the repository of the CoP. The cross links from these resources to a previous discussion can also help the CoP members during their exchange.

2. The informal knowledge of the CoP contained in the different discussions can be more easily accessed, using the links between this informal knowledge and the formalized knowledge of the CoP included in some reference documents available in the repository.

3. The production of new documents using DocReuse can be enhanced by the retrieval capabilities of LinkWidget.

4.5.2 Learn-Nett

The Learn-Nett CoP members use the forum of moodle for their discussions. They also produce documents in SweetWiki and store them in BayFac. LinkWidget can help them to make a link between their discussions and the resources produced in Sweetwiki and those stored in BayFac. Moreover, the two repositories have in this case semantic capabilities that will enable the use of LinkWidget with a minimal annotating effort from the users.

4.6. Conclusions

In this chapter, we presented a cross-services retrieval and annotation widget, which aims to help CoPs members to exploit more efficiently two of knowledge channels available in the CoP knowledge space: interactions and stored resources.

This work takes advantage of the new technologies of the Web 2.0, together with semantic web technologies. It has been inspired by related work: for instance, (i) PiggyBank [Huynh et al., 2005] a browser extension that allows user to extract semantic data from web pages, (ii) Diggo[12] a social annotation tool, that allows users to collect and share their bookmarks, and to annotate web pages.

The particularity of our work is the use of semantic annotations to link resources across different knowledge channels, in the context of a CoP. It takes advantage of the previous development of WP3, mainly (i) the O'CoP ontology used as a reference ontology to annotate resources, and (ii) the basic KM services to achieve basic operations of knowledge management: annotation, retrieval.

[12] http://www.diigo.com/

Chapter 5 Faceted Classification and Search Service: BayFaC

5.1. Functional update

This part describes the new functionalities developed or being developed in response to the questions and remarks of CoP members. The first part explains the different roles we have defined inside BayFac. We will, then, see that with the connection process due to the role management, we can specify some documents to be accessible only to known users (private access) and others that can be read by an anonymous visitor (public access). In a third part, we will see the way to manage facets within BayFac. In the fourth part, an installer developed in order to facilitate the diffusion of BayFac will be described. Finally, we will explain some changes in the facet model.

5.1.1 Role management

BayFac includes now the management of roles. This increases security, allowing us to manage access to resources and actions, according to different kinds of users. Currently, anonymous or normal users only have a consultative access: they can browse and search the set of public documents. Login is mandatory to access private documents, and also to access other functions than search.

As illustrated in Figure 5.1, one single role is allowed per user. Roles are hierarchically organized. The different roles are defined by the action(s) allowed, and each role inherits from the actions allowed for its parent. Here is the list of defined roles, in increasing rights order:

- Search: Allows us to use the search functionality only.
- Classify: Allows us to use the search and classification functionalities.
- Facet management: Allows us to manage facet spaces and facets.
- Ontology management: A link to Generis (the ontology manager detailed in the chapter 3 of the [D.KNO.04]) is provided in addition to the previous rights, allowing making changes in the ontology and more generally the knowledge base used by BayFac.
- Administration: An administration tab is provided to access the previous functionalities. In addition, it allows user management and some system functions, as the “Empty cache” function.

Figure 5.1 presents the user management interface. The screen is composed of the users list to see who is defined and with which role. The manager can change the rights of each person by clicking on his login or simply delete him with the basket button. A little form to add a user is also provided. In this form, the manager has to specify the login and the password for connection but also the role of this user. The possible role list is deployed in the print screen of figure 5.1.

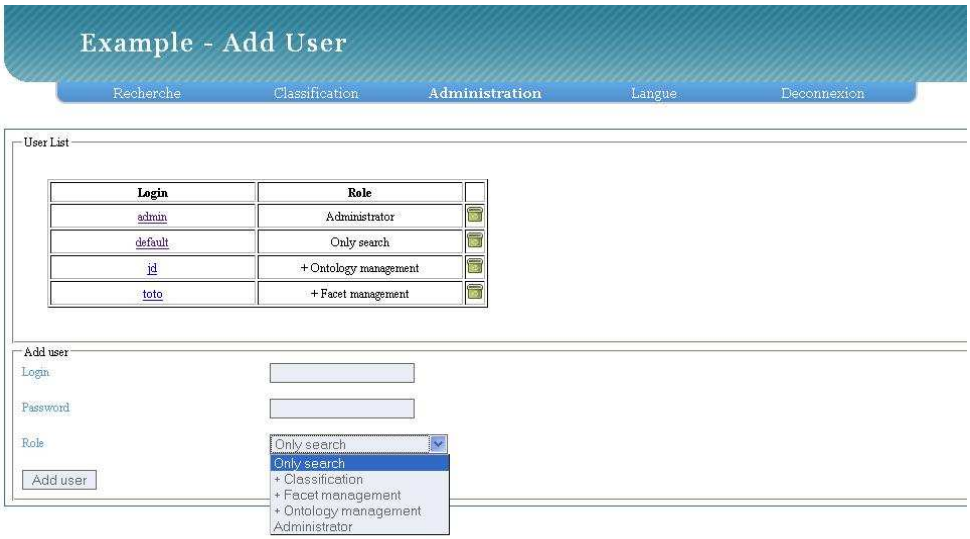


Figure 5.1: BayFac user management

5.1.2 Private document management

Once a role has been given to a user, this role allows private documents access. When a user adds a document, this user can tell if it is a public or a private one, as illustrated by figure 5.2. Public documents are visible by anyone who comes on this BayFac without connection requirement (login and password). The private documents access needs a user connection.

Figure 5.2: Private document management

5.1.3 Facet management

The facet spaces, as well as the facets, exploit concepts of the CoP's ontology used for document or other entity description and classification. One of the most important assets of BayFac is its genericity regarding the facets: the facets can be chosen from the CoP's ontology and their function can be defined regarding the facet model explained in [D.KNO.04]. That is why we have provided an interface to manage[13] the facets and the facet spaces.

[13] To manage the facets, the user must have a role 3 or more (cf. point 5.1.1. Role management).

Facet space configuration

The facet space defines the classification scheme (i.e. the facets) in which the user will index his documents or other entities and perform search on them. As explained previously in [D.KNO.04], a facet space is defined by a class of the ontology, corresponding to the entities to be classified. As displayed in figure 5.3, the user can choose this class from the list of classes of the CoP's ontology. Once the class is chosen ("Groupe" in figure 5.3), BayFac must be informed if a file has to be attached to the instances added.

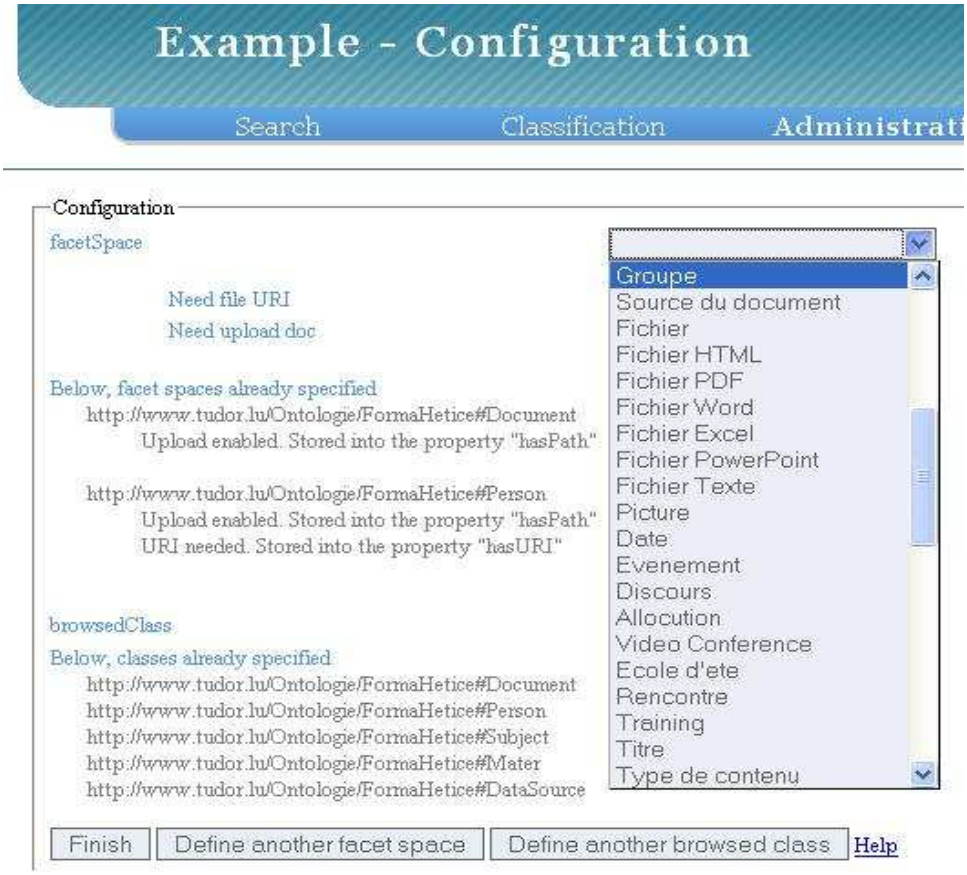


Figure 5.3: Choose a facet space

The user specifies the need of upload of document or/and the need of the specification of an URI (see figure 5.4). This seems easy to understand for search & classify documents but it is lesser for a person for whom no document is needed. This eventual attached document should be specified by a property in the business ontology. If the user does not know which property it is, a default one could be selected via the "I don't know" box as illustrated in the picture above.

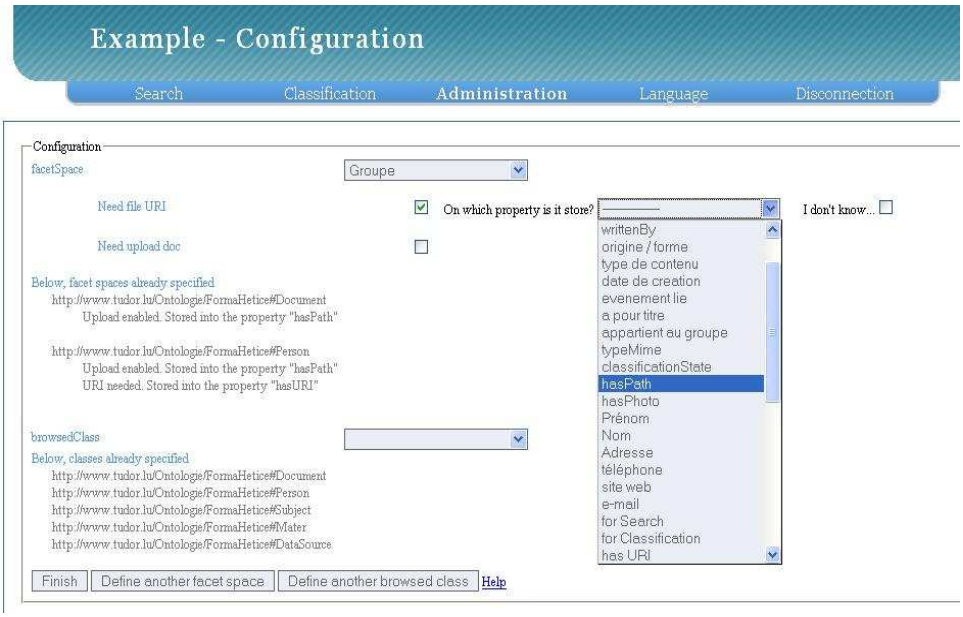


Figure 5.4: Choose a property to be specified when a document is attached

To validate its choice, the user has to click on a button in the bottom of the page. If he clicks on the "Define another facet space" button, the information he has specified about the facet space will be stored and the interface reappears with the new configuration as we can see for the Document concept already specified in figure 5.4. In the same way if he clicks on the "Define another browsed class" button, the information he has specified about the browsed (by the semantic browser, see section 5.2 of the [D.KNO.04]) class. Until the user clicks on the "Finish" button, the result of its actions will be written on the configuration screen as illustrated in figure 5.4: "Below, facet spaces already specified…".

Facet configuration

Once the facet spaces are defined, the user can define the facets for each facet space. A listing of the facets is reachable, as illustrated in figure 5.5, for users with at least a role 3.

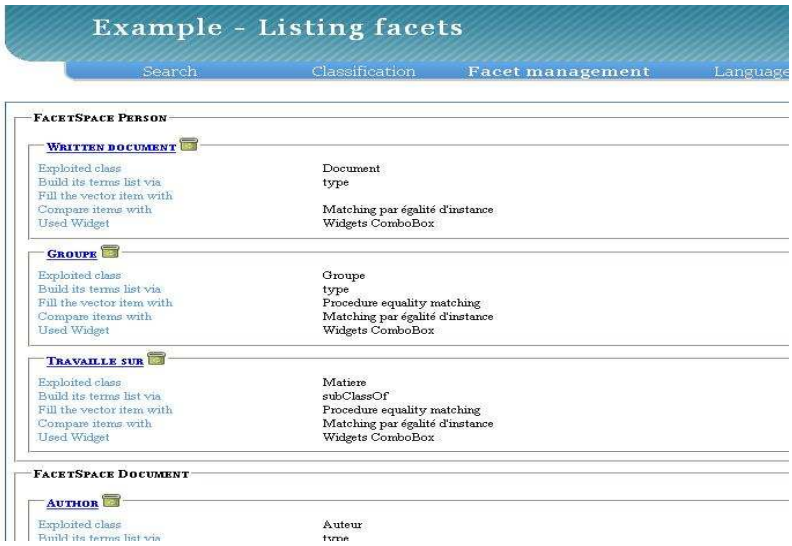


Figure 5.5: Listing of the defined facets

In this interface, the user can choose to delete, add or update a facet, as shown on the printed screen in figure 5.5.

The label of each facet is a link to its modification page. The basket next to the label is a deletion button to remove the facet.

Creation or update of a facet can be done using the same form, which is illustrated in figure 5.6.

Figure 5.6: Facet definition form

In this interface (figure 5.6), the user can define the facets according to the facet model (see section 5.2 of the [D.KNO.04]). The model must be well known by the user. A detailed documentation has to be done to describe it in order to help the user to define a facet by himself without help from BayFac's developers.

5.1.4 Installation and configuration

BayFac is provided with an installation system. When the Generis and the BayFac directories are deployed (simply by copy) on the concerned Web server, the user can call, with a browser, the root of his BayFac repository. The installation interface (see figure 5.7) appears then and the three-step configuration of BayFac begins: First, the installation of the CoP's ontology, then the facet spaces configuration, and finally the facet configuration.

Figure 5.7: Installation form

As illustrated in figure 5.7, in the first step, the user needs to have the file of the CoP's ontology he wants to exploit with BayFac. He needs to specify:

- The name he wants to see appearing on "his" BayFac, typically, the name of the CoP.
- The model namespace is the namespace of the ontology.
- The local namespace is the namespace of the instances.
- The RDFS file of the ontology.

BayFac creates a module for itself in Generis with the facet ontology and the CoP's one (specified thanks to the form of figure 5.7). If it is done correctly, the user is directed to the second step: the facet space configuration to choose what he wants to search & classify (cf section "facet space configuration"). Finally, the user can create the facets for each facet space as explained in the previous section. After these steps, BayFac is properly configured and ready to be used.

5.1.5 Facet Ontology improvement

The use of BayFac by PALETTE CoPs (namely Form@Hetice and LearnNett) has generated new requirements. For some of them, a solution is found more logically by modifying the facet model, than by developing a new function in the application.

In the first part, we explain how a facet can specify a property of the facet space concept in the ontology. And in the second one, we present the new facet model.

Property specification

The classification process in BayFac can specify some properties of the concerned facet space class (for example, the "has Author" property of the facet space "document"). The case occurs when the facet exploits a class directly linked to the class of the facet space by a property. For instance, it is the case for the facet "Author" of the facet space "Document": a property "WrittenBy" links directly the two concerned classes. In a first step, BayFac searches the concerned property by searching a property linking the two concerned classes. This way to proceed finds its limitation when several properties link the two concerned classes. In order for the system to be able to identify exactly the property which is relevant for the facet, one as to specify explicitly this property in the facet model. The retained solution is to add a property to the facet class: "directPropertySpecification", which is used with the "exploit" property, defining thus the exact path defining a facet: the facets values are linked to the facet's exploited class linked to the facet space domain class by the property specified in

"directPropertySpecification". It is to note that this solution is considered as temporary and some redesign might still occur in the future to insure the genericity of the model.

New Facet model

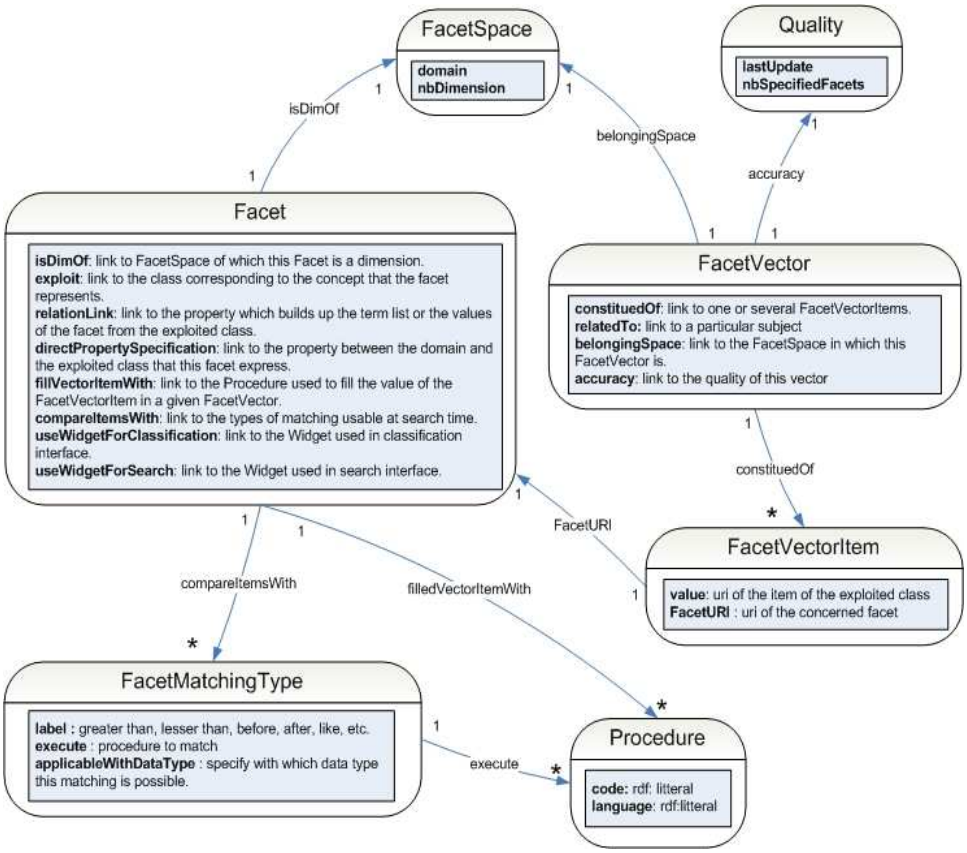


Figure 5.8: The new facet model

5.2. Interface improvement

The interface improvement is a continuous process. Since its first presentation, and thanks to the different remarks the developer's team received on it, BayFac improves its interface to become more and more practical to use.

The CoP mediator and some CoP members provided feedback on two main functionalities: search and classification.

5.2.1 Classification evaluation

For the classification, no problems were found for the following functionalities:

- Add a new document
- Download a document
- Facets specification
- Document deletion
- Correction of a classification
- Add a choice in the facet

The problems found (calendar closing, and update of items) in this classification part are described below. For each problem, the correction or the solution for a future correction is explained.

- In the date facet, the calendar does not close when clicking somewhere else; it needs the user click on “close”. This problem comes from the CSS (Cascading Style Sheets) of the calendar used. It is not a serious problem. It will be fixed for the next release.
- In the facet, it is possible now to insert a new item but it is still impossible to delete or modify one of them without using Generis. To solve this, new buttons will be put next to the “add item” button. When an item is selected, the user can click on the deletion button or on the update button. For the update button, a popup in the same style as for the add one, will be opened with the instance to modify. This functionality will be fixed for the next release.

5.2.2 Search evaluation

For the search, three main problems were identified:

- In the key word search, BayFac was searching only in label and the description of the document, not in the content and not in the facet specified. This is fixed now. The key word search can find the entered word in the content, in the facets specified and of course in the label and description.
- A green arrow was utilized before to display or not a facet in the search. Its utility was not obvious and its functioning not perfect. As a solution, it as been deleted to alleviate the interface.
- In the listing of resources matching with a search query, it will be useful to have link to see the classification of each resource and another to a preview of the document instead of having to download it to see what it is. These links will be provided in the next release.
- In the semantic browser, a button was inserted to return to the last search performed. Indeed, the history of the browser is not of any help to go back to the search when using the semantic browser, we just need a shortcut button to go back to the last search. This shortcut has been added.

5.3. Bayesian Performance feedback and evaluation

Automatic classification will be based, in this case and for these facets, on Bayesian techniques. Bayesian classifiers are used to provide automatic categorization. They learn first from manual classification made by users, and then propose automatic categorizations of documents in each facet once there is a sufficient amount of documents processed in a supervised manner to enable correct predictions. The benefits for the users are to have new documents automatically classified according to known useful categories.

In order to measure the Bayesian performance of BayFac, we have to have a feedback of the proposed classification’s quality. Each facet according to which the document is classified is managed by a single classifier. As a consequence, each classifier has to be evaluated independently from the others. For that, two main steps are required. Firstly, the data concerning the proposed classification’s quality must get back and then must be processed.

1. There are two ways to get information back. The first one is done without the user’s knowledge and the second one requires his evaluation:
    - When a classification is proposed by a Bayesian classifier (for one facet), the

user must valid or not the proposition. If the user considers that the classification is not appropriated for the facet in question, he can modify it. In this case, the modification is recorded in a log file. There is one single log file for all classifiers.

●The second solution is to ask the user to evaluate the quality of the classification done by the classifier. For instance, for the Form@HETICE CoP and for a facet giving the classification of a document according to its content type, the classifier proposes “pedagogical”. Users validate the classification proposition by changing if it is necessary and by rating it. A mark from 0 (this is not the good classification) to 3 (this is exactly the good classification) is. 1 stands for quite false classification and 2 stands for a quite good classification but the proposition is even though changed. In the same way, each rate of each classifier is recorded in a log file. Now, the log file must be treated.

2. As there are two ways to feed Bayesian performance data back there are also two ways to handle them:

●For the first manner, we have a log file with records of modification or not of the classification when it is validated. In order to see the evolution of good classifications for one classifier (related to its facet learning), we make a graph with number of good classifications per 10 propositions. The chart shows thus the learning speed of a classifier. The learning speed is not be related to time but to the number of documents processed. The more a classifier process documents and propositions are validated, the more it will learn and become efficient.

●The second manner adds to the learning speed, the quality of its learning. Actually, instead of building a graph with the number of good classifications, we use the average of 10 classification’s rates.

5.4. Interoperability

5.4.1 REST Web Services

REST is an approach for Web services development based on the basic Web principles. It is meant to be more lightweight and coarser grained than SOAP Web services. As it is defined in [Fielding, 2000] it cannot be considered as an architecture but as an architectural style, thus we searched a mean to have a concrete set of guidelines to implement our RESTful Web services. We have found ROA (Resource Oriented Architecture) as a well-defined set of best practices for the design of RESTful Web services, described in [Richardson and Ruby, 2007]. We tried to follow these practices as strictly as possible in the implementation of our services, and we used their methodology to turn our requirements into read/write resources (see [Richardson and Ruby, 2007] pp 147-148).

The main functionalities we have implemented as RESTful services are the search and classification features of BayFac. The first resources considered were thus, Document and Facet. Then we added the Facet vector, as the expression of the classification of the document relative to the facets. Subsequently we added all the resources on which depend these first resources, like FacetSpace and FacetItem, which are described below. These RESTful Web services are implemented in our current BayFac architecture as a new presentation tier, beside our classic web presentation tier, because it was difficult to adapt our current framework, based on MVC, which models web pages in terms of actions, with the REST approach, where every resource must be named with a noun and not an action verb. This independence from the legacy permitted us to better fit to the REST principles, and we checked all our resources

to verify if they met safety[14] and idempotence[15] criteria on their supported methods.

Service description

The service described here constitutes the first version of the Web services provided by BayFac. All this service resources together with possible actions on them are defined in greater details in the Web service WADL, most notably the messages XML schema (See Annex 1). We use hereafter the URI Template notation [Gregorio et al., 2007]. For example, if we consider the following URI Template:
http://somehost/users/{user_id}
{user_id} is called a path variable and as such, if we set {user_id} to "john_doe", the final URI is:
http://somehost/users/john_doe

The following HTTP methods are used in the rest of this document in their common sense:

- ·GET: read the content of a resource (one of its representations)
- ·POST: create a resource (resource factory)
- ·PUT: update a resource
- ·DELETE : delete a resource

Moreover the HEAD and OPTIONS methods are supported:

- ·HEAD: GET returning only headers
- ·OPTIONS: list of supported methods on a given resource

The right management system implemented in BayFac is also supported through the Web service interface. Clients are authenticated through the classical HTTP Basic Authentication mechanism, by passing the credentials through HTTP headers in each request requiring it.
An alternative mean to pass the credentials in a request, is to embed them in the URI, for example:
http://login:password@somehost/users/
Rights are managed at the resource and method level.

Error reporting is made through the standard HTTP mechanism, and status codes. As described in the WADL file, each request can generate an error. In such a case, the HTTP status code is set consequently in the response, accompanied by an XML message containing the textual representation of the error in its body.
(For the complete list of HTTP/1.1 status codes, please see
http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)

5.4.1.1.Context

The *fs* resource handles facet spaces. It is the root of our web service, all other resources are subordinate to this one. It is coherent with the BayFac model, since all operations are done in the context of a facet space. Actually, *fs* corresponds to the domain property of the FacetSpace class in the facet ontology and denotes the class of classified entities (or classification subjects). The path variable {fs_id} of the fs resource allows to specify a specific facet space (a particular instance). This resource is read-only.

[14] GET and HEAD methods are *safe*, since they do not modify the dataset.
[15] GET, HEAD, PUT, DELETE methods are idempotent, one ore more of these operations should give an identical result.

Resource	Method	Input	Output
/fs	GET	/	The list of facet spaces
/fs/{fs_id}	GET	/	A representation of a facet space

5.4.1.2.BayFac Search

The search is done through the *instance* resource, which refers to the classification subjects of a facet space. For example, if the facet space "Document" is defined on a Document class (i.e. specifying that Document instances are the subjects of classification), the resource *instance* is the list the instances of that Document class that are present in the system. Two query string parameters are optional for the search:

·keywords: a list of comma separated keywords

Example: keywords=palette,project

·facet_items: a list of semicolon separated couples. Each couple is constituted of the id of a facet and a value (it can be the id of a facet item, or a string). The two elements of the couple are separated by a comma.

Example:
facet_items=i1239831232313,12/09/07;i34238942333900,i3129847384783

All id mentioned in the facet_items parameter can be found through the resources facet and facet_item (see below).

Resource	Method	Input	Output
/fs/{fs_id}/instance/	GET	Query string parameters: - keywords - facet_items	List of instances matching the search criteria
/fs/{fs_id}/instance/{instance_id}	GET	/	The description of an instance

5.4.1.3.BayFac Classification

In order to classify an instance, it must be added to the BayFac database. This can be done with the POST method on the *instance* resource. It also supports methods PUT and DELETE to respectively update and delete an *instance*. If the instance refers to an external document, its URI has to be mentioned in the input message.

Facet descriptions can be read through the *facet* resource. If facets are constituted of discrete elements, these ones are available through the *facet_item* resource for each facet.

The classification is made through the creation of a facet vector, associated to an *instance*. The facet vector defines for a given *instance* a list of couples, each couple refers to a facet id and a value (possibly a facet item).

Resource	Method	Input	Output
/fs/{fs_id}/instance/	POST	A representation of an instance	A validation or error message
/fs/{fs_id}/instance/{instance_id}	PUT	A representation of an instance	A validation or error message
/fs/{fs_id}/instance/{instance_id}	DELETE	/	A validation or error message
/fs/{fs_id}/facet/	GET	/	List of facets
/fs/{fs_id}/facet/{facet_id}	GET	/	A representation of a facet
/fs/{fs_id}/facet/{facet_id}/facet_item/	GET	/	A list of facet items
/fs/{fs_id}/facet/{facet_id}/facet_item/{facet_item_id}	GET	/	A representation of a facet item
/fs/{fs_id}/facet_vector/	GET	/	A list of facet vectors
/fs/{fs_id}/facet_vector/	POST	A representation of a facet vector	A validation or error message
/fs/{fs_id}/facet_vector/{facet_vector_id}	PUT	A representation of a facet vector	A validation or error message
/fs/{fs_id}/facet_vector/{facet_vector_id}	DELETE	/	A validation or error message

Remarks

The currently provided Web services permit to access the basic functionalities of BayFac, namely search and classification. For the moment several resources are not available for write operations; this will be possible after a homogenization with Generis Web services.
Moreover, it seems also complicated to describe in XML the data contained in a knowledge base, because XML schema are not flexible enough. It could be interesting to study the possibility to use semantic web technologies, for example by exchanging RDF messages and by describing the service with WADL and RDF Schema. This would have the advantage that the WADL file could be dynamically generated from the knowledge base.

5.4.2 Data import

This functionality comes from a Form@HETICE CoP's request, which agreed to use BayFac only with facilities to import its already existing documents in it (as it concerned nearly 200 documents).

The first kind of import that will be available is import via RSS feeds. This is available for documents in Form@HETICE.

In next versions of BayFac, other datasources than RSS feeds might be proposed, as for example mail or forum. Datasources can be defined as kinds of elements to import (RSS, mails, forums) containing several documents not always well structured.

Documents import from Form@HETICE

As mentioned previously, Form@HETICE documents import is realized via RSS feeds.
RSS (Really Simple Syndication or Rich Site Summary) is an XML Web Feed Format allowing sharing content among different websites.

Example of RSS feed, used in the context of Form@HETICE:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- generator="FeedCreator 1.7.2" -->
<rss version="2.0">
<channel>
<title>Form@Hetice documents RSS Export</title>
<description>RSS export of Form@Hetice docu-
ments</description>
<link>http://www.stecrifa.ulg.ac.be/formahetice/index.php?opt
ion=com_docman</link>
<lastBuildDate>Tue, 27 Nov 2007 10:49:44
+0100</lastBuildDate>
<generator>FeedCreator 1.7.2</generator>
<item>
<title>1 ecolEte intro</title>
<link>http://sim.tudor.lu/palette/FormaHetice/uploads/1_ecolE
te_intro.ppt</link>
<description>1 ecolEte intro</description>
<category> École d'été</category>
</item>
<item>
<title>Guide utilisation site</title>
<link>http://sim.tudor.lu/palette/FormaHetice/uploads/Guide_u
tilisation_site.pdf</link>
<description>Guide utilisation site</description>
<category>Guide utilisation</category>
</item>
</channel>
</rss>
```

How does it work?

Two steps are necessary. The first step is done on Form@HETICE current website side: A RSS feed is generated every night in Form@HETICE with the last documents added.

This RSS feed is calculated by querying the database in order to get for each of the last added documents, their name, link, category, and description. Then the RSS feed is constructed, and given back to the user as a result to his query. The RSS feed generated is compatible with the RSS 2.0 standard and thus can be used in any current web aggregator.

The second step is the following. Every night, after the generation of the RSS feed, a batch in BayFac interrogates it. The RSS is parsed in order to get information concerning each document to import. For each of those documents, the following steps are realized:

1. The physical document is copied in BayFac
2. An instance of the document is created with the label (title), comment (description), classification state (none at the moment), and the posted date (the date of creation in BayFac)
3. The content of the document is parsed in order to generate keywords (the label and comment are also parsed for keywords)
4. The physical link is added to the Document instance in BayFac.

The documents are then saved in BayFac, and corresponding keywords are added. Users can then access to them for classification.

Chapter 6 Conclusions

This deliverable detailed the new developments of SweetWiki (collaborative creation services) and BayFac (document classification and search services) and presented two new complex KM services: SemanticFAQ for semi-automatic annotation of a corpus of e-mails and information retrieval from such e-mails and KM LinkWidget for linking resources and the discussions.

SweetWiki, SemanticFAQ and BayFac have been implemented and tested by at least one CoP: SweetWiki by several CoPs such as TFT CoP (see deliverables [D.KNO.04] and [D.KNO.05]), Bayfac by Form@Hetice and Learn-Nett CoPs, SemanticFAQ by @pretic CoP. Notice that this deliverable reports the results of @pretic CoP's user-centered analysis on SemanticFAQ portal and the suggestions of Form@Hetice CoP for BayFac. The KM Link Widget has not yet been implemented but once implemented, it will also be tested by Did@actic and Learn-Nett CoPs.

As further work, we will achieve the implementation of KM Link Widget and improve the three other complex KM services, as indicated in the corresponding chapters: interface and tagging capabilities for SweetWiki, semantic navigation and generation of answers to queries about e-mails for SemanticFAQ, experimentations of BayFac with Form@Hetice and LearnNett CoPs and study new services (knowledge evolution and knowledge evaluation services).

A CoP-oriented evaluation of each of these complex KM services will be performed with at least one CoP.

Last, we will study the interoperability of some KM services with information services (e.g SweetWiki and Amaya) and collaboration services (e.g. SweetWiki and CopeIt!) or other services (e.g. BayFac and CAKB, service browser, or value evaluation service).

Bibliography

[Azouaou, 2006] Azouaou F.. Modèles et outils d'annotations didactiques collectives pour des mémoires de formation. Thèse de l'université Joseph Fourier, 2006.
[Bastien and Scapin, 2001] Bastien, J. M. C., & Scapin, D. L., Évaluation des systèmes d'information et Critères Ergonomiques. In C. Kolski (Ed.*), Systèmes d'information et interactions homme-machine. Environnement évolués et évaluation de l'IHM. Interaction homme-machine pour les SI* (Vol. 2, pp. 53-79) Paris : Hermes.
[Bechhofer et al., 2003] Bechhofer S., Goble C., Carr L., Hall W., Kampa S., De Roure D. COHSE: Conceptual Open Hypermedia Service. In Annotation for the Semantic Web, S Handschuh and S. Staab (eds), IOS Press, Amsterdam 2003.
[Bloehdorn et al., 2005] Bloehdorn S., Petridis K., Saathoff C., Simou N., Tzouaras V., Avrithis Y, Handschuh S., Kompatsiaris Y., Staab S., Strintzis, M.G. Semantic Annotation of Images and Videos for Multimedia Analysis, In Proceedings 2nd European Semantic Web Conference (ESWC 2005), Heraklion, Greece, 29 May-1 June 2005.
[Black et al., 2005] Black W.J., McNaught J., Vasilakopoulos A., Zervanou K., Theodoulidis B., Rinaldi F. CAFETIERE Conceptual Annotations for Facts, Events, Terms, Individual Entities, and Relations, Parmenides Technical Report TR-U4.3.1, 11 Jan, 2005.
[Cao et al., 2004] Cao, T.-D., Dieng-Kuntz, R., Fiès B. An Ontology-Guided Annotation System for Technology Monitoring, IADIS International WWW/Internet 2004 Conference, Madrid, Spain, 6-9 October 2004.
[Chapman et al, 2004] Sam Chapman, Alexiei Dingli, Fabio Ciravegna: Armadillo: harvesting information for the semantic web. SIGIR 2004: 598
[Cimiano et al., 2004] Cimiano P., Handschuh S., Staab S. (2004) Towards the self-annotating web. In Proceedings 13th International World Wide Web Conference, (WWW 2004), May 17-22, 2004, New York, NY.
[Ciravegna et al., 2002] Ciravegna F., Dingli A., Petrelli D., Wilks Y. (2002) User-System Cooperation in Document Annotation based on Information, In 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), 1-4 October 2002 – Sigüenza, Spain.
[Ciravegna et al., 2003] Ciravegna F., Wilks Y. Designing Adaptive Information Extraction for the Semantic Web in Amilcare, in S. Handschuh and S. Staab (eds), Annotation for the Semantic Web, in the Series Frontiers in Artificial Intelligence and Applications, IOS Press, Amsterdam, 2003.
[Corby and Faron, 2007] Corby O., Faron-Zucker C.. Implementation of SPARQL Query Language based on Graph Homomorphism, In Proc. of the 15th International Conference on Conceptual Structures (ICCS'2007), Sheffield, UK.
[Corby et al., 2004] Corby O., Dieng-Kuntz R. and Faron-Zucker C.. Querying the Semantic Web with Corese Search Engine, in: Proc. of the 16th European Conference on Artificial Intelligence (ECAI'2004), Prestigious Applications of Intelligent Systems, Valencia, Spain, de Mantaras, Saitta (editors), August 22-27 2004, p. 705-709.
[D.IMP.04] Sire, S., Karacapilidis, N., Karousos, N., Gateau, B. , Naudet, Y., Vagner, A. , Watrinet, M.-L., Chiu Man Yu, M., Geneves, P.. Updated version of guidelines for development - Towards a Palette Service Platform Architecture. Deliverable D.IMP.04, Palette FP6-028038, November 2007.

[D.KNO.02] Tifous, A., Dieng-Kuntz, R., Durville, P., El Ghali, A., Evangelou, C., Giboin, A., and Vidou, G.., CoP-dependent ontologies. Deliverable D.KNO.02, Palette FP6-028038, March 2007.
[D.KNO.03] El Ghali, A., Corby, O., Dehors, S., Dieng-Kuntz, R., Durville, P., Evangelou, C., Gandon, F., Giboin, A., Latour, T., Plichart, P., Tifous, A., and Vidou, G. Specification of the CoP-oriented Knowledge Management Tool offering basic CoP-adapted KM services. Deliverable D.KNO.03, Palette FP6-028038, August 2006.
[D.KNO.04] El Ghali, A., Dieng-Kuntz, R., Giboin, A., Tifous, A., Gateau, B., Naudet, Y., Vagner, A., and Watrinet., M.-L., Basic CoP-oriented Knowledge Management Tool offering basic CoP-adapted KM Services. Deliverable D.KNO.04, Palette FP6-028038, October 2007.
[D.KNO.05] Dieng-Kuntz, R., Denis, B., El Ghali, A., Gateau, B., Lebails, J.-D., Makni, B., Naudet, Y., Peeters, R., Rieppi, S., Tifous, A., Vandeput, E., and Vidou, G., Extensions of OCoP Ontology. Deliverable D.KNO.05, Palette FP6-028038, January 2008.
[Dehors et al., 2005]Dehors, S., Faron-Zucker, C., Giboin, A., Stromboni, J.-P. (2005). Semi-automated Semantic Annotation of Learning Resources by Identifying Layout Features, in: Proceedings of AIED'2005 International Workshop on Applications of Semantic Web Technologies for E-Learning, Amsterdam, July 2005.
[Desmontils and Jacquin, 2002] Desmontils E. and Jacquin C.. Annotations sur le web : notes de lecture. In Journées scientifiques de l'AS Web sémantique, CNRS, 2002.
[Durville and Gandon, 2007] P. Durville and F. Gandon. Sewese: Semantic web server. In Proc. of WWW2007 Developers track, 2007.
[Dzbor et al., 2003] Dzbor, M., Domingue, J. & Motta, E. Magpie: Towards a semantic web browser. Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, US, October.
[Fielding, 2000] Fielding, Roy Thomas, *Architectural Styles and the Design of Network-Based Software Architectures*, Doctoral dissertation, University of California, Irvine, 2000 (http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm)
[Golebiowska et al., 2002] Golebiowska, J., Dieng-Kuntz, R., Corby, O., Mousseau, D. Samovar: using ontologies and text-mining for building an automobile project memory, in Knowledge Management and Organizational Memories, p89-102, Kluwer Academic Publishers, Boston, 2002
[Gregorio et al., 2007] Gregorio, Hadley, Nottingham, Orchard, *URI Template, IETF Internet-Draft*, v02, 2007
[Huynh et al. 2005] David Huynh, Stefano Mazzocchi, and David Karger. Piggy Bank: Experience the Semantic Web Inside Your Web Browser. International Semantic Web Conference (ISWC) 2005.
[Jongwan et al., 2007] Jongwan K., Dejing D., Haishan L. and Donghwi K. Constructing A User Preference Ontology for Anti-spam Mail Systems. In *Proc. the 20th Canadian Conference on Artificial Intelligence (Canadian AI'07)*. LNCS/LNAI 4509, pp. 272-283.
[Khelif et al., 2005] Khelif, K., Dieng-Kuntz, R., Barbry, P. Semantic web technologies for interpreting DNA microarray analyses: the MEAT system, in: Proceedings of the 6th International Conference on Web Information Systems Engineering (WISE'05), New York, USA, Springer, Lecture Notes in Computer Science, November 2005.
[Koivunen, 2005] Koivunen M-R., Annotea and Semantic Web Supported Collaboration, Invited talk at Workshop on User Aspects of the Semantic Web (UserSWeb), at European Semantic Web Conference (ESWC 2005) Heraklion, Greece, 29 May 2005.
[Makni et al, 2007] Makni B., Khelif K., Dieng-Kuntz R. & Cherfi H., Création semi-automatique d'une ontologie et des annotations sémantiques pour une liste de diffusion d'une communauté de pratique, Atelier Ontologies et Textes associé à la 7ème conférence Terminologie et Intelligence Artificielle TIA'07, 8-10 Octobre, Sophia Antipolis, France.

[Makni et al, 2008] Makni B., Khelif K., Dieng-Kuntz R. & Cherfi H., Utilisation du Web Sémantique pour la gestion d'une liste de diffusion d'une CoP, 8èmes journées francophones Extraction et Gestion des Connaissances, INRIA Sophia Antipolis Méditerranée, January 29 - February 1st, 2008.

[McDowell et al., 2003] McDowell, L. Etzioni, O., Gribble, S.D., Halevy, A., Levy, H., Pentney, W., Verma, D., Vlasseva. S. Mangrove: Enticing Ordinary People onto the Semantic Web via Instant Gratification In Proc. 2nd International Semantic Web Conference (ISWC2003), Sanibel Island, Florida, USA, October 20-23, 2003.

[Munzner and Burchard, 1995] Munzner, T. and Burchard, P. Visualizing the structure of the World Wide Web in 3D hyperbolic space. In *Proceedings of the First Symposium on Virtual Reality Modeling Language* (San Diego, California, United States, December 13 - 15, 1995). VRML '95. ACM, New York, NY, 33-38.

[Popov et al., 2003] Popov B., Kiryakov A., Ognyanoff D., Manov D., Kirilov A., and Goranov M. Towards Semantic Web Information Extraction, In ISWC2003 Human Language Technologies Workshop. 20 October 2003, Florida, USA.

[Popov et al., 2004] Popov B., Kiryakov A., Ognyanoff D., Manov D., Kirilov A., and Goranov M.. Kim a semantic platform for information extraction retrieval. Nat. Lang. Eng., 10(3-4) :375–392, 2004.

[Quint and Vatton, 1997] Quint V., Vatton I. An Introduction to Amaya, W3C NOTE 20-February-1997, (http://www.w3.org/TR/NOTE-amaya-970220.html accessed on 28 July 2004)

[Richardson and Ruby, 2007] Richardson, Leonard & Ruby, Sam, *RESTful Web Services*, O'Reilly, 2007.

[Sakurai and Suyama, 2005] Sakurai S. and Suyama A.. An e-mail analysis method based on text mining techniques. Appl. Soft Comput., 2005: 62-71

[Uren et al., 2005] Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegana, F. Semantic annotation for knowledge management: Requirements and a survey of he state of the art. Journal of Web Semantics, n. 4:14-28, 2005.

[Vandeput and Ledent, 2007] Vandeput, E. and Ledent, M. Usability report: SweetWiki. Internal report, Palette FP6-028038, June 2007.

[Vargas-Vera et al., 2002] Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A. & Ciravegna, F. MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. In Gómez-Pérez, A. & Benjamins, V. R. eds, Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, Proc. of the 13th International Conference, EKAW 2002, Sigüenza, Spain, October 1-4, LNAI 2473, p. 379-390.

[Zhong et al., 2002] Zhong, N., Matsunaga, T., and Liu, C.. A Text Mining Agents Based Architecture for Personal E-mail Filtering and Management. In *Proceedings of the Third international Conference on intelligent Data Engineering and Automated Learning* (August 12 - 14, 2002). H. Yin, N. M. Allinson, R. Freeman, J. A. Keane, and S. J. Hubbard, Eds. Lecture Notes In Computer Science, vol. 2412. Springer-Verlag, London, 329-336.

Appendix A BayFac WADL

A.1. Main WADL file

```
<?xml version="1.0"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://research.sun.com/wadl/2006/10 wadl.xsd"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:bay="bayfac"
        xmlns:bfs="bayfac:facet:show"
        xmlns:bfi="bayfac:facet:index"
        xmlns:bfis="bayfac:facetItem:show"
        xmlns:bfii="bayfac:facetItem:index"
        xmlns:bfvs="bayfac:facetVector:show"
        xmlns:bfvi="bayfac:facetVector:index"
        xmlns:bfvc="bayfac:facetVector:create"
        xmlns:bfvu="bayfac:facetVector:update"
        xmlns:bfss="bayfac:fs:show"
        xmlns:bfsi="bayfac:fs:index"
        xmlns:bis="bayfac:instance:show"
        xmlns:bii="bayfac:instance:index"
        xmlns:bic="bayfac:instance:create"
        xmlns:biu="bayfac:instance:update"
        xmlns="http://research.sun.com/wadl/2006/10">
        <grammars>
                <include href="defaultMessage.xsd"/>
                <include href="facet/show.xsd"/>
                <include href="facet/index.xsd"/>
                <include href="facetItem/show.xsd"/>
                <include href="facetItem/index.xsd"/>
                <include href="facetVector/show.xsd"/>
                <include href="facetVector/index.xsd"/>
                <include href="facetVector/create.xsd"/>
                <include href="facetVector/update.xsd"/>
                <include href="fs/show.xsd"/>
                <include href="fs/index.xsd"/>
                <include href="instance/show.xsd"/>
                <include href="instance/index.xsd"/>
                <include href="facetVector/create.xsd"/>
                <include href="facetVector/update.xsd"/>
        </grammars>
        <resources base="http://sim.tudor.lu/exampleBayFac/index.php/rest/">
                <resource path="fs">
                        <method name="GET" id="fs_index">
                                <request/>
                                <response>
                                        <representation status="200" mediaType="application/xml"
element="bfsi:fsIndex"/>
                                        <fault status="400 401 404 405 409 500 503" me-
diaType="application/xml"
                                                element="bay:message"/>
                                </response>
                        </method>
                        <resource path="{fs_id}">
                                <param name="{fs_id}" style="template" type="xsd:string"/>
                                <method name="GET" id="fs_show">
                                        <request/>
                                        <response>
                                                <representation status="200" me-
diaType="application/xml" element="bfss:fs"/>
                                                <fault status="400 401 404 405 409 500 503" me-
diaType="application/xml"
                                                        element="bay:message"/>
                                        </response>
                                </method>
                                <resource path="instance">
                                        <method name="GET" id="instance_search">
                                                <request>
                                                        <param name="keywords" style="query"
type="xsd:string"/>
                                                        <param name="facet_items" style="query"
type="xsd:string"/>
                                                </request>
                                                <response>
                                                        <representation status="200" me-
diaType="application/xml"
                                                                element="bii:instances"/>
                                                        <fault status="400 401 404 405 409 500
503" mediaType="application/xml"
```

```
                                                element="bay:message"/>
                                        </response>
                                </method>
                                <method name="POST" id="instance_create">
                                        <request>
                                                <representation me-
diaType="application/xml" element="bic:instance"/>
                                        </request>
                                        <response>
                                                <representation status="201" me-
diaType="application/xml"
                                                        element="bay:message"/>
                                                <fault status="400 401 404 405 409 500
503" mediaType="application/xml"
                                                        element="bay:message"/>
                                        </response>
                                </method>
                                <resource path="{instance_id}">
                                        <param name="{instance_id}" style="template"
type="xsd:string"/>
                                        <method name="GET" id="instance_show">
                                                <request/>
                                                <response>
                                                        <representation status="200" me-
diaType="application/xml"
                                                                element="bis:instance"/>
                                                        <fault status="400 401 404 405 409
500 503"
                                                                me-
diaType="application/xml" element="bay:message"/>
                                                </response>
                                        </method>
                                        <method name="PUT" id="instance_update">
                                                <request>
                                                        representation me-
diaType="application/xml" element="biu:instance"/>
                                                </request>
                                                <response>
                                                        <representation status="200" me-
diaType="application/xml"
                                                                element="bay:message"/>
                                                        <fault status="400 401 404 405 409
500 503"
                                                                me-
diaType="application/xml" element="bay:message"/>
                                                </response>
                                        </method>
                                        <method name="DELETE" id="instance_delete">
                                                <request/>
                                                <response>
                                                        <representation status="200" me-
diaType="application/xml"
                                                                element="bay:message"/>
                                                        <fault status="400 401 404 405 409
500 503"
                                                                me-
diaType="application/xml" element="bay:message"/>
                                                </response>
                                        </method>

                                </resource>
                        </resource>
                        <resource path="facet">
                                <method name="GET" id="facet_index">
                                        <request/>
                                        <response>
                                                <representation status="200" me-
diaType="application/xml"
                                                        element="bfi:facets"/>
                                                <fault status="400 401 404 405 409 500
503" mediaType="application/xml"
                                                        element="bay:message"/>
                                        </response>
                                </method>
                                <resource path="{facet_id}">
                                        <param name="{facet_id}" style="template"
type="xsd:string"/>
                                        <method name="GET" id="facet_show">
                                                <request/>
                                                <response>
                                                        <representation status="200" me-
diaType="application/xml"
                                                                element="bfs:facet"/>
```

```
                                                        <fault status="400 401 404 405 409
500 503"
                                                                me-
diaType="application/xml" element="bay:message"/>
                                                </response>
                                        </method>
                                        <resource path="facet_item">
                                                <method name="GET" id="facetItem_index">
                                                        <request/>
                                                        <response>
                                                                <representation
status="200" mediaType="application/xml"
                                                                        ele-
ment="bfii:facetItems"/>
                                                                <fault status="400 401 404
405 409 500 503"
                                                                        me-
diaType="application/xml" element="bay:message"/>
                                                        </response>
                                                </method>
                                                <resource path="{facetItem_id}">
                                                        <param name="{facetItem_id}"
style="template" type="xsd:string"/>
                                                        <method name="GET"
id="facetItem_show">
                                                                <request/>
                                                                <response>
                                                                        <representation
status="200" mediaType="application/xml"
                                                                                ele-
ment="bfis:facetItem"/>
                                                                        <fault status="400
401 404 405 409 500 503"
                                                                                me-
diaType="application/xml" element="bay:message"/>
                                                                </response>
                                                        </method>
                                                </resource>
                                        </resource>
                                </resource>
                                <resource path="facet_vector">
                                        <method name="GET" id="facetVector_index">
                                                <request/>
                                                <response>
                                                        <representation status="200" me-
diaType="application/xml"
                                                                element="bfvi:facetVectors"/>
                                                        <fault status="400 401 404 405 409 500
503" mediaType="application/xml"
                                                                element="bay:message"/>
                                                </response>
                                        </method>
                                        <method name="POST" id="facetVector_create">
                                                <request>
                                                        <representation me-
diaType="application/xml" element="bfvc:instance"/>
                                                </request>
                                                <response>
                                                        <representation status="201" me-
diaType="application/xml"
                                                                element="bay:message"/>
                                                        <fault status="400 401 404 405 409 500
503" mediaType="application/xml"
                                                                element="bay:message"/>
                                                </response>
                                        </method>
                                        <resource path="{facetVector_id}">
                                                <param name="{facetVector_id}" style="template"
type="xsd:string"/>
                                                <method name="GET" id="facetVector_show">
                                                        <request/>
                                                        <response>
                                                                <representation status="200" me-
diaType="application/xml"
                                                                        ele-
ment="bfvs:facetVector"/>
                                                                <fault status="400 401 404 405 409
500 503"
                                                                        me-
diaType="application/xml" element="bay:message"/>
                                                        </response>
                                                </method>
```

```
                                                <method name="PUT" id="facetVector_update">
                                                        <request>
                                                                <representation me-
diaType="application/xml" element="bfvu:instance"/>
                                                        </request>
                                                        <response>
                                                                <representation status="200" me-
diaType="application/xml"
                                                                        element="bay:message"/>
                                                                <fault status="400 401 404 405 409
500 503"
                                                                        me-
diaType="application/xml" element="bay:message"/>
                                                        </response>
                                                </method>
                                                <method name="DELETE" id="facetVector_delete">
                                                        <request/>
                                                        <response>
                                                                <representation status="200" me-
diaType="application/xml"
                                                                        element="bay:message"/>
                                                                <fault status="400 401 404 405 409
500 503"
                                                                        me-
diaType="application/xml" element="bay:message"/>
                                                        </response>
                                                </method>

                                        </resource>
                                </resource>
                        </resource>
                </resource>
        </resources>
</application>
```

A.2. defaultMessage.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="bayfac"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="message">
    <xs:complexType>
      <xs:all>
        <xs:element name="code" type="xs:string"/>
        <xs:element name="content" type="xs:string"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.3. facet/index.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:facet:index"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="facets">
    <xs:complexType>
      <xs:all>
        <xs:element name="facet" type="xs:anyURI" minOccurs="0" maxOc-
curs="unbounded"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.4. facet/show.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:facet:show"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="facet">
    <xs:complexType>
      <xs:all>
        <xs:element name="label" type="xs:string"/>
        <xs:element name="comment" type="xs:string" minOccurs="0"/>
        <xs:element name="exploit" type="xs:string"/>
        <xs:element name="facetItems" type="xs:anyURI" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.5. facetItem/index.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:facetItem:index"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="facetItems">
    <xs:complexType>
      <xs:all>
        <xs:element name="facetItem" type="xs:anyURI" minOccurs="0" maxOc-
curs="unbounded"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.6. facetItem/show.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:facetItem:show"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="facetItem">
    <xs:complexType>
      <xs:all>
        <xs:element name="label" type="xs:string"/>
        <xs:element name="comment" type="xs:string" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.7. facetVector/create.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:facetVector:create"
       elementFormDefault="qualified" attributeFormDefault="unqualified">
       <xs:element name="facetVector">
              <xs:complexType>
                     <xs:all>
                            <xs:element name="label" type="xs:string"/>
                            <xs:element name="comment" type="xs:string" minOc-
curs="0"/>
                            <xs:element name="instance" type="xs:anyURI"/>
                            <xs:element name="vector">
                                   <xs:complexType>
                                          <xs:all>
```

```
                                                <xs:element name="facetVectorItem"
minOccurs="0" maxOccurs="unbounded">
                                                      <xs:complexType>
                                                            <xs:all>
                                                                  <xs:element
name="facet" type="xs:anyURI"/>
                                                                  <xs:element
name="facetItem" type="xs:string"/>
                                                            </xs:all>
                                                      </xs:complexType>
                                                </xs:element>
                                          </xs:all>
                                    </xs:complexType>
                              </xs:element>
                        </xs:all>
                  </xs:complexType>
            </xs:element>
</xs:schema>
```

A.8. facetVector/index.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:facetVector:index"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="facetVectors">
    <xs:complexType>
      <xs:all>
        <xs:element name="facetVector" type="xs:anyURI" minOccurs="0" maxOc-
curs="unbounded"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.9. facetVector/show.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:facetVector:show"
      elementFormDefault="qualified" attributeFormDefault="unqualified">
      <xs:element name="facetVector">
            <xs:complexType>
                  <xs:all>
                        <xs:element name="label" type="xs:string"/>
                        <xs:element name="comment" type="xs:string" minOc-
curs="0"/>
                        <xs:element name="instance" type="xs:anyURI"/>
                        <xs:element name="vector">
                              <xs:complexType>
                                    <xs:all>
                                          <xs:element name="facetVectorItem"
minOccurs="0" maxOccurs="unbounded">
                                                <xs:complexType>
                                                      <xs:all>
                                                            <xs:element
name="facet" type="xs:anyURI"/>
                                                            <xs:element
name="facetItem" type="xs:string"/>
                                                      </xs:all>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:all>
                              </xs:complexType>
                        </xs:element>
                  </xs:all>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

A.10. facetVector/update.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:facetVector:update"
       elementFormDefault="qualified" attributeFormDefault="unqualified">
       <xs:element name="facetVector">
              <xs:complexType>
                     <xs:all>
                            <xs:element name="label" type="xs:string"/>
                            <xs:element name="comment" type="xs:string" minOc-
curs="0"/>
                            <xs:element name="instance" type="xs:anyURI"/>
                            <xs:element name="vector">
                                   <xs:complexType>
                                          <xs:all>
                                                 <xs:element name="facetVectorItem"
minOccurs="0" maxOccurs="unbounded">
                                                        <xs:complexType>
                                                               <xs:all>
                                                                      <xs:element
name="facet" type="xs:anyURI"/>
                                                                      <xs:element
name="facetItem" type="xs:string"/>
                                                               </xs:all>
                                                        </xs:complexType>
                                                 </xs:element>
                                          </xs:all>
                                   </xs:complexType>
                            </xs:element>
                     </xs:all>
              </xs:complexType>
       </xs:element>
</xs:schema>
```

A.11. fs/index.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:fs:index"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="fsIndex">
    <xs:complexType>
      <xs:all>
        <xs:element name="fs" type="xs:anyURI" minOccurs="0" maxOc-
curs="unbounded"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.12. fs/show.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:fs:show"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="fs">
    <xs:complexType>
      <xs:all>
        <xs:element name="label" type="xs:string"/>
        <xs:element name="instances" type="xs:anyURI"/>
        <xs:element name="facets" type="xs:anyURI"/>
        <xs:element name="facetVectors" type="xs:anyURI"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.13. instance/create.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:instance:create"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="instance">
    <xs:complexType>
      <xs:all>
        <xs:element name="label" type="xs:string"/>
        <xs:element name="comment" type="xs:string" minOccurs="0"/>
        <xs:element name="hasURI" type="xs:anyURI" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.14. instance/index.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:instance:index"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="instances">
    <xs:complexType>
      <xs:all>
        <xs:element name="instance" type="xs:anyURI" minOccurs="0" maxOc-
curs="unbounded"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.15. instance/show.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:instance:show"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="instance">
    <xs:complexType>
      <xs:all>
        <xs:element name="label" type="xs:string"/>
        <xs:element name="comment" type="xs:string" minOccurs="0"/>
        <xs:element name="postDate" type="xs:date" minOccurs="0"/>
        <xs:element name="hasPath" type="xs:string" minOccurs="0"/>
        <xs:element name="hasURI" type="xs:anyURI" minOccurs="0"/>
        <xs:element name="classificationState" type="xs:string"/>
        <xs:element name="facetVector" type="xs:anyURI" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.16. instance/update.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetName-
space="bayfac:instance:show"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="instance">
    <xs:complexType>
      <xs:all>
        <xs:element name="label" type="xs:string"/>
        <xs:element name="comment" type="xs:string" minOccurs="0"/>
        <xs:element name="hasURI" type="xs:anyURI" minOccurs="0"/>
        <xs:element name="classificationState" type="xs:string"/>
        <xs:element name="facetVector" type="xs:anyURI" minOccurs="0"/>
      </xs:all>
    </xs:complexType>
```

```
  </xs:element>
</xs:schema>
```