Projet no. FP6-028038

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

Integrated Project

Technology-enhanced learning

D.KNO.04
Basic CoP-oriented Knowledge Management Tool offering basic CoP-adapted KM Services

Due date of deliverable: 31 July 2007
Actual submission date: 15 November 2007

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

Project co-funded by the European Commission within the Sixth Framework Program		
Dissemination Level		
P	Public	**PU**

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

MODIFICATION CONTROL			
Version	**Date**	**Status**	**Modifications made by**
0.1	08/05/2007	Outline	Adil El Ghali
0.2	09/05/2007	Outline	Rose Dieng-Kuntz
0.3	17/06/2007	ECCO	Alain Giboin
0.4	30/06/2007	Generis, BayFac	Benjamin Gateau, Yannick Naudet, Alain Vagner, Marie-Laure Watrinet
0.5	15/07/2007	Sweetwiki	Amira Tifous
1.0	29/10/2007	pre-Final version for reviewers	Adil El Ghali
1.0	29/10/2007	Send to Reviewers	Adil El Ghali
1.2	4/11/2007	Feedback of Reviewer	Olivier Corby
1.3	4/11/2007	Feedback of Reviewer	Nikos Karacapilidis
1.4	4/11/2007	Feedback of Reviewer	Rose Dieng-Kuntz
1.5	12/11/2007	Corrections	Adil El Ghali
2.0	13/11/2007	Sent to SC	Rose Dieng-Kuntz

Deliverable manager Adil El Ghali (INRIA)

Contributors

Adil El Ghali	(INRIA)
Benjamin Gateau	(CRP-HT)
Alain Giboin	(INRIA)
Yannick Naudet	(CRP-HT)
Amira Tifous	(INRIA)
Alain Vagner	(CRP-HT)
Marie-Laure Watrinet	(CRP-HT)

Evaluators

Olivier Corby	(INRIA)
Nikos Karacapilidis	(CTI)

Acknowledgments
We want to thank M. Buffa, P. Durville and F. Gandon for their assistance.

Executive Summary
This deliverable presents the Knowledge Management (KM) services developped in the context of WP3.

On the one hand, it presents the technical descriptions of basic KM services based on Corese/SeWeSe and Generis: these services (previously specified in D.KNO.03) provide low level operations of KM, they will not be directly used by the CoPs, but by the designers and the developers of the Palette project.

On the other hand, it presents functional specifications and a first report of the use of complex KM services: ECCO, SweetWiki and BayFac. These services are or will be used by the Palette CoPs. The description complex services will be detailed in the next deliverable D.KNO.06.

Contents

Chapter 1

Introduction

The main goal of WP3 is to offer knowledge management (KM) services to manage CoPs knowledge resources, in order to improve access, sharing, and reuse of knowledge; and the (individual or collective) creation of new knowledge.

The Tasks 3.3 and 3.5 aim to provide CoPs with KM services. In D.KNO.03 [13] we specified the building block services. The implemented basic KM services are :

- Knowledge creation: to manage ontologies;
- Knowledge annotation: to manage annotations;
- Knowledge retrieval: to achieve semantic search on resources.

These building blocks were implemented as web-services based on the libraries presented in D.KNO.03 [13]. The services based on Corese/SeWeSe are provided as SOAP web-services. The services based on Generis are provided as REST and/or SOAP web-services. In addition, this deliverable includes the specifications of some complex services (ECCO, Sweetwiki and BayFac), and gives some preliminary results on the use of these complex services by the Palette CoPs. The services presented in chapters 2 and 3, are the elementary KM services (specified in the deliverable D.KNO.03) implementing low level operations of KM and not directly used by the CoPs. On the contrary, the complex services presented in chapters 4 and 5 represent high level KM services, that are or will be used by CoPs. Some of these complex services are in an "early" stage of development, the next deliverable (D.KNO.06) will elaborate all of the complex services..

After this introduction, this deliverable is composed of four chapters:

- The chapter 2 presents the **Corese/SeWeSe web-services**. In this chapter we detail the following services: Ontology and Annotation management service, Retrieval service, and Administration service. For each of these services we present the Input/Output and possible exceptions of their operations. We also provide some examples in different programming languages: it shows how these services can be used by the other technical partners of Palette, in order to prepare the implementation of the interoperability of Palette services specified in WP5.
- The chapter 3 presents the **Generis web-services**. In this chapter we detail the following services: Ontology management service, Annotations service, and Retrieval service (semantic and full-text search on the resources). Examples of use of these services are given in PHP.
- The chapter 4 presents **ECCO**, a tool for collaborative and contextual construction of ontologies. This service was used in the context of the task 3.2 to build the O'CoP ontology. We focus here on how ECCO relies on the basic KM services, giving an example on the possible combinations of basic KM services in order to build a complex one.

- The chapter 5 presents two of the complex services dedicated to the CoPs:
  - **Sweetwiki** is a semantic wiki, providing CoPs with a platform for collaborative work, enhanced with semantic web facilities. We present some first results on the use of Sweetwiki by Palette CoPs. We also draw a picture of the future development of Sweetwiki, the main lines of these future developments had been defined together with the CoPs in the context of the Palette PDM.
  - **BayFac** is a classification service of textual resources (including documents, e-mails, ...) produced in a CoP, w.r.t. a vector of concepts revelant for the CoP. It is based on faceted classification and Bayesian approach to classification. BayFac provides a web service allowing the classification of resources, and a web portal to be directly used by the CoP. It is currently tested with the Form@Hetice CoP.

  In the conclusion, we will evoke the future developments of the complex services and some interoperability issues with other Palette tools and services.

Chapter 2

Basic KM services using Corese/SeWeSe

2.1 Introduction

SeWeSe is a semantic web application development platform offering common functionalities of knowledge management. It relies on the semantic engine Corese. The goal of SeWeSe is to provide reusable, configurable and extensible primitives and components in order to reduce the amount of time spent to develop new semantic web applications. Developers can then focus on the applications specificities. Corese/SeWeSe can be used in three different ways depending on the nature of the application that use it :

Java Library To be used in a Java application. An API describing the functionalities of SeWeSe is provided at: `http://www-sop.inria.fr/acacia/soft/sewese`;

JSP taglib To be used in a JSP webapp, the *semtags* allow JSP developers to use (and work with) RDF/S and OWL ontologies and annotations in their JSP pages. The API describing this taglib is provided at:
`http://www-sop.inria.fr/edelweiss/software/sewese/v1_5/tld-api`;

Web services Some of the functionalities are provided as SOAP web services: *Ontologies* and *Annotations* management and semantic *Retrieval* on these resources.

These services can be used by any application that needs a knowledge management functionality. The corresponding service can be called w.r.t. its description given in its WSDL. The general architecture of these services is shown in the Fig. 2.1 :

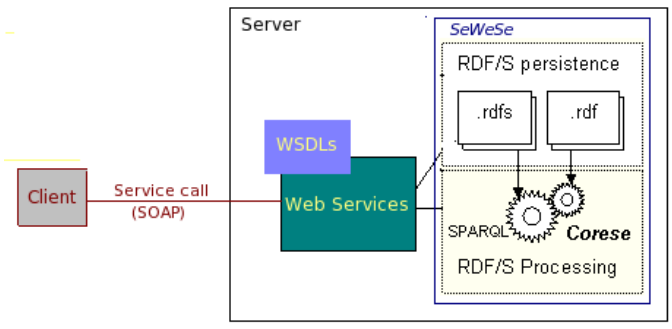


Figure 2.1: A simple view of the architecture of web-services

In this chapter, we will describe these services giving their inputs and outputs, the operations they can achieve, and the exceptions they can throw. We will also provide some examples illustrating how they can be called.

2.2 Administration service

In order to work with the semantic services, users[1] have to setup an engine on the server. An engine, defined by its engineID, is an instance of Corese/SeWeSe dedicated to a user. The following operations are available to create and modify an instance of the engine.

2.2.1 Engine creation

Two operations are available to create and initialize an engine, the default one, init, creates an engine with default parameters, the second, customInit, creates an engine and initializes it with user's provided parameters to set the maximum number of projections done by Corese, the maximum number of results that should be returned by the engine to queries, and a Boolean parameter that indicates if a join operation should be applied to the results.

init Creates and initializes a new instance of the engine.

customInit Creates a new instance of the engine and initializes it with custom parameters.

- Input ontosURI, annotsURI, rulesURI are the URIs of the ontologies, the annotations and the rules on which the engine will work.
Additional input for customInit are: projectionMax which is the maximum projections number; resultMax is the maximum results number; resultJoin is a Boolean that tells the engine (when set to `true`) to join the results.

- Output engineID is the ID of the engine provided by the server.

- Exceptions the InvalidDataException exception is thrown if one of ontosURI, annotsURI or rulesURI is not a valid URI or if they denote a malformed content.

The initialization operation can be called (in Java) using the following code :

```
public static void main(String[] args) {
  // We create a local service
  Service serviceModel = new ObjectServiceFactory().create(AdminService.class);
  // then we need to create a proxy for the distant service
  AdminService service = (AdminService) new XFireProxyFactory().create(
   serviceModel,"http://argentera.inria.fr/semservices/Admin?wsdl");

  // We invoke the service with the O'CoP ontology and no annotations or rules.
  long engineID = service.init("http://ns.inria.fr/palette/2007/07/ocop#", "", "");
  // And display the obtained engineID
  System.out.println("The engine ID is " + engineID);
 }
```

2.2.2 Engine modification

After its creation, an engine can be modified using one of the following three operations, (i) adding new data to the engine load, (ii) remove data from the engine unload, (iii) reset the engine to get it with its original creation data reset.

[1] By users here we mean the users of the web services, not the end users.

load Loads an additional ontology or annotation or rule after engine initialization.

- Input newData is the URI of the new data to load, the newData can be ontologies, annotations or rules; engineID is the ID of the engine to modify.

- Exceptions two exceptions can be thrown by this operation : (i) EngineNotFoundException if the engineID does not denote an engine. (ii) InvalidDataException if newData is not a valid URI or if it denotes a malformed content.

reset Reloads an engine with its initialization parameters and data.

- Input engineID is the ID of the engine to reload.

- Exceptions the EngineNotFoundException exception can be launched if engineID does not denote an engine.

unload Unloads an ontology, an annotation or a rule after initialization.

- Input dataURI is the URI of ontologies, annotations or rules to remove from the engine; engineID is the ID of the engine to modify.

- Exceptions two exceptions can be thrown by this operation : (i) EngineNotFoundException if engineID does not denote an engine. (ii) InvalidDataException if dataURI is not a valid URI or if it denotes a malformed content.

The operations can also be invoked using a dynamic client, a small example of such client is shown below:

```
public void testAdmin() throws MalformedURLException, Exception {
  // We create a client from the WSDL
  Client client = new
        Client(new URL("http://argentera.inria.fr/semservices/Admin?wsdl"));

  // We invoke operations
  // 1. we create an empty engine
  Object[] results = client.invoke("init",
                          new Object[] {"", "", ""});
  long engineID = (long) results[0];
  // 2. we load the sioc ontology into the engine
  results = client.invoke("load",
                          new Object[] {"http://rdfs.org/sioc/ns#", engineID});
  // 3. we get and  display the status of the engine (see next section)
  results = client.invoke("isValidEngineID",
                          new Object[] {engineID});
  if ((boolean) results[0]) System.out.println("The engine is OK");
  else System.out.println("The engine is KO");
}
```

2.2.3 Engine status

The users may also need to know if the ID they own refers to a valid and running engine. This can be achieved using the operation isValidEngineID.

isValidEngineID Tests if the given ID is refering to an engine. Note that an engine ID expires after a given amount of time. So a valid ID can become invalid after a timeout.

- Input engineID is the ID of the engine to test.

- Output a Boolean indicating (when `true`) that the given ID is associated with an engine.

2.3 Ontology management service

The ontology management service provides operations for the creation, modification and removal of ontologies. These operations can be of three types: (i) ontology level operations: to create or remove an ontology; (ii) class or property level operations: to add or remove a class or a property; (iii) class or property details level operations: to add, remove or modify the characteristics of a class or a property.

2.3.1 Ontology level operations

The first type offered by the ontology management service is related to the creation, modification of ontologies. Users can use the operations detailed below:

create Creates a new ontology file at the specified URI. This operation allows to create an ontology file, which belongs to an engine engineID, and to specify information related to this ontology.

- Input filename is the URI that represents the filename of the ontology to create, base is the URI base (xml:base) of the ontology, graph is the URI graph (cos:graph) of the ontology, label is the owl:label, comment is the owl:comment, version is the owl:versionInfo, override is a Boolean that indicates, in case of the existence of a file with the URI filename, if it can be overriden, forceUpdate is a Boolean that indicates if the engine should be reloaded after the creation of the ontology; engineID is the ID of the engine in which the ontology is created.

- Output a Boolean is returned, its value is `true` if a file has been created, `false` otherwise.

- Exceptions the EngineNotFoundException exception is thrown if forceUpdate is `true` and `engineID` does not denote an engine. The InvalidDataException exception is thrown if the filename URI is an invalid URI.

remove Removes an ontology file. This operation must be used carefully, since another ontology or annotations can have a reference to a concept/property of this removed ontology: in this case, the concept/property will exist anyway without its characteristics (labels/comments and so on) defined in the removed ontology.

- Input filename is the URI that represents the filename of the ontology to remove; forceUpdate is a Boolean indicating that the engine should be reloaded when set to `true`; engineID is the ID of the engine from which we want to remove the ontology ;

- Output a Boolean is returned, its value is `true` if a file has been removed, `false` otherwise.

- Exceptions the EngineNotFoundException exception is launched if forceUpdate is `true` and `engineID` does not denote an engine. The InvalidDataException exception is launched if the filename URI is an invalid URI.

modifyComment Modifies the ontology comment (`owl:comment`).
modifyLabel Modifies the ontology label (`owl:label`).
modifyVersion Modifies the ontology version (`owl:versionInfo`).

- Input filename is the URI of the ontology;
comment is the string representing the new value of `owl:comment`;
label is the string representing the new value of `owl:label`;
version is the string representing the new value of `owl:versionInfo`.

- Exceptions the InvalidDataException exception is thrown if the filename URI is an invalid URI or denotes a malformed content.

To use the service in C#, we need first to build a proxy class and compile it by executing the following code :

```
C:>wsdl.exe http://argentera.inria.fr/semservices/Ontology?wsdl
C:\>csc /t:library c:\Fortune.cs
```

The resulting C# file `Ontology.cs` contains a class called Ontology with the operations of the ontology management service, *e.g.* `void modifyLabel(String filename, String label)`. We can invoke these operations as follows :

```
using System;
using System.IO;

class WSConsumer
{
  public static void Main()
  {
    Ontology o = new Ontology();
    // We create an ontology
    if (o.create("ontocop1", "http://ns.inria.fr/palette/2007/10/ontoCoP1#",
                 "", "", "", "1.0", true, true, engine))
      Console.WriteLine("The ontology is created");
    // and modify its label
    o. modifyLabel(, "Ontology for CoP1") );
  }
};
```

2.3.2 Class/Property level operations

These operations allow users to add or remove classes and properties.

addConcept Adds a class to an ontology,
addProperty Adds a property to an ontology

- Input conceptID is the ID of the class to add;
propertyID is the ID of the property to add;
labels is an array of labels and their associated languages (for each n, labels$[n][0]$ is a label and labels$[n][1]$ is its language);
comments is an array of comments and their associated languages (for each m, comments$[m][0]$ is a comment and comments$[m][1]$ is its language);
parents is the set of URIs of the parents of the class or the property to add; definedBy is the list of the defined by URIs; and ontology the URI of the ontology to which the class or the property will be added.

- Exceptions the InvalidDataException exception is triggered if the ontology URI is an invalid URI or denotes a malformed content, and the ConceptOrPropertyAlreadyAvailableException exception is triggered if the class URI or the property URI already exists.

removeConcept Removes a concept from an ontology, this operation also permits to replace all the references to the removed class in the knowledge base by another class.

removeProperty Removes a property from an ontology, this operation also permits to replace all the references to the removed property in a knowledge base by another property.

- Input conceptID is the ID of the class to remove;
propertyID is the ID of the property to remove; ontology is the URI of the ontology from which the class or the property is to be removed; updateData is a Boolean parameter that indicates (when it is set to `true`) that the whole base (ontologies and annotations) should be updated; substituteID is the ID of the class ot the property to use to substitute the removed one; and engineID is the ID of the engine containing the ontologies and annotations to be updated. The two last parameters are optional and used only if updateData is set to `true`.

- Exceptions the InvalidDataException exception is thrown if the ontology URI is an invalid URI or denotes a malformed content, the ConceptOrPropertyNotFoundException exception is thrown if conceptID or propertyID is not defined in the given ontology, and the EngineNotFoundException exception is thrown if engineID does not denote an engine (only if updateData is set to `true`).

2.3.3 Class/Property characteristics level operations

The last type of operations of ontology management is related to the edition of the characteristics of classes and properties. These operations can be used to edit the comments, labels, parents and defined-by clause for both classes and properties; and to edit the domain and range of properties.

addComment Adds a comment to a class or a property.

addLabel Adds a label to a class or a property.

- Input comment is the `String` corresponding to the new comment to add;
label is the new label to add;
lang is the language of the comment or the label to add; notionID is the URI of the class or property to modifiy; and ontology is the URI of the ontology which the class or property belongs to;

- Exceptions the InvalidDataException exception is thrown if the ontology URI is an invalid URI or denotes a malformed content; the ConceptOrPropertyNotFoundException exception is thrown if notionID is not defined in the given ontology.

removeComment Removes a comment of a class or property.

removeLabel Removes a label of a class or a property.

- Input comment is the comment to remove;
label is the label to remove;
notionID is the URI of the class or property to modifiy; and ontology is the URI of the ontology which the class or property belongs to;

- Exceptions the InvalidDataException exception is thrown if ontology URI is an invalid URI or denotes a malformed content; the ConceptOrPropertyNotFoundException exception is thrown if notionID is not defined in the given ontology.

addDefinedBy Adds a definedBy clause to a class or a property.

- Input definedBy is the class or property URI of the defined by clause to add; notionID is the URI of the class or property to modifiy; and ontology is the URI of the ontology to which the class or property belongs;

- Exceptions the InvalidDataException exception is launched if ontology URI is an invalid URI or denotes a malformed content; the ConceptOrPropertyNotFoundException exception is launched if notionID is not defined in the given ontology.

removeDefinedBy Removes a definedBy clause of a class or a property.

- Input definedBy is the class or property URI of the defined by clause to remove; notionID is the URI of the class or property to modifiy; and ontology is the URI of the ontology to which the class or property belongs;

- Exceptions the InvalidDataException exception is thrown if ontology URI is an invalid URI or denotes a malformed content; the ConceptOrPropertyNotFoundException exception is thrown if notionID is not defined in the given ontology.

addParent Adds a parent to a concept/property.

- Input parent is the URI of the class or the property to set as parent for the notionID, the class or property to modify; the ontology is the URI of the ontology to which the class or property belongs;

- Exceptions the InvalidDataException exception is triggered if ontology URI is an invalid URI or denotes a malformed content; the ConceptOrPropertyNotFoundException exception is thrown if notionID is not defined in the given ontology.

removeParent Removes a parent of a concept/property.

- Input parent is the URI of the class or the property to remove; notionID is the class or the property to modify; and ontology is the URI of the ontology to which the class or property belongs;

- Exceptions the InvalidDataException exception is thrown if ontology URI is an invalid URI or denotes a malformed content; the ConceptOrPropertyNotFoundException exception is thrown if notionID is not defined in the given ontology.

addRange Adds a range to a property.
addDomain Adds a domain to a property.

- Input range is the URI of the class to be set as range for notionID the property to modify; domain is the URI of the class to be set as domain for the property notionID; and ontology is the URI of the ontology to which the property belongs;

- Exceptions the InvalidDataException exception is triggered if ontology URI is an invalid URI or denotes a malformed content; the ConceptOrPropertyNotFoundException exception is triggered if the property notionID is not defined in the given ontology.

removeRange Removes a range of a property.

removeDomain Removes a domain of a property.

- Input range is the URI of the class to be removed from the range of the property notionID; domain is the URI of the class to be removed from the domain of the propertynotionID; ontology is the URI of the ontology to which the property belongs;

- Exceptions the InvalidDataException exception is thrown if the ontology URI is an invalid URI or denotes a malformed content; the ConceptOrPropertyNotFoundException exception is thrown if the notionID is not defined in the given ontology.

2.4 Annotation management

The annotation management service allows to create, modify and remove semantic annotations. These annotations are represented in XML/RDF. Let us see the details of these three operations.

create Creates a new annotation file at the specified URL.

- Input filename is the URI that represents the filename of the new annotation to create; content represents the annotation triples in an XML/RDF syntax; the override parameter is a Boolean that indicates, in case of the existence of a file with the same URI, if it can be overriden (when `true`) or not; forceUpdate is a Boolean parameter that indicates (when `true`) if the engine in which the annotation is created should be reloaded after the creation; engineID is the ID of the engine in which the annotation will be created;

- Output the operation returns a Boolean, set to `true` when the annotation was successfully created, and `false` otherwise.

- Exceptions the InvalidDataException exception is thrown if filename URI is an invalid URI or denotes a malformed content; and the EngineNotFoundException exception is thrown when engineID does not denote an engine.

remove Removes an annotation. It should be noticed that if another annotation has a reference to an instance of this(these) removed instance(s), this(these) instance(s) will exist anyway without its(their) characteristics set in the removed annotation!

- Input filename is the URI that represents the filename of the annotation to remove; forceUpdate is a Boolean parameter that indicates (when `true`) if the engine in which the annotation is created should be reloaded after the removal of the annotation; engineID is the ID of the engine from which the annotation will be removed.

- Output the operation returns a Boolean, set to `true` when the annotation was successfully removed, and `false` otherwise.

- Exceptions the InvalidDataException exception is thrown if filename URI is an invalid URI or denotes a malformed content; and the EngineNotFoundException exception is thrown when engineID does not denote an engine.

modify Modifies an annotation file.

- Input filename is the URI that represents the filename of the annotation to modify; kind represents the modification to perform, it should be set to one of `'add'`, `'del'`, or `'replace'`; the xpath is the XPath to use to get the impacted node(s): the parent node(s) in case of an addition, or the node(s) to delete or replace by new one(s); usedNamespacesInXPath is a table representing the used namespaces in the XPath expression, e.g. [["`rdf`","`http://www.w3.org/1999/02/22-rdf-syntax-ns#`"], ["`rdfs`", "`http://www.w3.org/2000/01/rdf-schema#`"]]; content represents the new triples to add or replace with (in an XML/RDF syntax); forceUpdate is a Boolean parameter that indicates (when `true`) if the engine in which the annotation is created should be reloaded after the removal of the annotation; engineID is the ID of the engine from which the annotation will be removed.

- Exceptions the InvalidDataException exception is thrown if filename URI is an invalid URI or denotes a malformed content; and the EngineNotFoundException exception is thrown when engineID does not denote an engine.

These operations can be used as follows (php):

```
<?PHP
$engineID = <...>
$annotation1 = <...>
$forceupdate = true;
$client = new SoapClient(null,
   array(
    'trace' => 1,
    'soap_version'   => SOAP_1_1,
    'uri' => 'http://argentera.inria.fr/semservices-1.6/Annotation?wsdl',
    ));

// call the remove operation
$client->__soapCall('remove', array($annotation1, $forceupdate, $engineID));
?>
```

2.5 Retrieval and export

The retrieval service is based on the SPARQL query language for RDF [24], the basic operation provided by this service is the query operation. This operation executes a SPARQL query in a previously created engine (see section 2.2). The service also provides some operations to test the validity of a SPARQL query: validate and isValid. In addition, a set of basic requests *e.g.* getParents, isConcept, ... is provided.

Let us see the details of these operations.

2.5.1 SPARQL search

query Sends a SPARQL query to a given engine and gets the response.

- Input sparqlQuery is a `String` containing a SPARQL query[2] ; it is executed on the engine engineID.

- Output the engine response is a `String` in SPARQL format.

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

- Exceptions the EngineNotFoundException exception is thrown when engineID does not denote an engine; the MalformedQuery is thrown if the query does not correspond to the SPARQL format; and QueryRequestRefused is thrown if the engine cannot respond to the query.

validate Validates a SPARQL query.

isValid Tests the validity of query.

- Input sparqlQuery is a `String` containing a SPARQL query; it is sent to the engine engineID to be validated.

- Output
In the case of validate: if the query sparqlQuery is valid then `null` is returned, otherwise a `String` containing the errors in the query is returned.
In the case of isValid: a Boolean is returned, its value is `true` if the query is valid `false` otherwise.

- Exceptions the EngineNotFoundException exception is throws when engineID does not denote an engine.

The query operation can be used to submit any SPARQL query to an engine. In the following example, we create an engine containing the ontology and the folksonomy of the instance of Sweetwiki in `http://argentera.inria.fr/sweetwiki` and we submit a query to get some information about the page `MainHome`.

```
public void testQuery() throws MalformedURLException, Exception {
  // We create a client from the WSDL
  Client clientAdmin = new
        Client(new URL("http://argentera.inria.fr/semservices/Admin?wsdl"));

  // We invoke operations
  // 1. we create an the engine
  Object[] results = client.invoke("init",
         new Object[] {"http://argentera.inria.fr/sweetwiki/files/ontologie/folksonomy
                       "http://argentera.inria.fr/files/", ""});
  long engineID = (long) results[0];
  results = client.invoke("load",
          new Object[] {"http://ns.inria.fr/sweetwiki/2007/08/wiki#",
                        engineID});
  // we submit the query to the engine

  try {
    results = client.invoke("query",
                      new Object[] {"prefix wiki: http://sweetwiki.inria.fr/ontology#
                      select ?author ?date ?seealso ?tag
                      where {
                      ?page wiki:name "MainHome" .
                      ?page wiki: author ?author .
                      ?page wiki: modification ?date .
                      ?page wiki: seeAlso ?seealso .
                      ?page wiki: hasForKeyWord ?tag }", engineID});
        System.out.println("(String) results[0]");
  } catch (Exception e) {
        System.out.println("The query has failed");
  }
}
```

The engine gives the following answer :

```
<?xml version='1.0' encoding='UTF-8'?>
<cos:result xmlns:cos='http://www.inria.fr/acacia/corese#'>
<cos:tquery>
<![CDATA[ prefix wiki: http://sweetwiki.inria.fr/ontology#
                select  ?author ?date ?seealso ?tag
                where {         ?page wiki:name "MainHome" .
                                ?page wiki: author ?author .
                                ?page  wiki: modification ?date .
                                ?page wiki: seeAlso ?seealso .
                                ?page wiki: hasForKeyWord ?tag } ]]>
</cos:tquery>
<cos:info><![CDATA[ Stop query after 100 projections 0.01 s for 100 projections ]]></cos:info>
<sparql xmlns='http://www.w3.org/2005/sparql-results#'
xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#' >
<head> <variable name='author'/> <variable name='date'/> <variable name='seealso'/> </head>
<results ordered='false' distinct='false' >
<result>
<binding name='author'>
<uri>http://sweetwiki.inria.fr/user#AdilElGhali</uri>
</binding>
<binding name='date'>
<literal datatype='http://www.w3.org/2001/XMLSchema#dateTime'>2007-06-26</literal>
</binding>
<binding name='seealso'>
<literal datatype='http://www.w3.org/2001/XMLSchema#string'>Documentation.DocumentationHome</literal>
</binding>
<binding name='tag'>
<uri>http://sweetwiki.inria.fr/folksonomy/proto#sweetwiki</uri>
</binding>
</result>

</results>
```

2.5.2 Implemented basic queries

getComments Gets the comments of a class or a property.

getLabels Gets the labels of a class or a property.

- Input uri is the URI of a class or a property; lang is the language of the comments or the labels to retrieve, and engineID is the ID of the engine to which the class or the property belongs.

- Output a table of `Strings` corresponding to the comments or the labels of the class or property uri with the language lang.

- Exceptions the EngineNotFoundException exception is thrown when engineID does not denote an engine; and the ConceptOrPropertyNotFoundException is thrown if the uri does not denote a class or a property.

getInstanceNb Gets the instances number of a class or a property.

- Input uri is the URI of a class or a property; all is a Boolean parameter that indicates, if it is set to `true`, that all sub-notions instances should be counted; and engineID is the ID of the engine to which the class or the property belongs.

- Output the number of instances is returned.

- Exceptions the EngineNotFoundException exception is triggered when engineID does not denote an engine; and the ConceptOrPropertyNotFoundException is triggered if the uri does not denote a class a property.

isConcept Tests if a given URI denotes a class.

isProperty Tests if a given URI denotes a property.

- Input uri is the URI to test; and engineID is the ID of the engine to which it belongs.

- Output a `Boolean` is returned, its value is set to `true` if the uri denotes a class or a property, and `false` otherwise.

- Exceptions the EngineNotFoundException exception is thrown when the engineID does not denote an engine.

getParents Returns the direct parents of a class or a property.

getChildren Returns the direct children of a class or a property.

- Input uri is the URI of a class or a property; and engineID is the ID of the engine to which the class or the property belongs.

- Output a table of `Strings` corresponding to the URIs of the direct parents or the direct children of the given class or property.

- Exceptions the EngineNotFoundException exception is thrown when engineID does not denote an engine; and the ConceptOrPropertyNotFoundException is thrown if the uri does not denote a class a property.

getRootConcepts Returns all the root classes.

getRootProperties Returns all the root properties.

- Input engineID is the ID of the engine from which the root classes or the root properties are retrieved.

- Output a table of `Strings` corresponding to the URIs of the root classes in the engine.

- Exceptions the EngineNotFoundException exception is thrown when engineID does not denote an engine.

These operations can be used as follows (Perl):

```
#!/usr/bin/perl -w
use SOAP::Lite;

$engineID = #an engine previously created
$concept = <STDIN>;
chomp($concept);

$ret = SOAP::Lite -> uri('urn:query:semservices')
                  -> proxy('http://argentera.inria.fr/semservices-1.6/Query?wsdl')
                  -> isConcept($concept, $engineID);

if ($ret->result) { print "The URI', $concept, "is a  class \n"; }
else { print "The URI', $concept, "is not a  class \n"; }
```

2.6 Conclusion

In this chapter, we present the basic KM services based on Corese/SeWeSe. These services provide low-level operations of knowledge management. These services are used by the complex KM services ECCO and SweetWiki presented in chapters 4 and 5. They will also be used by developers of Palette CoPs who need KM capabilities for their services. Some of these uses were identified in the scenarios in D.PAR.03 [8].

The complete descriptions of the services in WSDL are given in the appendix A.

Chapter 3

Basic KM services using Generis

3.1 Introduction to Generis

Generis is an ontology management tool enabling collaborative annotation of any kind of resources in a distributed way. A Peer-to-Peer network can be constituted using a set of interconnected modules to reflect the geographically distributed knowledge.

Generis enables the management of an ontology in the form of a web resource (according to RDF and RDFS standards). RDFS being fully implemented, Generis enables to manage (creation, edition removal) any kind of resource on all abstraction levels of resources modeling. According to the model or meta-model, user interfaces are dynamically generated to enable the user to manage lower level resources.

Generis also provides facilities to perform full text queries or structured queries (queries expressed according to the model) on the knowledge base. Generis may be accessed using three different interfaces: a Graphical User Interface (GUI), an Application Programming Interface (API), and a series of Web Services (WS).

Furthermore, Generis provides services for the management of users and their access privileges with respect to resources, as well as the communication with other modules, including the management of module subscribers and subscriptions and associated rights (for more informations see D.KNO.03).

Generis proposes several services in addition of the basic ontology services.

- SPARQL queries
- full text search
- XML/RDF export

3.2 How do Generis REST services work?

Generis REST services are called by the way of an URL, like this way:

- http://localhost/kb/palette/
- http://localhost/kb/palette/?lang=keywords&keywords=ctivity%2Comp
- http://localhost/kb/palette/?lang=fetch&r=%2311721388277034

In the URL called:

- *http://localhost/kb* / enables to call the kb file (kb.php[1]) which will redirect information to the dedicated service
- palette/ represents the database where data is located
- and then parameters identify the services to be called
  - query: contains the query to be executed
  - lang: if lang is set to 'fetch', enable to get an XML description of the 'r' parameter
  - keywords: enables to realize a full text search on keywords

3.3 Ontology Edition

Those services allow the management of a Knowledge Base, and so creation and management of ontologies. They are only available as SOAP services.

3.3.1 Edition of a class

Description of the SOAP service:

This service 'editClass' allows editing a class of the ontology. It takes as entries:

- The session of authentication
- The id of the class in case of modification
- The description language of the class
- The label of the class
- Comments of the class
- The parent of the class

Example of use of the SOAP service (php)

```
<?PHP

$client = new SoapClient(null,
   array(
    'trace' => 1,
    'soap_version'  => SOAP_1_1,
    'uri' => 'http://localhost/generis/generis/GenerisKernel/',
    'location' => 'http://localhost/generis/generis/GenerisKernel/webservices.php'
    ));

$session = $client->__soapCall('authenticate',
     array(array("palette"),array("palette"),
     array(""),array("palette")));
```

[1]The kb.php file is detailed in the appendix B

```
// calls the editClass service
$client->__soapCall('editClass', array($session->pSession,
                                       $idclass, $lg,
                                       $labels, $comments,$subClassOf));
?>
```

3.3.2 Edition of a property

Description of the SOAP service

This service allows editing a property of a class of the ontology. It takes as entry:

- The session of authentication
- The id of the property if applicable
- The description language of the property
- The label
- The comments
- The domain of the property (to which class it applies)
- Its range (which values it can have)
- Its widget (how it will be displayed, for example a textbox)

Example of use of the SOAP service (php)

```
<?PHP

$client = new SoapClient(null,
   array(
    'trace' => 1,
    'soap_version'  => SOAP_1_1,
    'uri' => 'http://localhost/generis/generis/GenerisKernel/',
    'location' => 'http://localhost/generis/generis/GenerisKernel/webservices.php'
    ));

$session = $client->__soapCall('authenticate',
     array(array("palette"),array("palette"),
     array(""),array("palette")));

// calls the editProperty service
$client->__soapCall('editProperty', array($session->pSession,$idproperty,
                                          $lg,$labels,$comments,
                                          $domain,$range,$widget));
?>
```

3.3.3 Edition of an instance

Description of the SOAP service

This service allows editing an instance of the ontology. It takes as entry:

- The session of authentication
- The id of the instance if applicable
- The description language
- The label
- The comment
- The type (the corresponding class)

Example of use of the SOAP service (php)

```
<?PHP

$client = new SoapClient(null,
   array(
    'trace' => 1,
    'soap_version'  => SOAP_1_1,
    'uri' => 'http://localhost/generis/generis/GenerisKernel/',
    'location' => 'http://localhost/generis/generis/GenerisKernel/webservices.php'
    ));

$session = $client->__soapCall('authenticate',
     array(array("palette"),array("palette"),
     array(""),array("palette")));

// calls the editInstance service
$client->__soapCall('editInstance', array($session->pSession,$idInstance,
                                          $lg,$labels,$comments,$type));
?>
```

3.4 Knowledge Annotation

Those services allow annotation (by reference) of any data (document) according to managed ontologies.

3.4.1 Description of the SOAP service

This service allows inserting a comment to an instance. It takes as entry:

- The resource id
- The textual comment

3.4.2 Example of use of the SOAP service (php)

```
<?PHP

$client = new SoapClient(null,
   array(
    'trace' => 1,
    'soap_version'  => SOAP_1_1,
    'uri' => 'http://localhost/generis/generis/GenerisKernel/',
```

```
    'location' => 'http://localhost/generis/generis/GenerisKernel/webservices.php'
    ));

// calls the insertComment service
$client->__soapCall('insertComment', array($resourceId, $newComment));
?>
```

3.5 Knowledge retrieval services

Those services allow searching for knowledge in the KB.

3.5.1 Sparql query

Enables to execute Sparql (query language for RDF) queries, which allows interrogating the ontology.

Description of the REST service

This service takes as entry the Sparql query and gives back a list of found elements.

The REST query can be called this way:

`http://localhost/kb/palette/?query=query\_to\_be\_executed`
`(query\_to\_be\_executed being the Sparql query).`

Example of use of the REST service (php)

```
<?php
$req = 'SELECT ?Resource
   WHERE {
   ?Resource
   <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   <https://bscw.ercim.org/bscw/bscw.cgi/d204606/process-activity.rdfs#Process>
    }';
echo '<a href="http://localhost/kb/palette/?query='.urlencode($req).'" >
      sparql test</a><br />';

?>
```

In this example, we first see the sparql query to be executed :

```
'SELECT ?Resource
 WHERE { ?Resource
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <https://bscw.ercim.org/bscw/bscw.cgi/d204606/process-activity.rdfs#Process>
    }'
```

So there, we want to select each resource (SELECT ?Resource) of type (WHERE { ?Resource `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`) Process (`<https://bscw.ercim.org/bscw/bscw.cgi/d204606/process-activity.rdfs#Process>` }').

And the REST service is called:

```
echo '<a href="http://localhost/kb/palette/?query='.urlencode($req).'" >
      sparql test</a><br />';
```

Description of the SOAP service

The Sparql service is also available as a SOAP service.

The SOAP Sparql service takes the same input and output as the REST Sparql service. The only difference is the way it is called. Indeed, SOAP has to be instantiated in order to execute the service.

Example of use of the SOAP service (php)

```
<?PHP

$client = new SoapClient(null,
   array(
    'trace' => 1,
    'soap_version'  => SOAP_1_1,
    'uri' => 'http://localhost/generis/generis/GenerisKernel/',
    'location' => 'http://localhost/generis/generis/GenerisKernel/webservices.php'
    ));

$session = $client->__soapCall('authenticate',
     array(array("palette"),array("palette"),
     array(""),array("palette")));

$req = 'SELECT ?Resource
      WHERE { ?Resource <http://www.w3.org/1999/02/22-rdf-
      syntax-ns#type>
      <https://bscw.ercim.org/bscw/bscw.cgi/d204606/process-activity.rdfs#Process>
      }';

//exec sparql query, returns  sparql xml result
$results =  $client->__soapCall('sparql', array($session->pSession,$req));

print_r($results);

?>
```

First, SOAP is instantiated, using the parameters soap_version, uri where the web services take place, and the location of the web service file where the Sparql SOAP service is located. Then the 'authenticate' service is called with the parameters login, password to access the database, and the databasename. Finally the query is built and the webservice 'sparql' called. The sparql service takes as input the session given back from the authenticate service, and the sparqlquery.

It gives back the list of results, as a Sparql result:

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
<head>
<variable name="Resource"/>
</head>
<results ordered="false" distinct="false">
<result>
<binding name="Resource">
<uri>
http://palette.tudor.lu/generis/2006/palette#annotation
</uri>
</binding>
</result>
<result>
```

```
<binding name="Resource">
<uri>
http://palette.tudor.lu/generis/2006/palette#11721388277034
</uri>
</binding>
</result>
<result>
<binding name="Resource">
<uri>
http://palette.tudor.lu/generis/2006/palette#117213908936604
</uri>
</binding>
</result>
<result>
<binding name="Resource">
<uri>
http://palette.tudor.lu/generis/2006/palette#117213910338384
</uri>
</binding>
</result>
</results>
</sparql>
```

3.5.2 Full text search

The full text search enables to realize a keyword-based search on the ontology. Each keyword has to be separated by a comma.

Description of the REST service

This service has for input the list of keywords, and for output the list of found elements as an RDF result.

The REST service can be called this way:
`http://localhost/kb/palette/?lang=keywords\&keywords=keyword1,keyword2`

Example of use of the REST service (php)

```
$keywords = 'ctivity,omp';
echo '<a href="http://localhost/kb/palette/?lang=keywords&
          keywords='.urlencode($keywords).'">keywords test</a><br />';
```

In this case, the full text search will be done on keywords 'ctivity' and 'omp'.

To call the service, it is needed to specify the parameter 'lang' with value 'keywords' and the 'keywords' parameter with for value the keywords to search, comma separated.

Description of the SOAP service

The full text search service is also available as a SOAP service.

The SOAP full text search service takes the same input and output as the REST full text search service. The only difference is the way it is called. Indeed, SOAP as to be instantiated in order to execute the service.

Example of use of the SOAP service (php)

```
// instantiation of the soap client
// takes as parameters 'trace' to 1, the soap version, the
// uri and location of the webservices
$client = new SoapClient(null,
   array(
    'trace' => 1,
    'soap_version'  => SOAP_1_1,
    'uri' => 'http://localhost/generis/generis/GenerisKernel/',
    'location' => 'http://localhost/generis/generis/GenerisKernel/webservices.php'
    ));


// authenticate the user to access the repository
$session = $client->__soapCall('authenticate',
  array(array("palette"),array("palette"),
  array(""),array("palette")));

// specifies parameters as an array
$keywordsArray = array("ctivit", "omp");
// calls the webservice 'fullTextSearch' with parameter
// session resulting from authentication, and the array of
// keywords to search
$results =  $client->__soapCall('fullTextSearch',
  array($session->pSession,$keywordsArray));

// returns the results
print_r($results);
```

The results of the request are specified in RDF format:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Statement rdf:ID="st136675">
<rdf:subject rdf:resource="https://bscw.ercim.org/bscw/bscw.cgi/d204590/collaboration.rdfs#is-composed-of"/>
<rdf:predicate rdf:resource="http://www.w3.org/2000/01/rdf-schema#range"/>
<rdf:object>https://bscw.ercim.org/bscw/bscw.cgi/d204606/process-activity.rdfs#Activity</rdf:object>
</rdf:Statement>
<rdf:Statement rdf:ID="st2242">
<rdf:subject rdf:resource="https://bscw.ercim.org/bscw/bscw.cgi/d204590/collaboration.rdfs#is-composed-of"/>
<rdf:predicate rdf:resource="http://www.w3.org/2000/01/rdf-schema#domain"/>
<rdf:object>https://bscw.ercim.org/bscw/bscw.cgi/d204590/collaboration.rdfs#Collaboration</rdf:object>
</rdf:Statement>
<rdf:Statement rdf:ID="st68890">
<rdf:subject rdf:resource="https://bscw.ercim.org/bscw/bscw.cgi/d204590/collaboration.rdfs#is-composed-of"/>
<rdf:predicate rdf:resource="http://www.w3.org/2000/01/rdf-schema#label"/>
<rdf:object>is-composed-of</rdf:object>
</rdf:Statement>
<rdf:Statement rdf:ID="st130102">
<rdf:subject rdf:resource="https://bscw.ercim.org/bscw/bscw.cgi/d204590/collaboration.rdfs#is-composed-of"/>
<rdf:predicate rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"/>
<rdf:object>http://www.w3.org/1999/02/22-rdf-syntax-ns#Property</rdf:object>
</rdf:Statement>
<rdf:Statement rdf:ID="st147162">
<rdf:subject rdf:resource="https://bscw.ercim.org/bscw/bscw.cgi/d204590/collaboration.rdfs#is-composed-of"/>
<rdf:predicate rdf:resource="http://www.w3.org/2000/01/rdf-schema#comment"/>
<rdf:object>collaboration is composed of one or several activity(ies)</rdf:object>
</rdf:Statement>
</rdf:RDF>
```

3.5.3 XML/ RDF export

Description of the REST service

This service enables to get an RDF format view of the repository.

The service can be called this way:
http://localhost/kb/palette/

Example of use of the REST service (php)

This REST service takes no entry and sends out an RDF view of the repository.

```
echo '<a href="http://localhost/kb/palette/">get model test</a><br />';
```

Description of the SOAP service

The SOAP service takes for entry the session given back by the authenticate method, and returns an RDF format of the repository.

Example of use of the SOAP service (php)

```
// instantiation of the soap client
// takes as parameters 'trace' to 1, the soap version, the
// uri and location of the webservices
$client = new SoapClient(null,
   array(
    'trace' => 1,
    'soap_version'  => SOAP_1_1,
    'uri' => 'http://localhost/generis/generis/GenerisKernel/',
    'location' => 'http://localhost/generis/generis/GenerisKernel/webservices.php'
    ));


// authenticate the user to access the repository
$session = $client->__soapCall('authenticate',
  array(array("palette"),array("palette"),
  array(""),array("palette")));

// call of the method 'exportxmlRDF' with for parameter the
// session of authentication
$results =  $client->__soapCall('exportxmlRDF',
  array($session->pSession));
```

The two calls return the entire model RDF description:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xml:base="http://palette.tudor.lu/generis/2006/palette#"
   xmlns:ns1="http://palette.tudor.lu/generis/2006/palette#"
   xmlns:ns2="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:ns3="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:ns4="https://bscw.ercim.org/bscw/bscw.cgi/d204602/lessonsLearnt.rdfs#"
   xmlns:ns5="http://www.tao.lu/datatypes/WidgetDefinitions.rdf#"
   xmlns:ns6="https://bscw.ercim.org/bscw/bscw.cgi/d204606/process-activity.rdfs#">
<ns1:i11703234128774 rdf:about="http://10.13.1.225/slides.ppt">
   <ns3:value xml:lang="EN"></ns3:value>
   <ns2:comment xml:lang="EN"><![CDATA[ ]]></ns2:comment>
```

```
   <ns2:isDefinedBy xml:lang="EN"><![CDATA[ ]]></ns2:isDefinedBy>
   <ns2:label xml:lang="EN"><![CDATA[Slides Presentation]]></ns2:label>
   <ns2:seeAlso xml:lang="EN"></ns2:seeAlso>
</ns1:i11703234128774>
...
<ns3:Property rdf:about="https://bscw.ercim.org/bscw/bscw.cgi/d204606/process-activity.rdfs#supplies">
   <ns2:comment><![CDATA[an outcome can supplie resources]]></ns2:comment>
   <ns2:domain resource="https://bscw.ercim.org/bscw/bscw.cgi/d204606/process-activity.rdfs#Outcome"/>
   <ns2:label><![CDATA[supplies]]></ns2:label>
   <ns2:range resource="https://bscw.ercim.org/bscw/bscw.cgi/d204590/collaboration.rdfs#Resources"/>
</ns3:Property>
</rdf:RDF>
```

3.5.4 Get description of an element (fetch method)

This service enables to get the description of an element of the ontology based on its URI.

Description of the REST service

This service takes for input the URI of the element and for output the RDF description of the element.

The REST service can be called this way:
http://localhost/kb/palette/?lang=fetch&r=#311721388277034
where #311721388277034 is the uri of the element to fetch.

Example of use of the REST service (php)

```
// element to fetch
$res = '#11721388277034';

echo '<a href="http://localhost/kb/palette/?lang=fetch&r='.urlencode($res).'" >
      fetch test</a><br />';
```

Description of the SOAP service

The SOAP service takes as entry: the session of authentication and the URI of the element to fetch.

Example of use of the SOAP service (php)

```
// instantiation of the soap client  takes as parameters 'trace' to 1,
// the soap version, the  uri and location of the webservices
$client = new SoapClient(null,
  array(
  'trace' => 1,
  'soap_version'  => SOAP_1_1,
  'uri' => 'http://localhost/generis/generis/GenerisKernel/',
  'location' => 'http://localhost/generis/generis/GenerisKernel/webservices.php'
  ));

// authenticate the user to access the repository
$session = $client->__soapCall('authenticate',
  array(array("palette"),array("palette"), array(""),array("palette")));
```

```
// call of the method 'getXMLDescription' with for parameter the session
// of authentication and the URI of the element to fetch
$r = "#11721388277034";
$results = $client->__soapCall('getXMLDescription',array($session->pSession,$r));
print_r($results);
```

The result obtained (using REST or SOAP calls) is the RDF description of the element:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Statement rdf:ID="st11240">
<rdf:subject rdf:resource="#11721388277034"/>
<rdf:predicate rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#type"/>
<rdf:object>https://bscw.ercim.org/bscw/bscw.cgi/d204606/process-activity.rdfs#Process</rdf:object>
</rdf:Statement>
<rdf:Statement rdf:ID="st105511">
<rdf:subject rdf:resource="#11721388277034"/>
<rdf:predicate rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#value"/>
<rdf:object></rdf:object>
</rdf:Statement>
<rdf:Statement rdf:ID="st42793">
<rdf:subject rdf:resource="#11721388277034"/>
<rdf:predicate rdf:resource="http://www.w3.org/2000/01/rdf-schema#comment"/>
<rdf:object> </rdf:object>
</rdf:Statement>
<rdf:Statement rdf:ID="st94427">
<rdf:subject rdf:resource="#11721388277034"/>
<rdf:predicate rdf:resource="http://www.w3.org/2000/01/rdf-schema#label"/>
<rdf:object>Process 1
</rdf:object>
</rdf:Statement>
<rdf:Statement rdf:ID="st164146">
<rdf:subject rdf:resource="#11721388277034"/>
<rdf:predicate rdf:resource="http://www.w3.org/2000/01/rdf-schema#seeAlso"/>
<rdf:object></rdf:object>
</rdf:Statement>
<rdf:Statement rdf:ID="st146309">
<rdf:subject rdf:resource="#11721388277034"/>
<rdf:predicate rdf:resource="http://www.w3.org/2000/01/rdf-schema#isDefinedBy"/>
<rdf:object> </rdf:object>
</rdf:Statement>
<rdf:Statement rdf:ID="st72033">
<rdf:subject rdf:resource="#11721388277034"/>
<rdf:predicate rdf:resource="http://www.w3.org/2000/01/rdf-schema#member"/>
<rdf:object></rdf:object>
</rdf:Statement>
</rdf:RDF>
```

3.6 Conclusion

In this chapter, we present the basic KM services based on Generis. These services provide low-level operations of knowledge management. These services are used by the complex KM service BayFac presented in chapter 5. They will also be used by developers of Palette CoPs who need KM capabilities for their services.

Chapter 4

An illustration of the use of the basic services: ECCO

In this chapter we illustrate how the basic KM services can be combined within a complex KM service such as ECCO. ECCO (in French: Editeur Collaboratif et Contextuel d'Ontologies) is an ontology editor effectively used in Palette. It is a tool for collaboratively creating contextualised ontologies (`http://argentera.inria.fr/ecco/index.jsp`). In Deliverable D.KNO.02 [27] (Chapter 11), ECCO has been described in terms of its constituent modules, which reflect the main steps of the workflow implemented in this ontology editor. In the current Deliverable, we will describe ECCO in terms of the basic KM services described in the chapter 2.

Our illustration of the combination of basic services in ECCO will mainly be functional. We will describe the different steps of the process of ontology creation implemented in ECCO, and for each of these steps show which basic KM services are used.

4.1 Knowledge Creation service – Cooperative Knowledge Creation service

The main services which can underlie a complex service like ECCO are the Knowledge Creation service, and the Cooperative Knowledge Creation service. The Knowledge Creation service is used to support the creation of ontologies, following a number of prescribed steps, or workflow, from identifying knowledge sources to formalizing the ontology using a formal language (see Figure 4.1).

Figure 4.1: The ECCO ontology creation workflow

During this process, an engine is associated to each user, it contains the constructed ontologies, the annotations associated to the knowledge sources, the rules added by the user. The creation and the management of this engine use the Administration service 2.2 to achieve the needed operations.

The cooperative service is used, e.g., for identifying contributors to the ontology creation, for identifying which elements (term, definition/context) have been created by which contributor (using differently coloured highlighting, see Figure), for filtering information to be

presented according to the user, for discussing and arguing the choice of terms (see Figure 4.2), definitions, etc.

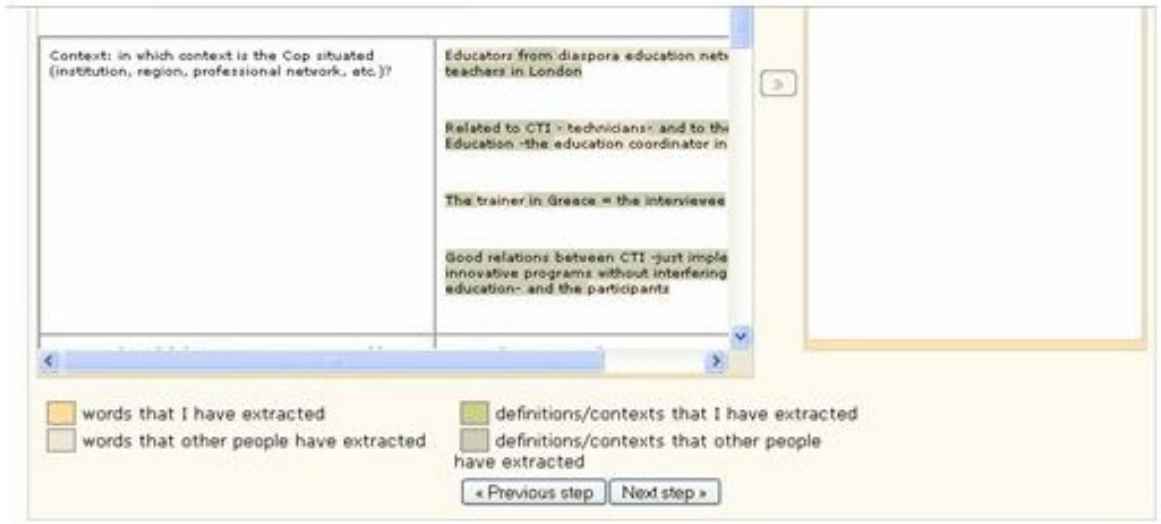


Figure 4.2: Identification of which elements have been created by which contributors

The identified elements are stored, at this step, in a flat ontology containing the terms and their contexts. The creation and management of this ontology is achieved using the ontology management services 2.3 (*e.g.* the addConcept/removeConcept operations are used to deal with terms and addDefinedBy/removeDefinedBy to manage contexts). In addition, annotations are created to associate terms and knowledge sources, these annotations are managed using the Annotation service 2.4.

The cooperative service allows users to be aware of which other users are connected to the editor, and of what is their "profile" or role (see Figure 4.3).

Figure 4.3: Awareness of connected users

Some collaborative aspects of ECCO appear also in other services underlying the ECCO ontology editor, for example the Knowledge Annotation service.

4.2 Knowledge Annotation service – Knowledge Retrieval service

Two other services can underlie a complex service like ECCO: a Knowledge Annotation service and a Knowledge Retrieval service. In ECCO, the vocabulary is manually annotated (a term can be annotated as a concept or as a relation, as to be discussed or as validated, etc.). Some search can be performed on the annotated vocabulary, and some information about the terms can be retrieved. A main function of the retrieval service is to test the relevance of ontologies.

For each term, the users constructing the vocabulary collaboratively can interact by discussing the interest of a term, the relevance of its synonyms and/or of its definitions, and by proposing modifications and explaining why a modification has been made (Figure 4.4).

Figure 4.4: Tagging the lexicon

Each discussion in ECCO is managed through RDF annotations that can be attached to each term. Each annotation encompasses the content of the comment, the name and "profile" (or role) of the author of the comment, and the creation date of the comment. These annotations are managed using the Annotation service 2.4.

4.3 Knowledge Presentation and Visualization service

A fifth service which can underlie a complex service like ECCO is a Knowledge Presentation and Visualization service. The knowledge that can be presented and visualized is primarily concepts and relations of the ontology. These elements can be presented using Natural Language or a formalized language such as RDFS. These elements can be visualized graphically as "indented trees" (see Figure 4.5). The export operations of the Retrieval service 2.5 (such as getRootConcepts/getChildrens) are used to retrieve these elements.

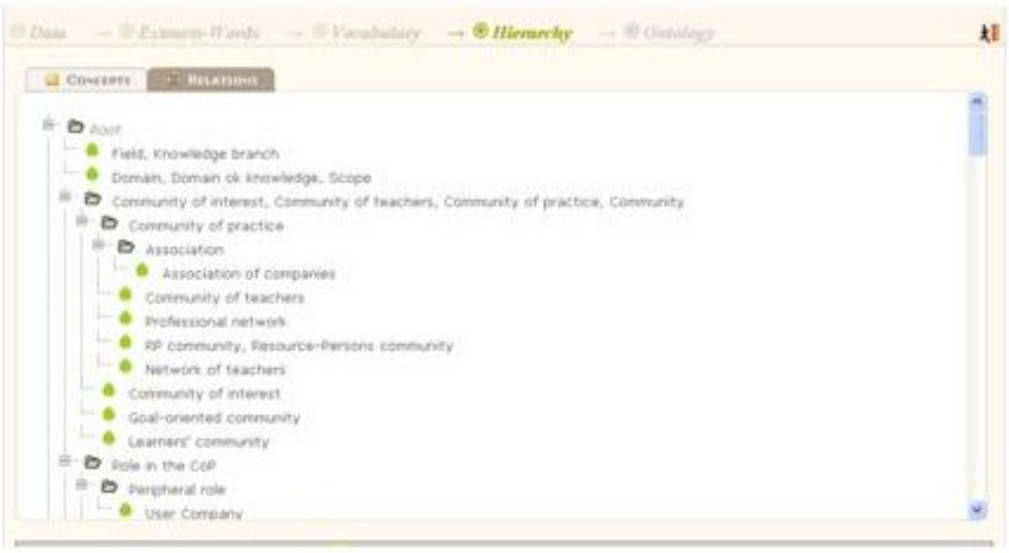


Figure 4.5: Visualizing a concept hierarchy in ECCO

Discussions about terms of the vocabulary can also be displayed. A user can consult these discussions via the interface by navigating through the vocabulary (see Figure 4.6). The "profile" (or role) of the users contributing to the discussion (e.g., a domain expert, see Figure 4.7) is represented by an icon.

Figure 4.6: Posting a message (comment, etc.) about a word from the vocabulary

Figure 4.7: Displaying a discussion

4.4 Knowledge Evaluation service

Another service which can underlie a complex service like ECCO is a Knowledge Evaluation service. This service may be used to assist domain experts in validating the various ontology elements. This validation use a set of queries to retrieve information about the created ontologies. These SPARQL queries are submited to the Retrieval service (2.5) and the result are displayed in order to allow users to validate their ontologies.

(see the ECCO "Ontology Tester" in Figure 4.8; this "tester" is currently under... testing)

4.5 The use and development of a complex service like ECCO in Palette

We have just illustrated how some basic KM services can be combined within a complex KM service such as ECCO. This illustration is not purely "academic". Developing an ECCO-like complex service based on actual basic KM services is currently discussed within Palette. words,

Among the development issues raised are the following ones:

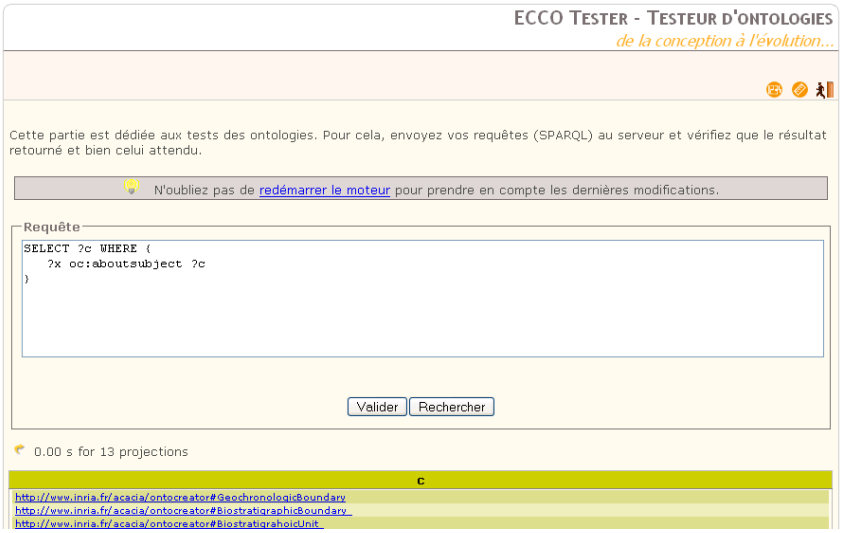


Figure 4.8: Testing ontologies in ECCO with the "Ontology Tester"

- Implementing an ECCO-like complex service dedicated the CoPs. We could envisage a "lighter" service than the current ECCO; a kind of "intermediary" service, between the CoP and some knowledge engineer external to the CoP, who will use the full complex service.
- Integrating the Knowledge Evolution and Maintenance service to ECCO.
- Connecting ECCO with SweetWiki: The SweetWiki tagging operation can be compared to the ECCO operation of selecting candidate terms for the ontology: both operations are performed on a given document; the output of both operations is a term or a phrase (plus a context for ECCO); the output of SweetWiki can be used as an input for ECCO; if so, both tools can be connected, or need to interoperate.
- Connecting ECCO with CoPe_It!!: With CoPe_It!! users can perform argumentation activities; the argumentation service of CoPe_It!! can be used in the context of an ECCO-like complex service, to the support the argumentation activities performed during the creation of an ontology.

If discussed, all these development issues are not to be solved in the context of Palette.

Chapter 5

Description of upcoming complex KM services

In this chapter, we will describe two complex KM services, that are used by Palette CoPs:

SweetWiki is semantic wiki used by many Palette CoPs to collaboratively produce and annotate documents. It relies on the basic KM services described in chapter 2 to achieve low level operations (*e.g.* tag creation use the operation addConcept to add a tag to the folksonomy).

BayFac is resources classification service, used by the Form@Hetice CoP, to classify the resources (documents, emails, ...) produced in the CoP. It relies on the basic KM service described in 3.

5.1 SweetWiki

5.1.1 What is Sweetwiki?

SweetWiki [5, 4] is a new wiki engine. In addition to its relying on the features implemented by most of the common wikis, such as the mechanisms of "WikiPages" and sometimes "WikiWebs" that are materialized into a hyperlink structure, SweetWiki also relies on semantic web technologies, thus providing additional and powerful structuring mechanisms. Indeed, SweetWiki makes use of:

- *An ontology of the wiki structure (the wiki Object Model: using OWL Lite, this ontology describes SweetWiki concepts, properties and relationships, such as "Document", "WikiPage", "KeyWord", "Link", "Backward link", "Author", "Version", "Attached file", "Attached picture", etc. The corresponding metadata are embedded in the wiki pages themselves. Making this structure and its ontology explicit allows to reason on it to generate widgets for helping the navigation (e.g. list of the related pages), for instance. This ontology can be modified and maintained by the wiki developers (e.g. re-engineer the wiki structure).*
- *An ontology of the topics (the Domain ontology): which enables the realization of the "social tagging" principle provided by SweetWiki. The pages and their attached documents (pictures, videos and attached files) can be tagged from within the editor, using the domain ontology formalized using RDFS. Thus, in the context of Palette, a CoP member can indicate that a page covers a particular field of knowledge of the CoP and that it is related to a particular activity handled in the CoP, for example. This mechanism is very simple to utilize and, at the same time, eases the navigation by reasoning on it (e.g. for finding the pages that are tagged with a concept, finding the semantically close concepts, formulating complex queries). This ontology can*

be modified by the wiki users (enriched directly by them and may be restructured by volunteers of the CoP or persons having a particular role in the CoP, so as to improve the navigation and querying capabilities) through the ontology editor that comes with SweetWiki.

Moreover, if a CoP needs a specific additional ontology that is already available in RDFS or OWL Lite (e.g. mathematics ontology), this ontology can be loaded into the underlying semantic web server of SweetWiki and then, becomes directly accessible to the users.

Architecture

Starting from the users' side, SweetWiki is based on Kupu[1], an XHTML editor in Javascript that provides a WYSIWYG interface in the user's browser. Since editing directly produces XHTML, we decided to use it as a persistence format. Thus, once saved, a page stands ready to be served by the Web server.

To address structuring and navigation problems in wikis, we wanted to include tagging at the core of the wiki concept, thus we integrated four new web technologies:

- RDF/S and OWL are W3C recommendations to model metadata on the web[2];
- SPARQL[3] is a candidate recommendation for a query language for RDF;
- RDFa[4] is a draft syntax for embedding RDF in XHTML;
- GRDDL[5] is a mechanism for getting RDF data out of XML and XHTML documents using explicitly associated transformation algorithms, typically represented in XSLT.

With RDFa, we have both page data and metadata in the same standalone XHTML file. Therefore, the pages can be crawled by external applications or saved by users using their browser without any loss of information.

Besides the topic tags, metadata include contextual information (e.g. page author, last modification on the page, etc.).

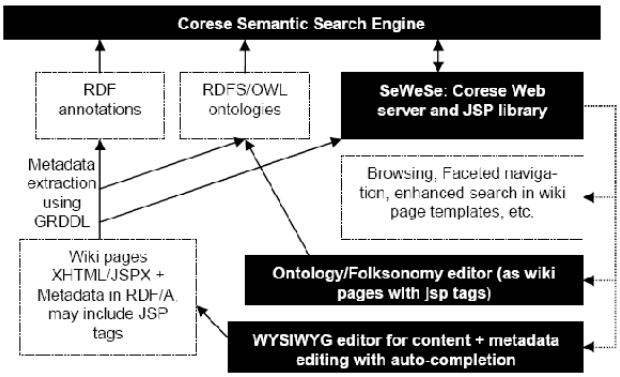


Figure 5.1: SweetWiki architecture

Figure 5.1 summarizes the architecture of SweetWiki. The implementation relies on the CORESE [7] semantic search engine for querying and reasoning and on SeWeSe [12], its associated web server extension that provides API and JSP tags to implement all the web-based interfaces that use ontologies, as well as a set of generic functionalities (security management, ontology editors, web application life cycle, etc.).

[1]Kupu: `http://kupu.oscom.org`
[2]W3C, Semantic Web Activity, `http://www.w3.org/2001/sw` and `http://www.w3.org/2001/sw/Activity`
[3]SPARQL Query Language for RDF `http://www.w3.org/TR/rdf-sparql-query`
[4]RDF/A Primer 1.0 `http://www.w3.org/2001/sw/BestPractices/HTML/2006-01-24-rdfa-primer`
[5]Gleaning Resource Descriptions from Dialects of Languages `http://www.w3.org/2004/01/rdxh/spec`

SweetWiki currently provided functionalities

1. Page editing

   As mentioned previously, SweetWiki uses the Kupu editor that has been considerably extended for the purpose of supporting metadata editing and semantic extensions, thus it enables:

   - To easily link a page to other internal pages using wizards that execute SPARQL queries for getting the list of existing pages. This mechanism can be used to:
     - Insert a forward link directly into a page content;

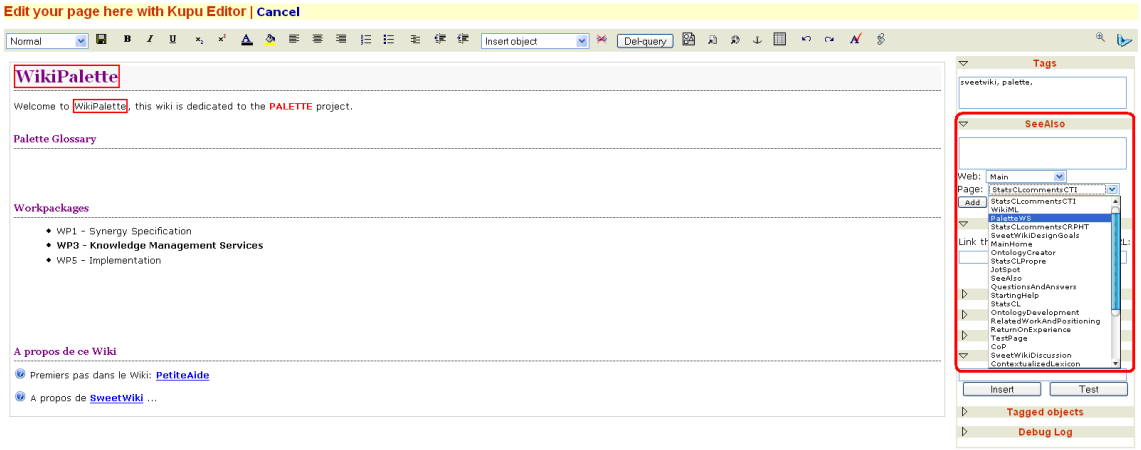


Figure 5.2: Insertion of a “See also” link to a page that is related to the current one

     - Insert a “See also” link to the page (see Fig. 5.2): the purpose of this metadata is to suggest the reader of the page to consult the page mentioned in the “See also” field that is shown in the footer of the current page (see Fig. 5.3). We observed that this manual-made operation is not often processed, which strengthens the relevance and usefulness of providing automatic and meaningful operations based on the information provided by the tags on the pages.

Figure 5.3: Result of the operation: layout of the new metadata

- To tag a page or parts of it (like included pictures or attached files): as the user types, an auto-completion mechanism (AJAX calls from within the editor) suggests existing keywords by issuing SPARQL queries to the semantic web server in order to identify existing concepts with compatible labels, in the underlying domain ontology. It shows the number of other pages sharing these concepts as an incentive to use them. Furthermore, related categories are also displayed in order to address the ambiguity of homonymy. With this approach, tagging remains easy (keyword-like) and becomes both motivating and unambiguous. Unknown keywords are collected and attached to new concepts to enrich the ontology. Later on, a person (or more) handling a particular role in the CoP may reposition them in the ontology (see Fig. 5.4);

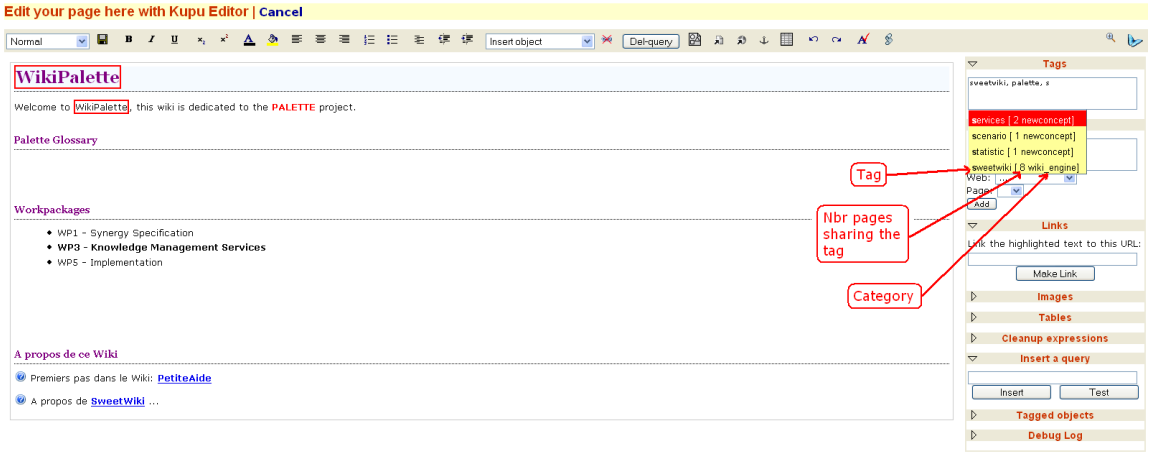


Figure 5.4: Tags are suggested as the user enters a keyword, the number of pages sharing that tag is displayed as well as the related category

  In addition to the fact that tagging enables to find easily the tagged document when searching for it, it also enables to have access to other documents tagged with concepts related to the one(s) used to tag the document. Indeed, when a page is saved, its metadata are extracted using the semantic web server API. This API uses a GRDDL XSLT stylesheet to extract the metadata in RDF/XML format and feed them to the CORESE engine, which generates faceted navigation widgets (see Fig. 5.5): the semantics of the tags is used to derive related topics, query the engine on similar pages using SPARQL queries, etc.

- To embed SPARQL queries into a page being edited (they will be translated at save time into JSP tags from the SEWESE tag lib). From the editor, SPARQL queries can be tested and validated before being inserted in the page, as illustrated by Figure 5.6. In addition, a query can be embedded in a page but directed to other SPARQL servers than the one of SweetWiki, thus allowing users to include results from external sites.

2. Office document integration

   SweetWiki comes with a module to import Open Office or Microsoft Office documents (in the current version, word processor files as well as spreadsheet files are supported). The documents are automatically translated into SweetWiki pages (including the pictures, etc.) that can in turn be edited, tagged and shared. The metadata available in these documents can be exploited as well. An Open Office server and ad hoc conversions modules are used behind the scene for converting the office documents.

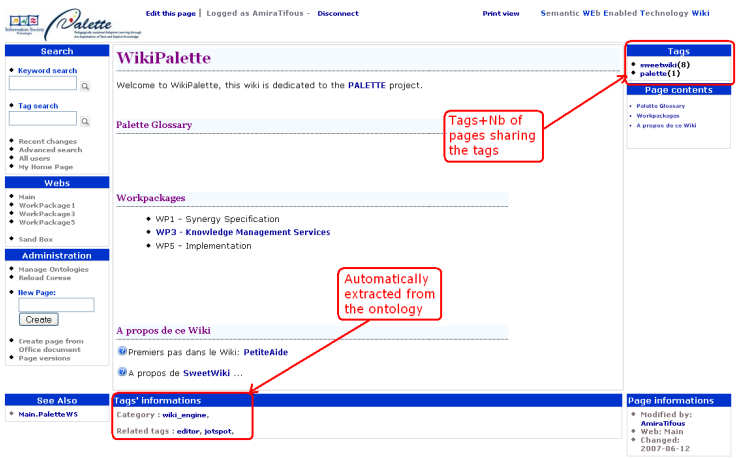


Figure 5.5: Faceted navigation links extracted from the tags

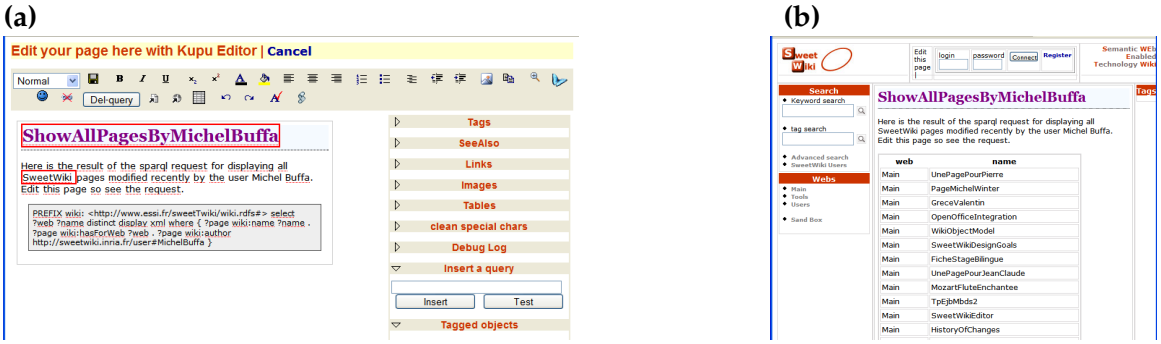


Figure 5.6: Example of an embedded SPARQL query, in the WYSIWYG editor (a). The query is in the greyed square and can be edited, tested and validated without leaving the editor. Once the page saved (b), the result of the request is displayed in a table (default presentation)

3. Querying on SweetWiki

   SweetWiki provides two means of searching for information into the contents of its pages:

   - A text based search, using the Lucene search engine, which processes a google-like search on the wiki pages;
   - A tag based search that relies on the Corese semantic search engine. As the user types the keyword, an auto-completion mechanism (still AJAX calls from within the search form) suggests existing keywords by issuing SPARQL queries to the semantic web server in order to identify existing concepts with compatible labels (see Fig. 5.7), in the underlying domain ontology, and shows the categories they are related to, as well as the number of other pages sharing these concepts. SweetWiki offers the possibility to submit simple queries made up of a list of terms and applies an OR operator to this list. In other words, when the user submits more than one term in the query, SweetWiki returns for each term of the query, the list of the documents (pages, images, etc.) that are tagged with it (see Fig. 5.8).

4. Ontology editor for maintaining and re-engineering the ontology/folksonomy

   In order to maintain and re-engineer the folksonomy, SweetWiki reuses web-based editors available in SeWeSe. Using these editors, the folksonomy and the annotations may

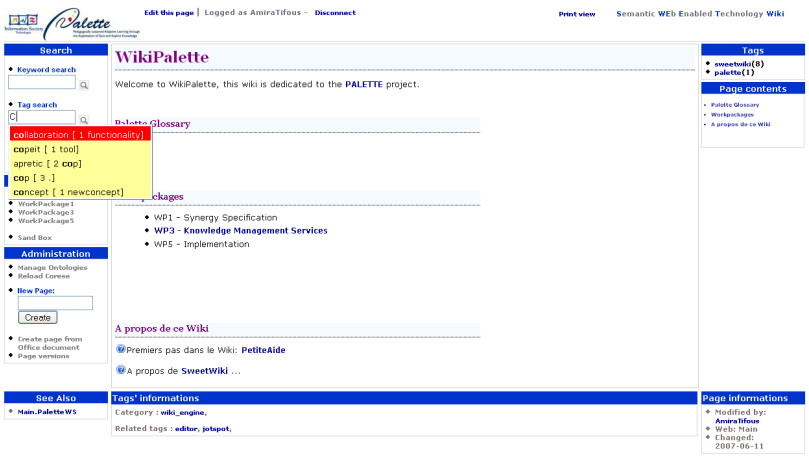


Figure 5.7: Tags are suggested as the user enters a keyword for her query

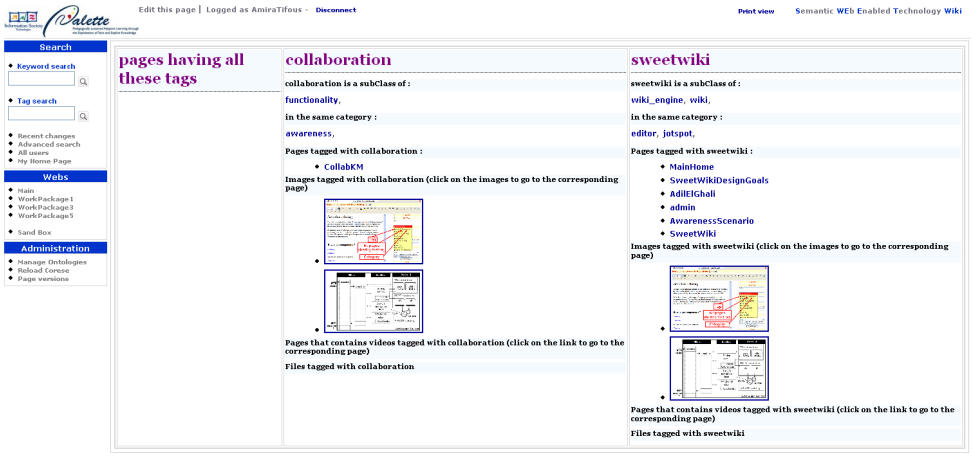


Figure 5.8: Results of the query “Collaboration, SweetWiki”: documents tagged with one of these keywords are listed, the category of each keyword is returned as well as the concepts belonging to the same category as the keyword

be updated. For instance, one can add/remove/edit concepts, community experts can pick a couple of tags and declare semantic relations between them such as subClassOf. They may also merge concepts when two tags are synonymous, etc.

Enhancements of the ontology seamlessly improve content sharing: search and faceted navigation benefit directly from the updates.

If a concept is suddenly missing from the folksonomy, it still remains as a tag for the pages it has been used to tag before being deleted, and it re-appears in the folksonomy, where it is just processed as a new tag.

5.1.2 Use by Palette CoPs

Example of scenarios

@pretic:

Expressed needs: Lot of knowledge is contained in the mailing-lists archives, but useless because of the difficulty of retrieval/access to it; this also makes the knowledge provided by the members lost whenever they leave the community. The community’s memory is not capitalized.

Actors involved in the scenario (of use of SweetWiki): @pretic's members, including: 'core members', 'near-core members' and 'satellite members'.

Tasks to be performed using SweetWiki:

- Producing and annotating wiki pages that contain digests of the mailing-lists archives, using the editor and tagging functionality of SweetWiki, on the basis of their topics (no particular expertise is required), in order to synthesize the knowledge accumulated since the beginning of the CoP. These pages are annotated according to the CoP ontology;
- Producing new knowledge (this is done by the members who have a certain level of expertise in some domains) using the editor as well as tagging functionality;
- Organizing the tags (folksonomy and ontology) used to qualify the produced knowledge (this is done by core members) by means of the ontology management functionality provided by SweetWiki;
- Queries are submitted on the wiki content using the tag-based search functionality provided by SweetWiki;
- Knowledge is also explored using the faceted navigation relying on the tags, through the wiki pages.

As knowledge is provided and grows through more and more tagged pages on SweetWiki, the need for efficient mechanisms of knowledge retrieval is met and the benefits of the tag-based as well as text-based search provided by SweetWiki are recognized.

Finally, the need for awareness mechanisms is expressed: it would be worthy to make the CoP members aware of the changes that occur in the wiki content, depending on their interests, so that they are motivated to participate.

Learn-Nett:

Expressed needs: a huge amount of knowledge has been produced and gathered for 10 years through the different activities handled within the CoP, thus leading to an important need to facilitate the access to these resources.

Actors involved in the scenario (of use of SweetWiki): Tutors.

Tasks to be performed using SweetWiki:

- Identifying the main sections of SweetWiki, which represent some activities held within the CoP or some important general topics;
- Some former resources produced by the CoP are transformed into wiki pages (eventually using the functionality of transforming MS or OO documents to wiki pages) and tagged according to their targeted audience, type and their thematic. These different aspects are formalized into a LearnNett-specific ontology (a specialisation of the O'CoP ontology, made by some members of the CoP). This enables to get a dynamic portal for Learn-Nett SweetWiki sections;
- New resources (monthly and annual reports, synthesis of recurrent problems met by tutors) are produced using SweetWiki editor and based on predefined templates, and are also tagged using Learn-Nett ontology;
- Knowledge is explored using the faceted navigation relying on the tags, through the wiki pages;

- Knowledge is also accessed by submitting queries on the wiki content using the tag-based search functionality provided by SweetWiki.

In addition to these functionalities, relying on the ontology concepts to annotate the resources created on SweetWiki makes it possible to have a dynamic page related to each section of the wiki

ePrep:

Expressed needs: a dictionary for the students in CPGEs in the fields of mathematics, physics, chemistry and English.

Actors involved in the scenario (of use of SweetWiki): Teachers produce knowledge and students have the possibility to access it.

Tasks to be performed using SweetWiki:

- Creating the sections of the wiki;
- Producing new knowledge by means of SweetWiki editor;
- Reusing existing knowledge using the functionality of transforming MS or OO documents to wiki pages;
- Annotating the pages with existing tags, creating new tags when needed, organizing the tags (using the ontology management functionality);
- Submitting tag-based queries to the wiki and exploring the wiki content using the faceted navigation using the tags.

The stakeholders of this scenario should be provided with awareness mechanisms, enabling them to receive notifications on what they're interested in (teachers as well as students) and also on the changes occurred on the pages they created (teachers).

First observations

Several CoPs of Palette have requested to use SweetWiki. Thus, several instances of SweetWiki have been deployed. At the time this deliverable is being written, the CoPs that use SweetWiki are the following (ordered by date of SweetWiki start of use):

Name of the CoP	Start of use	Sweetwiki instance
ePrep	14/12/2006	`http://argentera.inria.fr/wikiprepas/data/Main/MainHome.jsp`
STE-CRIFA (ULg)	14/02/2007	`http://argentera.inria.fr/swikiulg/data/Main/MainHome.jsp`
@Pretic	27/02/2007	`http://argentera.inria.fr/swikiapretic/data/Main/MainHome.jsp`
Form@Hetice	19/04/2007	`http://argentera.inria.fr/swikifh/data/Main/MainHome.jsp`
ADIRA	26/04/2007	`http://argentera.inria.fr/swikiadira/data/Main/MainHome.jsp`
TFT	07/05/2007	`http://argentera.inria.fr/swikitft/data/Main/MainHome.jsp`
CdE	08/05/2007	`http://argentera.inria.fr/swikiCdE/data/Main/MainHome.jsp`
Learn-Nett	07/06/2007	`http://argentera.inria.fr/swikiln/data/Main/MainHome.jsp`

In spite of the fact that STE-CRIFA is not a Palette CoP, it is yet a community of researchers and assistants from the University of Liège (ULg), which is one of the partners involved in the Palette project. In addition, these researchers adopted a CoP-like functioning in order to "simulate" a CoP and make their researches more accurate and realistic.

As for TFT (Transition – Formation – Travail), it is an emerging CoP involved in the Palette project and initiated by the CRIFA centre of ULg. This CoP aims at enabling a collaborative work and exchanges on the issues that concern the training of nurses-students. It is composed

of persons in charge of welcoming the trainees as well as the teachers that supervise their trainings.

Finally, CdE is a new emerging Palette CoP constituted of persons who accompany new projects of enterprise creation. This CoPs aims at enabling its members to exchange about their experiences and formalise their knowledge and know-how.

Statistics and information on the use of SweetWiki by different CoPs

In the following, we present some statistics on the use of SweetWiki. We'll rely on the instances that have been implemented the earliest, considering that their users have had the time to get used to SweetWiki functionalities. Thus, we'll analyze the use of SweetWiki by ePrep, STE-CRIFA and @Pretic. This analysis is made for a period extending from May 13th till June 12th for ePrep and @pretic, and from June 2nd till June 14th for STE-CRIFA.

We'll rely on the statistics provided using Google Analytics for the following indicators:

Distribution of the number of visits/sessions indicates the number of unique sessions initiated by the wiki visitors. If a user is inactive on the wiki for one hour and 30 minutes or more, any future activity will be attributed to a new session. This number gives a clue on the frequentation and popularity of the wiki.

Time on Site for all visitors is one way of measuring visit quality. If visitors spend a long time visiting SweetWiki, they may be interacting extensively with it. However, Time on Site can be misleading because visitors often leave browser windows open when they are not actually viewing or using the wiki.

Bounce rate is the percentage of single-page visits (i.e. visits in which the person left SweetWiki from the entrance page). Bounce Rate is a measure of visit quality and a high Bounce Rate generally indicates that wiki entrance pages aren't relevant to the visitors.

Average pageviews Average Pageviews is another way of measuring visit quality. A high Average Pageviews number suggests that visitors interact extensively with SweetWiki. In the case of SweetWiki, a high Average Pageviews results from the high quality content as well as the involvement of the users (since they are the knowledge producers on SweetWiki). Conversely, a low Average Pageviews indicates that the wiki does not deliver what was promised to the visitor and that they do not participate much to enrich it. We'll provide statistics on this indicator only if needed to provide more details on the general observations.

Top Content Which are the most commonly viewed pages on SweetWiki?

The detailed statistics of use by the CoPs are presented in appendix C.

5.1.3 Functionalities to be developed

Synthesis of the needs expressed concerning SweetWiki functionalities

The table below summarizes the main functionalities not yet provided by SweetWiki and for which needs have been expressed by Palette CoPs. They are illustrated based on the three scenarios described previously as well as relying on the usability analysis[6] performed in the context of WP1 (see the first feedback on the usability of SweetWiki: `https://bscw.ercim.org/bscw/bscw.cgi/302122`).

We categorize these functionalities, according to their implementation achievement and based on their respective priority degree and required effort, in:

[6] To be included in the deliverable D.PAR.04 of WP1 (M24).

- Mid-term developments that will be achieved in the context of the Palette project;
- Long-term developments that will be achieved in the context of the Palette project;
- Perspectives: the developments that may be considered to enhance SweetWiki after Palette.

	@pretic/TFT/ULg	ePrep	Learn-Nett
Mid-term achievement	- Subscription to awareness queries - Enhanced look & feel (interface graphics and organization)	- Subscription to awareness queries	- Templates for the pages depending on the activities they're related to
Long-term achievement	- Enhanced look & feel (interface language, page management – renaming and suppression) - Enhanced ontology management - Enhanced page versioning ("Diff")	- Enhanced look & feel (interface language) - Enhanced ontology management	- Enhanced ontology management
Perspectives	- Subscription to complex awareness queries - Enhanced editor	- LateX imports	

To these functionalities, additional ones have been identified in order to improve the wiki usability and effectiveness, so as to improve learning using SweetWiki. According to their nature, we can classify the whole functionalities into three main categories:

1. **Interface related functionalities** The aim is to provide parameterized look & feel, through the use of templates for the pages. These templates would be designed depending on the Web[7] they belong to, for instance. The use of templates would enable to reuse pages components. In addition, another improvement will be to make the interface of SweetWiki adapt to the user's preferred language, defined in her profile. Furthermore, page management functionalities will be provided, allowing to rename a page or delete it when becoming deprecated. This last would be allowed to the administrator of the wiki.
2. **Tagging**
   (a) Tag versioning: So as to allow 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 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).
   (b) Improve tagging and tag-based search: Tagging improvement by enabling the authenticated users to tag the wiki pages without imposing them to activate the pages editor; As for the tag-based search, enabling the submission of complex tag-based queries by:
      - using the AND and OR logical operators;
      - providing a query interface that allows the user to parameterize the query by showing, for each concept chosen to formulate the query, its attributes and relations to other concepts.

[7] A section/ workspace in the wiki. It can correspond to an activity handled within the CoP, for instance.

(c) (Semi)-Automatically organize tags: The idea behind is to help the user in charge of maintaining the ontology in this task, providing her with assistance for reorganizing the ontology, based on the use of the tags in the wiki (webs and pages), their use amongst other tags, 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.

(d) Enhancing tags management: Consists of making the ontology edition more user-friendly (ergonomics, drag & drop mechanism for structuring the ontology, enabled multiple inheritance, etc.). This functionality of SweetWiki relies on the Hierarchy interface and Ontology editor of ECCO (see Chapter 4).

3. **Awareness functionality** Enabling the users to subscribe to the set of predefined queries that meet their respective needs, so as to receive notification mails with information about the changes on the wiki content and the statistics that they are interested in.

5.2 BayFaC: Bayesian Faceted Classifier

5.2.1 Service description and functions

This service aims at providing a mean to semi-automatically index textual documents (documents, emails, forum posts, wiki, blogs, etc.) regarding a vector of concepts relevant to a CoP, hence allowing classification according to multiple facets. These concepts will typically come from the domain ontology of the CoP, but also from other ontologies specifying *e.g.* the tools used, the kind of knowledge shared, etc. The benefits for the users are to have incoming documents automatically classified according to known useful categories, and to be able to search information in a more efficient manner thanks to this indexation. A user interface exploiting this indexation will provide ways to navigate in the documents (knowledge) base, to search, and classify new documents, in a very intuitive way, easier than by using the ontology directly, while keeping the benefits of having it in the background for keeping semantic relations between concepts and allowing further inference to be performed.

BayFac relies on two theories, namely the faceted classification [23], and the Bayesian approach to classification [2, 6]. Faceted classification consists in clustering the categorization space (which in our case is fed by ontologies) into different subspaces of linked concepts, then allowing to link a document to a set of concepts, belonging to known facets, instead of a single one. This allows the classification of documents into a multi-dimensional categories space, which has proven to be of very intuitive use and very efficient for browsing and searching tasks. Bayesian classifiers, will be used to provide automatic categorization of documents in each facet. They will learn first from manual classification made by users, and then propose automatic categorizations once there is a sufficient amount of documents processed in a supervised manner to enable correct predictions.

Concretely, the service will be deployed on a Generis platform and made available to the external world in the forms of:

- a web service, allowing direct use by software applications, for classification (manual or automatic) of documents, according to a set of facets relevant to the CoP domain (which will have to be determined), and for search of documents according to a vector of terms coming from the facets;

- a web portal allowing browsing and search through the documents (knowledge) base, manual and automatic classification, all using the facets.

The Bayesian classifiers, as well as facets, will be hosted by the platform and each CoP using the service will have its own classifiers and facets. Each CoP willing to use the service will have the possibility of using either the web service or the web portal for classifying its documents (knowledge). Whatever the interface used for accessing the service, there are then two possibilities of use:

- For providing good results, the classifier will need to learn from a big amount of manual classifications, thus the best way to start is that some CoP members manually classify as much as possible of the CoP already existing document corpus, thus allowing the classifiers to learn. In this case, automatic classifications will then be proposed for each new document. This task might potentially be performed (semi-)automatically using text analysis tools, but since this would provide machine classifications, the results further provided by the classifiers will inherently be biased. Thus it is better to rely on manual categories assignments.
- Another possibility is to use the service directly with new documents. In this case, the classifiers will not provide any proposition at the beginning, but once a sufficient amount of document has been manually classified. The users still benefit from the facetted browsing and searching facilities.

Whatever the initial choice, the more the classifiers will learn from manual classifications, the more they will be efficient and precise in their predictions. The user will also have the possibility to change a classification, automatically provided or not.

Screenshots

A prototype of BayFac has been developed for the *Form@Hetice* CoP, and is available for testing. Here are some screenshots taken from it.

Figure 5.9: BayFac browsing screen

Figure 5.9 shows the browsing screen of BayFac. Facets appear on the left side, together with a field for keyword-based searching. Figure 5.10 shows the content of a facet "Type de document", from which the user chooses values for performing its classification. Results are displayed on the right side, once the user has clicked on the search button.

Figure 5.10: Choosing a facet value

Figure 5.11 shows the classification screen. Once a document has been uploaded in the system (only a link to its actual location is kept), it can be classified by the user, which is automatically given suggestions provided by the Bayesian classifiers, or others when it is simpler to use another kind of classifier.

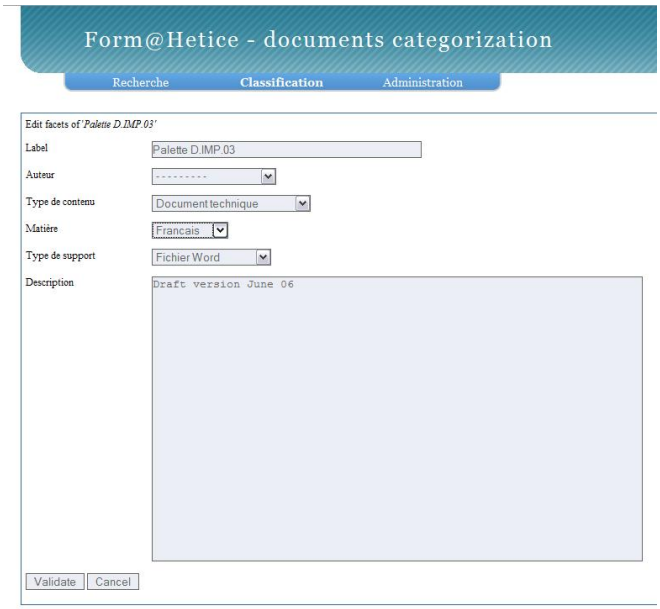


Figure 5.11: Classification of a document

5.2.2 Theoretical Backgrounds

Semantic Web ontologies are efficient and useful models describing the semantics of a domain, which can be then used to describe or annotate objects or resources on the web according to concepts having a formally defined semantic. Although ontologies are often designed

by specialists, their primary goal is to serve as a basis for machine reasoning and further automation. When used directly by humans, for *e.g.* annotating a resource, finding his way through the potential complexity of an ontology can become a very fastidious task.

Navigation and search among some object corpus can obviously be performed thanks to annotations when they are available. However, a simple classification of the objects can be of much easier use for the end-user. We propose here to combine the advantage of having objects semantically described according to an ontology and that of using a specific and intuitive classification based on facets. The classification scheme being expressed itself in an ontology form, both the processes of annotating and classifying a document can take benefit from each other as any instance created will be a class or property defined in the ontology. Additionally, the indexing of the classified resources is done in such a way that the inverted index is directly instantiated in the ontology of the corpus of all those resources, allowing thus good performances during search and browse, as no complex inference is necessary.

About facets

Facets can be seen as aspects, dimensions, viewpoints, perspectives, or thematics, generally clustering the considered domain into orthogonal groups. Classification using a faceted approach is not new: the theory of facets as been introduced around 1933 [23] by Ranganathan. According to the theoretical basis for the faceted analysis he laid, any document can be characterize by six facets, the first one corresponding to the domain it concerns and the five others being aspects of the reality: Personality, Matter, Energy, Space, and Time, known as the PMEST formula. This universal classification system is known as the *Colon System*. According to [16], *Personality* represents the main object of the domain (*what*), *Matter* is the substance or property of this object (*properties*), *Energy* is the action performed relatively to it (*how*), and *Space* (*where*) and *Time* (*when*) are complements allowing specifying a location in place and time. Ranganathan's Colon Classification provides very generic facets, which can be used almost with any objects to classify. The original Facet theory allows to easily characterize a phrase representing a document (*e.g.* the document's title, subject) according to the later general facets, since according to classical grammar, this phrase would be naturally constructed this way: <*subject* (P), *verb* (E), *manner complement or object* (M), *place complement* (S), *time complement* (T)] >. Others generic classification schemes have been since defined, like *e.g.* the well known second edition of the Bliss Bibliographic Classification [3], which contains thirteen facets. In particular, this scheme allows for more detailed classifications and is said to be sufficient in almost all domains or areas of knowledge [10].

The faceted approach is simply another way to classify things. This could be defined as a way to characterize things according to some of their specific properties, which are found to be important enough so that they can be said as fully characterizing those things in a given context. Facets are thus functions of two fundamental variables: the things or objects they will help to classify, which we will refer as *subjects*, and the context of this classification.

How to determine facets? : As explained in [19], facet analysis is also called analytico-synthetic classification, because it is derived from two processes: *analysis*, which consists in breaking down things to classify (called subjects) into elemental concepts, and *synthesis*, which consists in combining concepts to reform subjects. Indeed, according to [Mapl95] Raganathan proved that the later process can be applied to any subject and systematized. Basically, it consists in clustering the knowledge universe of concern into multiple subsets containing terms all related to a single abstract concept. Facets can be deduced from the analysis and synthesis of a set of documents taken from a corpus to be classified.

Generalized from an example in [22], taking its origins into a method called *Literary Warrant* [31] considering documents' titles, a process of building facets can be the following:

- Take a representative excerpt of corpus subjects, which we will call the training set;
- For each subject in the training set, build a profile of terms and/or phrases summarizing or representing it (this can be a title if available, or in case of textual documents, some words or phrases extracted from the document by some text analysis technique);
- Take each term or phrase in the profiles and group them into clusters, where every elements of a same cluster are related to a same general abstract concept;
- The facets are then named from the general concepts represented by each cluster.

But the original analytico-synthetic approach defined by Raganathan and illustrated there, can be seen as one way among others to determine facets. Instead of deducing facets from a set of domain data, they can be predetermined by an ad-hoc choice that seems relevant. This choice allows to avoid a tedious task and can reveal as effective.

Once facets have been defined, each subject can be represented by a vector in the facet space. Let $F_{FS,i}$ be one facet of a facet space FS of dimension n, a subject S can be represented by a profile vector $P_S = (f_{FS,1,S}, f_{FS,2,S}, ..., f_{FS,n,S})$, where $f_{FS,k,S}$, $k = 1, .., n$ is the term or value of the facet $F_{FS,k}$ matching the content of S.

Properties of facets and facets' values : This section discusses the characteristics of facets as well as the values in a facet, starting from the original facet theory and showing which level of flexibility can be worth considering.

In Raganathan's theory, facets can be seen as exclusive categories constituting an orthogonal basis of a multidimensional space in which subjects can be projected. Orthogonality between facets is in fact quite logical: each facet exists independently from the others, whatever its contents. A simple but very illustrative example is given in [Fmap06] for two facets. Also from the original theory, a subject can have assigned only one term or value per facet. The fact that terms of a facet are not simultaneously assignable could look restrictive at first glance when trying to define facets. But looking at the example in [Fmap06], we can derive the following rule that shows this should not be a restriction but should rather emerge from a good classification scheme : *if two terms can logically be combined when assigned to a resource, then they can not belong to a same facet*. Nevertheless, a definition of facet can be found in [Fmap06] that is less strict: "...the assignment of one heading to a resource limits the assignment of that resource of other headings in the set.". If we agree on that, we can imagine going further by creating rules formalizing kinds of limitations other than strict mutual exclusivity, or the possible combinations between terms intra- or inter- facets. This is an open research path that we have not explored yet.

Facets are often defined as a flat list of values or terms, having no relationship between each others except the fact that they are all possible values of a same facet. This is however not always so simple and there can be cases where it can be useful to define other kinds of facets and use relationships between a facet values. Even from the original facet's theory, the *Time* facet introduced a specific kind of facet having values defined not by a set of terms but by a set of numerical values. Generalizing, any *numeric facet* has values not necessarily discrete and finite, and will most often be displayed to the end-user in a form different from the one used for more classical flat term list facets: e.g. a simple or formatted text field in the former case, versus a choice list in the latter.

Then, values of a facet can be organized and have relationships between each others. We can derive two kinds of *hierarchical facets*. The first kind considers facets having values classified according to a hierarchy. In this case, these facets are often single-choice, but can be multi-choice when organized in sub-facets defining different dimensions. The second kind, considered in [Mah06], relates to facets having values that are paths in a hierarchy. Like

hierarchical facets of the first kind organized into sub-facets, facets of the second kind define somehow a sub facet space in the original facet space. In the latter case, an order relation is defined on the subspace dimensions and a value in a dimension depends from values of the preceding dimension. In this paper, we will not consider this kind of hierarchical facets. Finally, the literature shows that the relationships existing between the possible values of a facet ranges from none to hierarchal, up to ontological ones. For example, in [Fuji05], different ontologies are used to describe cultural digital resources from different facets. In this case, the content of each facet is a complete ontology.

Facets are not defined once for all, but must stay flexible and open to modifications: when new subjects are added to the corpus, new facets can emerge and must be taken into consideration. The facets and their related terms both can vary in form and quantity. The conclusions of [Hudo01] concerning nominative subdivisions lead to a more generic problem which is the static or dynamic nature of facets and rise the following questions: 1) should the term list or values of each facet be static and determined prior to any classification? and 2) should the facets and their number be fixed prior to any classification? It is obvious that a stable kernel of facets having determined content must be obtained first either from subjects analysis, either from expert domain knowledge. But once obtained this way, the classification scheme might evolve: occurrence of new facets because of new subjects linked to new categories, same thing for facets' terms or values. Stating this, facets having content dynamically defined like the ones that would correspond to nominative subdivisions ca be considered: e.g., a facet *Member*, containing the list of the members of an organization. In conclusion, we can consider that the list of terms or values of a facet is time dependent and in particular can be filled at runtime; that is what we will call *dynamic facets*. In [18], dynamic facets are considered and can in particular be remotely populated.

In [18], flat and hierarchical facets are considered, as well as static and dynamic ones. This means respectively that there can be a relationship between terms in a facet, and that facets terms can be determined at runtime. In [18], a facet is defined by a tuple <name, type, isHierarchical , rule>, where rule specifies how a facet term is assigned to a resource. In particular, dynamic facets allow for distant population (*i.e.* gathering the value from a distant service through a URI).

Facets were originally used to described what a document is about, referring then to its content. With the works that have been performed on the use of facets for browsing web resources, related meta-data becomes also faceted: when meta-data is provided by ontologies, this is related as *semantic faceted search* [26]. In the following, we will not make a distinction between normal facets and those related to meta-data. As soon as we use ontologies to describe the domain of our subjects as well as for annotations, we make the hypothesis that any concept used by facets should be present in the domain ontologies or in the annotation ones. As domain ontologies represent a formal description of what the subject is about, our facet schemes will indifferently contain normal or meta-data related ones.

The facet ontology that is exploited by BayFaC and presented in the following sections considers term-based or numeric facets, which can be dynamic. Moreover, it allows multiple values assignment per facet (although we have not use it so far). For the moment, hierarchical or full ontological facets have not been considered because there was no need to do it, but the facet ontology is flexible enough so as to allow these relationships with few modifications.

Our approach for Ontology-based Faceted Classification

Except for the particular case where facet classification is used to construct an ontology [22], which is not our scope, facets have been used for browsing semi-structured information defined by semantic web ontologies [17, 20, 15]. In this way, facets are exploited as a mean to provide a user-oriented view on the ontologies. The facet-based classification aspect has not

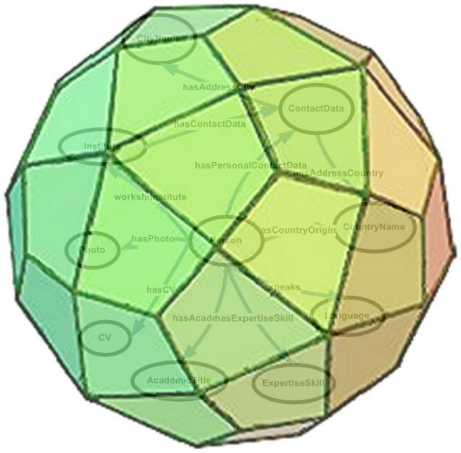


Figure 5.12: Facets to hide ontology complexity to end-users

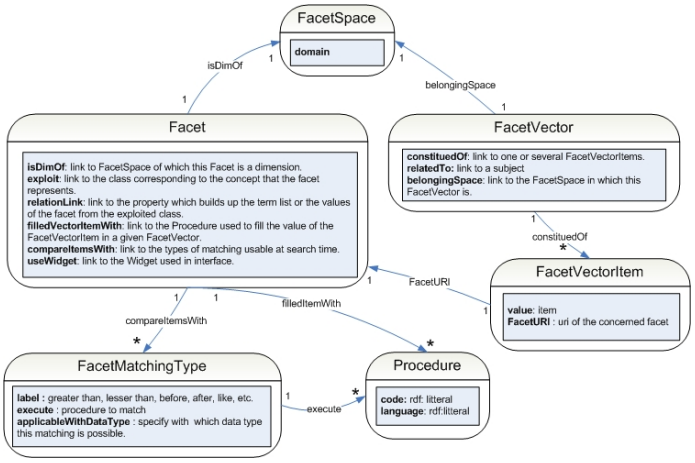


Figure 5.13: The Facet Ontology

been addressed so far with ontological information. Contrary to these approaches, we propose here to add a faceted classification layer on top of an ontology layer, to allow hiding as much as possible the complexity to the end-user. Figure 5.12 provides a good illustration of this. Exploiting this abstraction, BayFac let the end-user choose between intuitive faceted classification or ontology-based annotation, and then provides him full access to classified or annotated resources through faceted browsing.

Ontologies allow putting annotations on a resource. As the primary goal of an ontology is also before all to formalize the semantics of terms it contains, relations between those terms, used for annotations, can be very complex. Facets provide a simple and intuitive view on subjects, grouping them according to their properties that are found most relevant for or even by the end-user. When annotating a resource according to concepts defined in an ontology, the user is in direct contact with the inherent complexity of the ontology. Using a faceted classification is much more easier and allow a fast characterization of subjects. An important point to notice however, is that doing classification does not mean describing the object. For example, a description would probably contain some comments or human-readable description, which will never be used in facets as they are not useful for classifying (except if a text-analysis processing extracts relevant concepts and constructs facets with them).

The facet ontology : Figure 5.13 shows the facet ontology we have developed. One of the advantages of our approach is that the original ontology does not have to be changed to be faceted. We call *original ontology* the set of ontologies that are used to describe or annotate resources, on which a facet layer will be added. The facet ontology gives the possibility to make a facet from any concept of the original ontology. In the ontology we propose, the *Facet* class allows to create facets that points to classes in the original ontology, having a path to the subject's class of any complexity allowed by the original ontology. The class *Facet*, is explain in the next section, while the other main class, namely *FacetVector*, which provides a way to classify subjects, is described in the subsequent one.

Facet creation. : The first step in the definition of facets is to know what kind of resources we want to classify, *i.e.* what are the subjects. In the facet ontology, the kind of subject is determined thanks to the *FacetSpace* class and its *domain* property, the latter pointing to the subject class in the original ontology. *Facet* instances are dimensions of a *FacetSpace* (*isDimOf* property). The *exploit* property allows linking a facet to the class its values will be instances of. For example, if someone wants to be able to classify and search documents knowing their authors, a facet *Author* might be created. The facet space domain is a *Document* class. In the original ontology, the author of a document has for class *Person*. The facet *Author* will thus points to that class, through the *exploit* property. In order to be able to construct the possible values of a facet from its "exploited class", the relation linking the latter and the facet values must be specified. This modelled thanks by the *relationLink* property of *Facet*. Thus, the values of a facet are all resources in the original ontology having a relation of kind *relationLink* with the "exploited class". In most case, this relation is *rdf:type* when the list is made up of the instances of the exploited class, but it can be *rdf:subClassOf* for example if we are looking for a type of object (*e.g.* a specific kind of document). Finally, to facilitate the presentation of facets in user interfaces, we have added the *useWidget* property allowing affecting a widget with which users will access the facet's values.

Classifying and searching subjects. : To classify a subject, we introduce the concept of *FacetVector*, which is composed of *FacetVectorItems* linked to the vector via the property *constitutedOf*. These items are couple of *FacetURI* and value. Thanks to its *FacetVector*, a subject is classified in the *FacetSpace*. The link between a subject and its facet vector is represented by the *relatedTo* property. Thanks to these facet vectors, we somehow provide an inverted index of the subjects.

Different facets can be described using the same set of *relationLink* and *exploit*. Indeed, the matching between the item selected by the user and the one concerning the document could be exact, approximate, or according to a relation of order, etc... For example, in the facet "Author", we can offer to the user a list of all authors available. But if there are too many authors, we can choose to display a text box with which only part of the name is necessary. That's the goal of the *compareItemsWith* property, which links a *Facet* to a *FacetMatchingType*. For engineering of facets, the matching type provides an explicit *rdf:label* and a link (the *applicableWithDataType* property) to the type of data that is possible to match with. The matching type is also link to a *procedure*. It is constituted of a *code* used to do the matching and a *language* to know in which one this script is encoded. Whatever matching operators are allowed or not by the tool exploiting the facet ontology, request matching will always be performed comparing the subjects facet vectors with the facets' values chosen for the request. For the moment, search in BayFac is only performed on enumerative facets, on the basis of an exact matching.

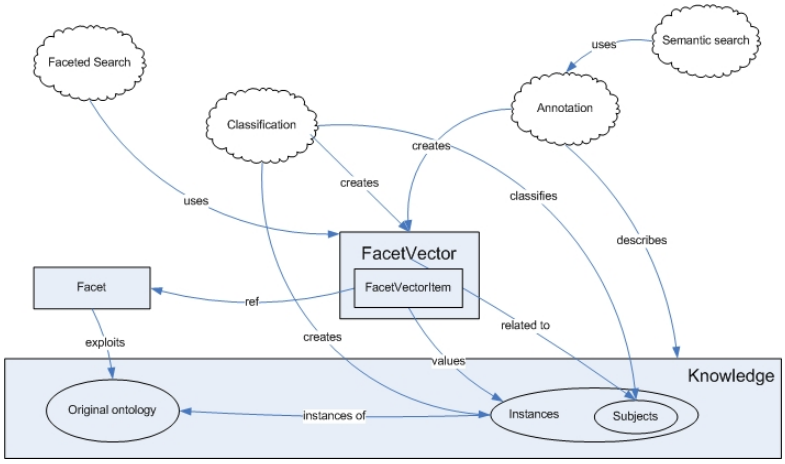


Figure 5.14: Federated approach to classification, annotation and search through facet vector indexation

Overview on classification, annotation and search : In our approach, faceted classification creates *FacetVector* instances for each subject that are categorized. As facets are built on top of the ontology that can be used for annotation, the process of annotating subjects allows also the background creation of their facet vectors. In the facet ontology, the *filledItemWith* property of *Facet* allows to specify a query that will be used to automatically construct a facet vector from the knowledge available in the original ontology with its instances. Then if for example an annotation service building external annotations is used on a subject before this one is handled by BayFac, the facet vector can be at least partially constructed as soon as the annotation service and BayFac work with the same ontology (referred as *Original Ontology* in Figure 6). Faceted search can thus be performed on the basis of faceted classification or annotations. Moreover, when a subject is classified according to facets, instances of the original ontology are created and will then be available as for any other annotations by meta-data related services. Figure 5.14 illustrates this kind of federated approach, which in the end offers the end-user the choice between annotations or classifications, while providing him the power of faceted search.

Bayesian automatic classification

Automatic classification will be based on Bayesian techniques, that will learn first from manual classification made by users, and then propose automatic categorizations once there is a sufficient amount of documents processed in a supervised manner to enable correct predictions. A Bayesian classifier is a probabilistic classifier based on a learning algorithm using the Bayes' theorem [2]. This theorem is used in statistical inference, in order to update probability estimations according to observations. In the case of a classifier, the probability of belonging to a given class of documents is evaluated for each new incoming document, using its characteristics, like for example the frequential analysis of the tokens it contains. Bayesian classification is notably used in spam filtering, like initially proposed by Paul Graham [14]. In this context, the efficiency of the Bayesian classifiers was demonstrated as being superior to manual classification [32]. NaŢve Bayes classifiers are known to be among the most efficient techniques to automatically classify documents. As a first implementation, we have used naŢve Bayes classifiers to perform classification predictions on enumerative facets, whose possible values are defined by a list of terms. Based on previous manual categorizations, the classifiers are able to propose facet's values for new documents entered in the system for being classified. In our context, Bayesian classifiers have been developed according to the

theory exposed hereafter.

Let F_{FS} be the set of facets of a facet space FS. If $F_{FS,i} \in F_{FS}$ is a facet of F_{FS}, the set of terms for $F_{FS,i}$ constitutes the base set from which a subject will be categorized for the facet F_{FS}. Let $CF_{FS,i}$ be this set, and $c_i \in CF_{FS,i}$ an element of this set. For a given subject s, the classification function $FC_i(s)$ represents the evidence for the fact that s belongs to c_i. In terms of probabilities, this could be expressed as:

$$FC_i(s) = P(c_i|\vec{s})$$

where $\vec{s}$ is the vectorial representation of subject s by a set of significative terms: $\vec{s} =< w_1, ...w_n >$. In the case of textual documents subjects, the vectorial representation is the set of significative words present in the document. Using the Bayes formula, we can write:

$$P(c_i|\vec{s}) = \frac{P(c_i)P(\vec{s}|c_i)}{P(\vec{s})}$$

The naŢve Bayes approach assumes the strong hypothesis of independence between the components of $\vec{s}$. Then the formula can be rewritten as:

$$P(c_i|\vec{s}) = \frac{P(c_i)\prod_{k=1}^{n} P(w_k|c_i)}{P(\vec{s})}$$

The best matching facet term $c_i^*(s)$ for the subject s can be found by founding the higher probability $P(c_i|s)$. As $P(\vec{s})$ is a constant for all c_i, the following formula can finally be used:

$$c_i^*(s) = \underset{c_i \in CF_{FS,i}}{argMax}\{P(c_i)\prod_{k=1}^{n} P(w_k|c_i)\}$$

First classifications obtained in the context of *Form@Hetice* and also for another project have shown promising results. However, even if results seemed to be good in average, we can not be sure until tests with a sufficient documents mass have not been conducted. We also plan to introduce certainty coefficient, so that to have an idea of when a prediction is trustable or not.

Previous work

Facets have already been used in conjunction with ontologies in the past. For example, [1] relates the build of a web interface using facets to access the Dublin Core 2003 conference's meta-data expressed in RDF/S. This is a case of faceted browsing / search based on meta-data. In [22], Prieto-Diaz proposes a method to construct rough (not completely formal) ontologies using faceted classification as a bottom-up approach to refine and validate a postulated high-level ontology obtained from an analysis of the domain (top-down approach). This method is based on a Domain Analysis approach. It is a quite logical approach that could be useful in many cases to enhance and validate an ontology empirically, based on the common use performed on knowledge (information) that will be described with that ontology. Going further in the automatization, [9] proposes automatic and scalable methods for the construction of faceted hierarchies. Finally, the concept of faceted ontology seems to have been introduced in 2001 by Tzitzikas *et al.* [29, 30]. Formally, a faceted ontology is defined as a set of ontologies, the later being here reduced to be a terminology, associated to a subsumption relation over it. In this approach, the concept of ontology is greatly reduced, and it seems that they simply provided another name for classical faceted classification schemes. Other articles from them reflect that they later changed the term into *taxonomy* [30]. However, they introduce in e.g. [30] an interesting approach that consists in formally specifying relations

between facet items that are valid and those that are invalid. Allowing thus to avoid non possible configurations, both during the indexation and browsing processes.

There exists an XML vocabulary that appears to be a de-facto standard for the definition of facets, aiming at providing a way to exchange hierarchical faceted metadata: XFML, the eXchangeable Faceted Metadata Language [11]. It provides concepts for defining facets, indexing web pages, also specifying mappings between facet sets, and putting belief values on indexing. Basically, XFML is inspired by Topicmaps. The map concept allows defining a set of facets; then, *facet* defines orthogonal containers containing hierarchies of *topics*; the *connect* concepts allows specifying equivalences between topics in different maps; and *occurrence* let you index a document, while specifying with the *strength* attribute the confidence degree you have in this indexing. The specification contains other concepts that we will not describe here; see [11]. Compared to XFML, our approach is slightly more flexible. In particular, we have not limited the content of facets to be hierarchies of topics. Instead, topics are related to their belonging facet by any kind of relation that is specified in the facet attributes. Moreover, a facet is not limited to contain terms but can also be filled with numeric values. Finally, the content of facets can be dynamically build, which is not possible using XFML. The advantage of having an ontology in the background of a facet set is that any facet can be defined according to any relation or sequence of relations between concepts defined in the ontology.

An approach similar to what we proposed with BayFac can be found in [21]. The focus is given to the use of metadata, which are determined and organized in facets groups according to the specific goal of the mining task, and then assigned to resources by classifiers (Bayesian or else). Additionally to automatically assign metadata facets terms to documents, they derive rules of combination between concepts in different facets. Then, inference conducted on classified documents will generate new metadata.

Additionally to the tool they present, *FacetMap*[8], [25] list some tools that use facets or similar approaches to facilitate data or metadata classification and retrieval. Among them, *Flamenco*[9] [Yee03], *Phlat*[10] [Cut06]. Most of these systems have shown that the faceted approach suits more users than using searches based on a single categorization and moreover provide ways to enrich the user experience by providing means to dynamically construct queries in a comprehensive way. Another example of facet-based browser is *FastAxon* [28], allowing the design of faceted taxonomies and navigation among indexed documents.

Ontogator is a tool having emerged from a recent initiative in Finland. The underlying work is probably the closest to our approach. In particular, the authors have highlighted the fact that classical faceted classification lacks exploiting the relations existing between objects [17]. This is their main argument to combine faceted search with ontologies. They have also argued for the use of facets for facilitating browsing and furthermore have proposed the separation of user-oriented facets from the ontologies used for annotations [Suomi06]. Like in our view, ontologies stay at a technical level, while facets are dedicated to the end-users, done for them or made by them independently from ontologies. A bridge method is then used to link facets to ontology concepts. The approach taken for Ontogator considers orthogonal hierarchical facets, constructed from an annotation ontology, like we do. The hierarchy rules allow to construct facets by making projections of the domain ontology on chosen relations (e.g. subClassOf). Like for our approach with the *FacetSpace* domain and the *exploit* and *relationLink* properties of the *Facet* class, facets can be automatically populated from the ontology. However, using a query defining the relationship between the root class of a facet and the facet values (like we do) is closer to the spirit of facet classification and provides directly user-centric facets. As written in [17], an inconvenient that we do not have

[8] `http://research.microsoft.com/vibe/projects/FacetMap.aspx`, not to be confounded with another Facetmap, at http://facetmap.com, which is a software package dedicated to faceted classification.

[9] `http://flamenco.berkeley.edu/`

[10] `http://research.microsoft.com/adapt/phlat/`

is that projections using hierarchy rules in Ontogator can provide facets values that are not relevant for the user and must be filtered in the user interface.

Semantic faceted search concerns facets on metadata (*i.e.* annotations). In [26], ontologies are first used to annotate documents. Then, faceted ontologies are constructed by taking each concept used in the annotation as the root class for a facet, which is then filled with a hierarchy of possible values that seems to be the classes specified as the range of the root class. Finally, end-user facets are also hierarchies of terms, but constructed independently from the annotation ontologies and mapped to the faceted ontologies to facilitate searching. As for our proposal, the approach in [26] makes a clear separation between the user facets and the annotations, and moreover allows to define different sets of facets regarding the audience.

In [20], facets are considered from a browsing perspective on ontology-based information and assimilated as properties of the class representing the subjects. This way facets can automatically be constructed from any RDFS dataset and the prototype interface allows an expressivity in the visually build queries that goes beyond the other existing faceted browsers. In particular, *existential selection* and *join selection* operators are introduced, which allow for non-standard queries searching a strict equality between a chosen facet value and a subject property. In our current work, the *compareItemsWith* property of the class *Facet* allows to define an existential selection, but goes far further as it provides enough flexibility to define any kind of matching operation that can be performed on a given facet. The join selection is not possible in our system, butit should be implementable. Finally, as each facet designed with our ontology is filled by a query defining the path from the facet space domain class (defining subjects) to its root (exploited) class, our approach allows also the so-called inverse selections highlighted in [20].

5.2.3 Use by a Palette CoP

Example of use

Two or three members of the CoP decide to identify a few appropriate facets by examining the already existing documents on the website. They are chosen thinking about what kind of search could be led in the future. Once this work done, a web interface allows any member of the CoP to create and manage facets. Then, all CoP members will be able to categorize old or new documents and to retrieve them easily.

When a CoP member wants to add a new document, he uploads it on the web site. The engine behind the site enables him to manually categorize his document according to the various facets which were defined. At the same time, the BayFaC service records this action and learns from that. In the same period of time, other members of the CoP, upload their own documents, or classify old documents. And so on and so forth, for months.

The BayFaC service will work step by step: at first, it will need to be manually "fed". Then, it will begin to develop a certain ability to guess how documents should be categorized, cautiously asking the user to confirm the validity of his guess. As time goes on, it will make less and less errors. Once documents are classified, the BayFaC service gives the possibility to search documents by browsing through categories determined by the combination of chosen facet values. For instance, search all documents from the *Author* facet where the author is *Peter*, filtered using the *Date* facet to specify that the date should be after the 8th of March.

Application with Form@hetice

We have designed a small ontology allowing documents classification in the scope of the *Form@Hetice* CoP (Figure 5.15). Currently, facets (in green), have been defined according to the input of a member of the CoP and analysis of the current documents stored in the CoP's web site.

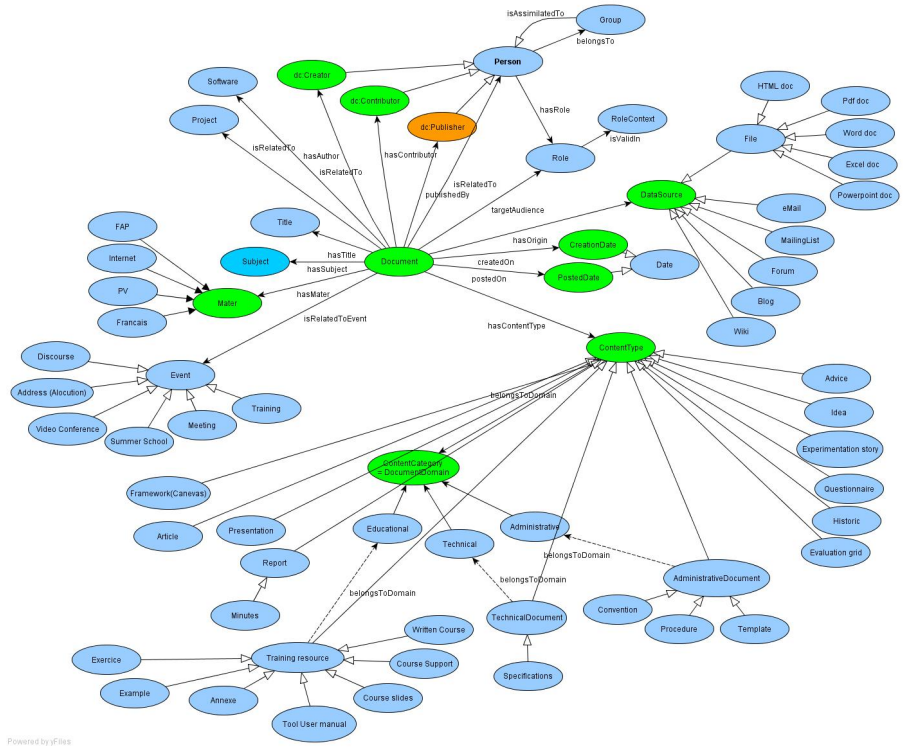


Figure 5.15: Documents ontology for Form@Hetice

Here is an example of how looks the "ContentType" facet in RDFS:

```
<fac:Facet rdf:about="http://www.tudor.lu/Ontologie/FormaHetice#ContentTypeFacet">
 <rdfs:label xml:lang="EN"><![CDATA[Type de contenu]]></rdfs:label>
 <fac:isDimOf resource="http://www.tudor.lu/Ontologie/FormaHetice#FormaHeticeFacetSpace "/>
 <fac:exploit resource="http://www.tudor.lu/Ontologie/FormaHetice#ContentType"/>
 <fac:relationLink resource="http://www.w3.org/2000/01/rdf-schema#subClassOf"/>
 <fac:filledVectorItemWith resource="http://www.tudor.lu/Ontologie/FormaHetice#QueryContentType"/
 <fac:compareItemsWith resource="#118061032957204"/>
 <fac:useWidget resource="http://www.tao.lu/datatypes/WidgetDefinitions.rdf#ComboBox"/>
</fac:Facet>

<fac:FacetSpace rdf:about="http://www.tudor.lu/Ontologie/FormaHetice#FormaHeticeFacetSpace">
 <facă:domain resource="http://www.tudor.lu/Ontologie/FormaHetice#Document"/>
</facă:FacetSpace>

<fac:Procedure rdf:about="http://www.tudor.lu/Ontologie/FormaHetice#QueryContentType">
 <fac:hasCode xml:lang="EN">
 <![CDATA[Select ?item Where {
    $idInstance
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ?item          }]]>
 </fac:hasCode>
<fac:hasLanguage xml:lang="EN"><![CDATA[ SPARQL]]></fac:hasLanguage>
</ns5:FacetQuery>
```

5.2.4 Further work

BayFac will be enhanced at the level of ergonomy and usability according to the feedbacks from tests conducted with the *Form@Hetice* CoP. In this context, the used ontology still needs

to be enhanced and linked to the O'CoP ontology. Concerning the functions provided by BayFac, further development needs to be done, in particular for managing non-enumerative facets, as well as including comparison operators access, and allowing automatic upload of documents posted in the CoP's portal. The automatic classification part will still have to be enhanced and tested. In particular enhancements will be investigated for providing trust values on predictions. Scaling tests will be required to know if the naïve Bayes approach is sufficient or if a more complex algorithm needs to be used. The use of other kinds of classifiers will also be investigated for non enumerative facets.

Chapter 6

Conclusion

This deliverable proposed a detailed description of the basic KM services to be used in the context of Palette, and relying either on Corese/SeWeSe or on Generis:

- Ontology management: to create, modify, or remove ontologies. And to add, remove or modify concepts or properties.
- Annotation management: to add, remove, or modify annotations of CoP resources.
- Retrieval: to search the resources of the CoP.

The basic services, as web-services, use both SOAP and REST technologies in order to offer several possibilities of integration with other Palette tools and services. In the majority of cases, these basic services will not be used directly by the CoP members, but will serve as building blocks for more complex services dedicated to the CoPs. We showed an example of this combination of services to build the complex service of ontology creation, ECCO.

In addition, we presented some of the complex services dedicated to the CoPs:

- The first one, **Sweetwiki**, is a semantic wiki that provide CoPs with a platform for collaborative work, enhanced with Knowledge management services, like semantic search, or semantic guided navigation or awareness. In this presentation, we focused on the use of Sweetwiki by Palette CoPs and we specified the future developments that had been suggested after a joint work with the CoPs in the context of the Participatory Design Methodology of Palette.
- The second complex service, **BayFac** is a semi-automatic classification service, based on faceted classification and Bayesian approach to classification. BayFac aims at providing CoPs with a platform to classify and search their resources.

As further work, we will:

- Continue the development of the presented complex services, in addition to some new complex services.
- Work with other Palette partners to achieve the interoperability of KM services with other Palette services and tools.
- Provide the CoPs with these KM services, in order to test and enhance them according to the feedbacks of users.
- Try to enhance scalability and optimization of the basic KM services.

Appendix A

WSDLs of Corese/SeWeSe services

A.1 Administration service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="http://admin.semservices.inria.fr/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        name="AdminServiceImplService"
        targetNamespace="http://admin.semservices.inria.fr/">
  <wsdl:types>
<xsd:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
            targetNamespace="http://admin.semservices.inria.fr/">
<xsd:element name="InvalidDataException" type="tns:InvalidDataException"/>
<xsd:complexType name="InvalidDataException">
<xsd:sequence>
<xsd:element name="invalidData" nillable="true" type="xsd:string"/>
<xsd:element name="invalidURI" nillable="true" type="xsd:boolean"/>
<xsd:element name="malformedData" nillable="true" type="xsd:boolean"/>
<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="EngineNotFoundException" type="tns:EngineNotFoundException"/>
<xsd:complexType name="EngineNotFoundException">
<xsd:sequence>
<xsd:element name="invalidEngineID" nillable="true" type="xsd:long"/>

<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="isValidEngineID" type="tns:isValidEngineID"/>
<xsd:complexType name="isValidEngineID">
<xsd:sequence>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="isValidEngineIDResponse" type="tns:isValidEngineIDResponse"/>
<xsd:complexType name="isValidEngineIDResponse">
<xsd:sequence>
<xsd:element name="return" type="xsd:boolean"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="customInit" type="tns:customInit"/>
<xsd:complexType name="customInit">

<xsd:sequence>
<xsd:element minOccurs="0" name="ontopath" type="xsd:string"/>
<xsd:element minOccurs="0" name="annotpath" type="xsd:string"/>
<xsd:element minOccurs="0" name="rulepath" type="xsd:string"/>
<xsd:element minOccurs="0" name="projection-max" type="xsd:int"/>
<xsd:element minOccurs="0" name="result-max" type="xsd:int"/>
<xsd:element minOccurs="0" name="result-join" type="xsd:boolean"/>
</xsd:sequence>
```

```
</xsd:complexType>
<xsd:element name="customInitResponse" type="tns:customInitResponse"/>
<xsd:complexType name="customInitResponse">
<xsd:sequence>
<xsd:element name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="reset" type="tns:reset"/>
<xsd:complexType name="reset">

<xsd:sequence>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="load" type="tns:load"/>
<xsd:complexType name="load">
<xsd:sequence>
<xsd:element minOccurs="0" name="new-data" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="init" type="tns:init"/>
<xsd:complexType name="init">
<xsd:sequence>
<xsd:element minOccurs="0" name="ontopath" type="xsd:string"/>
<xsd:element minOccurs="0" name="annotpath" type="xsd:string"/>
<xsd:element minOccurs="0" name="rulepath" type="xsd:string"/>

</xsd:sequence>
</xsd:complexType>
<xsd:element name="initResponse" type="tns:initResponse"/>
<xsd:complexType name="initResponse">
<xsd:sequence>
<xsd:element name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="unload" type="tns:unload"/>
<xsd:complexType name="unload">
<xsd:sequence>
<xsd:element minOccurs="0" name="data" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
  </wsdl:types>

  <wsdl:message name="customInitResponse">
    <wsdl:part element="tns:customInitResponse" name="result">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="reset">
    <wsdl:part element="tns:reset" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="init">

    <wsdl:part element="tns:init" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="initResponse">
    <wsdl:part element="tns:initResponse" name="result">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="unload">
    <wsdl:part element="tns:unload" name="parameters">

    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="isValidEngineID">
    <wsdl:part element="tns:isValidEngineID" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="customInit">
```

```
    <wsdl:part element="tns:customInit" name="parameters">
    </wsdl:part>

  </wsdl:message>
  <wsdl:message name="load">
    <wsdl:part element="tns:load" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="EngineNotFoundException">
    <wsdl:part element="tns:EngineNotFoundException" name="EngineNotFoundException">
    </wsdl:part>
  </wsdl:message>

  <wsdl:message name="isValidEngineIDResponse">
    <wsdl:part element="tns:isValidEngineIDResponse" name="result">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="InvalidDataException">
    <wsdl:part element="tns:InvalidDataException" name="InvalidDataException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="AdminService">

    <wsdl:operation name="isValidEngineID">
      <wsdl:input message="tns:isValidEngineID" name="isValidEngineID">
    </wsdl:input>
      <wsdl:output message="tns:isValidEngineIDResponse" name="isValidEngineIDResponse">
    </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="customInit">
      <wsdl:input message="tns:customInit" name="customInit">
    </wsdl:input>

      <wsdl:output message="tns:customInitResponse" name="customInitResponse">
    </wsdl:output>
      <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="reset">
      <wsdl:input message="tns:reset" name="reset">
    </wsdl:input>
      <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">

    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="load">
      <wsdl:input message="tns:load" name="load">
    </wsdl:input>
      <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
    </wsdl:fault>
      <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
    </wsdl:fault>

    </wsdl:operation>
    <wsdl:operation name="init">
      <wsdl:input message="tns:init" name="init">
    </wsdl:input>
      <wsdl:output message="tns:initResponse" name="initResponse">
    </wsdl:output>
      <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
    </wsdl:fault>
    </wsdl:operation>

    <wsdl:operation name="unload">
      <wsdl:input message="tns:unload" name="unload">
    </wsdl:input>
      <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
    </wsdl:fault>
      <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
```

```
<wsdl:binding name="AdminServiceImplServiceSoapBinding" type="tns:AdminService">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="isValidEngineID">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="isValidEngineID">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="isValidEngineIDResponse">
      <soap:body use="literal"/>

    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="customInit">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="customInit">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="customInitResponse">
      <soap:body use="literal"/>

    </wsdl:output>
    <wsdl:fault name="InvalidDataException">
      <soap:fault name="InvalidDataException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="load">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="load">
      <soap:body use="literal"/>

    </wsdl:input>
    <wsdl:fault name="InvalidDataException">
      <soap:fault name="InvalidDataException" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="reset">

    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="reset">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="init">

    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="init">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="initResponse">
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="InvalidDataException">
      <soap:fault name="InvalidDataException" use="literal"/>

    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="unload">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="unload">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:fault name="InvalidDataException">
      <soap:fault name="InvalidDataException" use="literal"/>

    </wsdl:fault>
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>
```

```
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="AdminServiceImplService">
    <wsdl:port binding="tns:AdminServiceImplServiceSoapBinding" name="AdminServiceImplPort">
      <soap:address location="http://argentera.inria.fr/semservices/Admin"/>

    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

A.2 Ontology management service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="http://edit.semservices.inria.fr/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        name="OntologyEditionServiceImplService"
        targetNamespace="http://edit.semservices.inria.fr/">
  <wsdl:types>
<xs:schema xmlns="http://jaxb.dev.java.net/array"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           attributeFormDefault="unqualified"
           elementFormDefault="unqualified"
           targetNamespace="http://jaxb.dev.java.net/array">
<xs:complexType final="#all" name="stringArray">
<xs:sequence>
<xs:element maxOccurs="unbounded" minOccurs="0" name="item" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
<xsd:schema xmlns:ns0="http://jaxb.dev.java.net/array"
            attributeFormDefault="unqualified"
            elementFormDefault="qualified"
            targetNamespace="http://edit.semservices.inria.fr/">
<xsd:element name="InvalidDataException" type="tns:InvalidDataException"/>
<xsd:complexType name="InvalidDataException">
<xsd:sequence>
<xsd:element name="invalidData" nillable="true" type="xsd:string"/>
<xsd:element name="invalidURI" nillable="true" type="xsd:boolean"/>
<xsd:element name="malformedData" nillable="true" type="xsd:boolean"/>

<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="ConceptOrPropertyNotFoundException"
             type="tns:ConceptOrPropertyNotFoundException"/>
<xsd:complexType name="ConceptOrPropertyNotFoundException">
<xsd:sequence>
<xsd:element name="notionURI" nillable="true" type="xsd:string"/>
<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="ConceptOrPropertyAlreadyAvailableException"
             type="tns:ConceptOrPropertyAlreadyAvailableException"/>
<xsd:complexType name="ConceptOrPropertyAlreadyAvailableException">
<xsd:sequence>
<xsd:element name="uri" nillable="true" type="xsd:string"/>
<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>

<xsd:element name="EngineNotFoundException" type="tns:EngineNotFoundException"/>
<xsd:complexType name="EngineNotFoundException">
<xsd:sequence>
<xsd:element name="invalidEngineID" nillable="true" type="xsd:long"/>
<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
```

```
</xsd:complexType>
<xsd:element name="removeDomain" type="tns:removeDomain"/>
<xsd:complexType name="removeDomain">
<xsd:sequence>
<xsd:element minOccurs="0" name="domain" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="addProperty" type="tns:addProperty"/>
<xsd:complexType name="addProperty">

<xsd:sequence>
<xsd:element minOccurs="0" name="property" type="xsd:string"/>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="labels" type="ns0:stringArray"/>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="comments" type="ns0:stringArray"/>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="parents" type="xsd:string"/>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="domains" type="xsd:string"/>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="ranges" type="xsd:string"/>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="defined-by" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="addDefinedBy" type="tns:addDefinedBy"/>
<xsd:complexType name="addDefinedBy">
<xsd:sequence>
<xsd:element minOccurs="0" name="defined-by" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>

</xsd:sequence>
</xsd:complexType>
<xsd:element name="removeDefinedBy" type="tns:removeDefinedBy"/>
<xsd:complexType name="removeDefinedBy">
<xsd:sequence>
<xsd:element minOccurs="0" name="defined-by" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="removeRange" type="tns:removeRange"/>
<xsd:complexType name="removeRange">
<xsd:sequence>
<xsd:element minOccurs="0" name="range" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>

</xsd:complexType>
<xsd:element name="modifyLabel" type="tns:modifyLabel"/>
<xsd:complexType name="modifyLabel">
<xsd:sequence>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
<xsd:element minOccurs="0" name="new-label" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="modifyVersion" type="tns:modifyVersion"/>
<xsd:complexType name="modifyVersion">
<xsd:sequence>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
<xsd:element minOccurs="0" name="new-version" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="removeLabel" type="tns:removeLabel"/>
<xsd:complexType name="removeLabel">

<xsd:sequence>
<xsd:element minOccurs="0" name="label" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="removeConcept" type="tns:removeConcept"/>
```

```
<xsd:complexType name="removeConcept">
<xsd:sequence>
<xsd:element minOccurs="0" name="concept" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
<xsd:element minOccurs="0" name="update-data" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="substitute" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="addParent" type="tns:addParent"/>

<xsd:complexType name="addParent">
<xsd:sequence>
<xsd:element minOccurs="0" name="parent" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="modifyComment" type="tns:modifyComment"/>
<xsd:complexType name="modifyComment">
<xsd:sequence>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
<xsd:element minOccurs="0" name="new-comment" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="addRange" type="tns:addRange"/>
<xsd:complexType name="addRange">
<xsd:sequence>

<xsd:element minOccurs="0" name="range" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="remove" type="tns:remove"/>
<xsd:complexType name="remove">
<xsd:sequence>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
<xsd:element minOccurs="0" name="update" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="removeResponse" type="tns:removeResponse"/>
<xsd:complexType name="removeResponse">
<xsd:sequence>
<xsd:element name="return" type="xsd:boolean"/>

</xsd:sequence>
</xsd:complexType>
<xsd:element name="removeParent" type="tns:removeParent"/>
<xsd:complexType name="removeParent">
<xsd:sequence>
<xsd:element minOccurs="0" name="parent" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="addConcept" type="tns:addConcept"/>
<xsd:complexType name="addConcept">
<xsd:sequence>
<xsd:element minOccurs="0" name="concept" type="xsd:string"/>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="labels" type="ns0:stringArray"/>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="comments" type="ns0:stringArray"/>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="parents" type="xsd:string"/>

<xsd:element maxOccurs="unbounded" minOccurs="0" name="defined-by" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="create" type="tns:create"/>
<xsd:complexType name="create">
<xsd:sequence>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
```

```
<xsd:element minOccurs="0" name="xml-base" type="xsd:string"/>
<xsd:element minOccurs="0" name="cos-graph" type="xsd:string"/>
<xsd:element minOccurs="0" name="label" type="xsd:string"/>
<xsd:element minOccurs="0" name="comment" type="xsd:string"/>
<xsd:element minOccurs="0" name="version" type="xsd:string"/>
<xsd:element minOccurs="0" name="override" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="update" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>

</xsd:complexType>
<xsd:element name="createResponse" type="tns:createResponse"/>
<xsd:complexType name="createResponse">
<xsd:sequence>
<xsd:element name="return" type="xsd:boolean"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="removeComment" type="tns:removeComment"/>
<xsd:complexType name="removeComment">
<xsd:sequence>
<xsd:element minOccurs="0" name="comment" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="addDomain" type="tns:addDomain"/>
<xsd:complexType name="addDomain">

<xsd:sequence>
<xsd:element minOccurs="0" name="domain" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="removeProperty" type="tns:removeProperty"/>
<xsd:complexType name="removeProperty">
<xsd:sequence>
<xsd:element minOccurs="0" name="property" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
<xsd:element minOccurs="0" name="update-data" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="substitue" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="addLabel" type="tns:addLabel"/>

<xsd:complexType name="addLabel">
<xsd:sequence>
<xsd:element minOccurs="0" name="label" type="xsd:string"/>
<xsd:element minOccurs="0" name="lang" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="addComment" type="tns:addComment"/>
<xsd:complexType name="addComment">
<xsd:sequence>
<xsd:element minOccurs="0" name="comment" type="xsd:string"/>
<xsd:element minOccurs="0" name="lang" type="xsd:string"/>
<xsd:element minOccurs="0" name="notion" type="xsd:string"/>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>

</xsd:schema>
  </wsdl:types>
  <wsdl:message name="addLabel">
    <wsdl:part element="tns:addLabel" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="removeDomain">
    <wsdl:part element="tns:removeDomain" name="parameters">
    </wsdl:part>
```

```
</wsdl:message>
<wsdl:message name="addProperty">
  <wsdl:part element="tns:addProperty" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="addComment">
  <wsdl:part element="tns:addComment" name="parameters">
  </wsdl:part>
</wsdl:message>

<wsdl:message name="removeLabel">
  <wsdl:part element="tns:removeLabel" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="removeDefinedBy">
  <wsdl:part element="tns:removeDefinedBy" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="modifyVersion">

  <wsdl:part element="tns:modifyVersion" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="ConceptOrPropertyAlreadyAvailableException">
  <wsdl:part element="tns:ConceptOrPropertyAlreadyAvailableException"
             name="ConceptOrPropertyAlreadyAvailableException">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="createResponse">
  <wsdl:part element="tns:createResponse" name="result">

  </wsdl:part>
</wsdl:message>
<wsdl:message name="create">
  <wsdl:part element="tns:create" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="addParent">
  <wsdl:part element="tns:addParent" name="parameters">
  </wsdl:part>

</wsdl:message>
<wsdl:message name="modifyLabel">
  <wsdl:part element="tns:modifyLabel" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="removeConcept">
  <wsdl:part element="tns:removeConcept" name="parameters">
  </wsdl:part>
</wsdl:message>

<wsdl:message name="remove">
  <wsdl:part element="tns:remove" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="ConceptOrPropertyNotFoundException">
  <wsdl:part element="tns:ConceptOrPropertyNotFoundException"
             name="ConceptOrPropertyNotFoundException">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="addConcept">

  <wsdl:part element="tns:addConcept" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="removeRange">
  <wsdl:part element="tns:removeRange" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="removeComment">
  <wsdl:part element="tns:removeComment" name="parameters">
```

```
  </wsdl:part>
</wsdl:message>
<wsdl:message name="EngineNotFoundException">
  <wsdl:part element="tns:EngineNotFoundException"
             name="EngineNotFoundException">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="removeParent">
  <wsdl:part element="tns:removeParent" name="parameters">
  </wsdl:part>

</wsdl:message>
<wsdl:message name="addDefinedBy">
  <wsdl:part element="tns:addDefinedBy" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="InvalidDataException">
  <wsdl:part element="tns:InvalidDataException" name="InvalidDataException">
  </wsdl:part>
</wsdl:message>

<wsdl:message name="removeProperty">
  <wsdl:part element="tns:removeProperty" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="addDomain">
  <wsdl:part element="tns:addDomain" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="modifyComment">

  <wsdl:part element="tns:modifyComment" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="addRange">
  <wsdl:part element="tns:addRange" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="removeResponse">
  <wsdl:part element="tns:removeResponse" name="result">

  </wsdl:part>
</wsdl:message>
<wsdl:portType name="OntologyEditionService">
  <wsdl:operation name="removeDomain">
    <wsdl:input message="tns:removeDomain" name="removeDomain">
  </wsdl:input>
    <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
  </wsdl:fault>
    <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
                name="ConceptOrPropertyNotFoundException">

  </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="addProperty">
    <wsdl:input message="tns:addProperty" name="addProperty">
  </wsdl:input>
    <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
  </wsdl:fault>
    <wsdl:fault message="tns:ConceptOrPropertyAlreadyAvailableException"
                name="ConceptOrPropertyAlreadyAvailableException">
  </wsdl:fault>

  </wsdl:operation>
  <wsdl:operation name="addDefinedBy">
    <wsdl:input message="tns:addDefinedBy" name="addDefinedBy">
  </wsdl:input>
    <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
  </wsdl:fault>
    <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
                name="ConceptOrPropertyNotFoundException">
  </wsdl:fault>
  </wsdl:operation>
```

```
<wsdl:operation name="removeDefinedBy">
  <wsdl:input message="tns:removeDefinedBy" name="removeDefinedBy">
</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="removeRange">

  <wsdl:input message="tns:removeRange" name="removeRange">
</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="modifyLabel">
  <wsdl:input message="tns:modifyLabel" name="modifyLabel">

</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="modifyVersion">
  <wsdl:input message="tns:modifyVersion" name="modifyVersion">
</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>

</wsdl:operation>
<wsdl:operation name="removeLabel">
  <wsdl:input message="tns:removeLabel" name="removeLabel">
</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
</wsdl:fault>
</wsdl:operation>

<wsdl:operation name="removeConcept">
  <wsdl:input message="tns:removeConcept" name="removeConcept">
</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
</wsdl:fault>
  <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
</wsdl:fault>

</wsdl:operation>
<wsdl:operation name="addParent">
  <wsdl:input message="tns:addParent" name="addParent">
</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
</wsdl:fault>
</wsdl:operation>

<wsdl:operation name="modifyComment">
  <wsdl:input message="tns:modifyComment" name="modifyComment">
</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="addRange">
```

```
  <wsdl:input message="tns:addRange" name="addRange">
</wsdl:input>

  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="remove">
  <wsdl:input message="tns:remove" name="remove">
</wsdl:input>
  <wsdl:output message="tns:removeResponse" name="removeResponse">

</wsdl:output>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
  <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="removeParent">
  <wsdl:input message="tns:removeParent" name="removeParent">
</wsdl:input>

  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="addConcept">
  <wsdl:input message="tns:addConcept" name="addConcept">
</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">

</wsdl:fault>
  <wsdl:fault message="tns:ConceptOrPropertyAlreadyAvailableException"
              name="ConceptOrPropertyAlreadyAvailableException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="create">
  <wsdl:input message="tns:create" name="create">
</wsdl:input>
  <wsdl:output message="tns:createResponse" name="createResponse">
</wsdl:output>

  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>
  <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="removeComment">
  <wsdl:input message="tns:removeComment" name="removeComment">
</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">

</wsdl:fault>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="addDomain">
  <wsdl:input message="tns:addDomain" name="addDomain">
</wsdl:input>
  <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
</wsdl:fault>

  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="removeProperty">
  <wsdl:input message="tns:removeProperty" name="removeProperty">
```

```
    </wsdl:input>
      <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
    </wsdl:fault>
      <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
                  name="ConceptOrPropertyNotFoundException">

    </wsdl:fault>
      <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="addLabel">
      <wsdl:input message="tns:addLabel" name="addLabel">
    </wsdl:input>
      <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
    </wsdl:fault>

      <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
                  name="ConceptOrPropertyNotFoundException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="addComment">
      <wsdl:input message="tns:addComment" name="addComment">
    </wsdl:input>
      <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
    </wsdl:fault>
      <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
                  name="ConceptOrPropertyNotFoundException">

    </wsdl:fault>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="OntologyEditionServiceImplServiceSoapBinding"
                type="tns:OntologyEditionService">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="removeDomain">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="removeDomain">
        <soap:body use="literal"/>

      </wsdl:input>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConceptOrPropertyNotFoundException">
        <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="addProperty">

      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="addProperty">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConceptOrPropertyAlreadyAvailableException">
        <soap:fault name="ConceptOrPropertyAlreadyAvailableException" use="literal"/>

      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="addDefinedBy">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="addDefinedBy">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>

      </wsdl:fault>
      <wsdl:fault name="ConceptOrPropertyNotFoundException">
        <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
      </wsdl:fault>
```

```
</wsdl:operation>
<wsdl:operation name="removeDefinedBy">
  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="removeDefinedBy">
    <soap:body use="literal"/>

  </wsdl:input>
  <wsdl:fault name="InvalidDataException">
    <soap:fault name="InvalidDataException" use="literal"/>
  </wsdl:fault>
  <wsdl:fault name="ConceptOrPropertyNotFoundException">
    <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="removeRange">

  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="removeRange">
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:fault name="InvalidDataException">
    <soap:fault name="InvalidDataException" use="literal"/>
  </wsdl:fault>
  <wsdl:fault name="ConceptOrPropertyNotFoundException">
    <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>

  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="modifyLabel">
  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="modifyLabel">
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:fault name="InvalidDataException">
    <soap:fault name="InvalidDataException" use="literal"/>

  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="modifyVersion">
  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="modifyVersion">
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:fault name="InvalidDataException">
    <soap:fault name="InvalidDataException" use="literal"/>

  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="removeLabel">
  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="removeLabel">
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:fault name="InvalidDataException">
    <soap:fault name="InvalidDataException" use="literal"/>

  </wsdl:fault>
  <wsdl:fault name="ConceptOrPropertyNotFoundException">
    <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="removeConcept">
  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="removeConcept">
    <soap:body use="literal"/>

  </wsdl:input>
  <wsdl:fault name="InvalidDataException">
    <soap:fault name="InvalidDataException" use="literal"/>
  </wsdl:fault>
  <wsdl:fault name="ConceptOrPropertyNotFoundException">
    <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
  </wsdl:fault>
```

```
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>

    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="addParent">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="addParent">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:fault name="InvalidDataException">
      <soap:fault name="InvalidDataException" use="literal"/>

    </wsdl:fault>
    <wsdl:fault name="ConceptOrPropertyNotFoundException">
      <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="addRange">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="addRange">
      <soap:body use="literal"/>

    </wsdl:input>
    <wsdl:fault name="InvalidDataException">
      <soap:fault name="InvalidDataException" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="ConceptOrPropertyNotFoundException">
      <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="modifyComment">

    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="modifyComment">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:fault name="InvalidDataException">
      <soap:fault name="InvalidDataException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="remove">

    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="remove">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="removeResponse">
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="InvalidDataException">
      <soap:fault name="InvalidDataException" use="literal"/>

    </wsdl:fault>
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="removeParent">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="removeParent">
      <soap:body use="literal"/>

    </wsdl:input>
    <wsdl:fault name="InvalidDataException">
      <soap:fault name="InvalidDataException" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="ConceptOrPropertyNotFoundException">
      <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="addConcept">
```

```
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="addConcept">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConceptOrPropertyAlreadyAvailableException">
        <soap:fault name="ConceptOrPropertyAlreadyAvailableException" use="literal"/>

      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="create">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="create">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="createResponse">
        <soap:body use="literal"/>

      </wsdl:output>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="EngineNotFoundException">
        <soap:fault name="EngineNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="addDomain">

      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="addDomain">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConceptOrPropertyNotFoundException">
        <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>

      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="removeComment">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="removeComment">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>

      </wsdl:fault>
      <wsdl:fault name="ConceptOrPropertyNotFoundException">
        <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="removeProperty">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="removeProperty">
        <soap:body use="literal"/>

      </wsdl:input>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConceptOrPropertyNotFoundException">
        <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="EngineNotFoundException">
        <soap:fault name="EngineNotFoundException" use="literal"/>

      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="addComment">
```

```
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="addComment">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>

      </wsdl:fault>
      <wsdl:fault name="ConceptOrPropertyNotFoundException">
        <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="addLabel">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="addLabel">
        <soap:body use="literal"/>

      </wsdl:input>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="ConceptOrPropertyNotFoundException">
        <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="OntologyEditionServiceImplService">
    <wsdl:port binding="tns:OntologyEditionServiceImplServiceSoapBinding"
               name="OntologyEditionServiceImplPort">
      <soap:address location="http://argentera.inria.fr/semservices/OntologyEdition"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

A.3 Annotation management service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="http://edit.semservices.inria.fr/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        name="AnnotationEditionServiceImplService"
        targetNamespace="http://edit.semservices.inria.fr/">
  <wsdl:types>
<xs:schema xmlns="http://jaxb.dev.java.net/array"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           attributeFormDefault="unqualified"
           elementFormDefault="unqualified"
           targetNamespace="http://jaxb.dev.java.net/array">
<xs:complexType final="#all" name="stringArray">
<xs:sequence>
<xs:element maxOccurs="unbounded" minOccurs="0" name="item" nillable="true" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:schema>
<xsd:schema xmlns:ns0="http://jaxb.dev.java.net/array"
            attributeFormDefault="unqualified"
            elementFormDefault="qualified"
            targetNamespace="http://edit.semservices.inria.fr/">
<xsd:element name="InvalidDataException" type="tns:InvalidDataException"/>
<xsd:complexType name="InvalidDataException">
<xsd:sequence>
<xsd:element name="invalidData" nillable="true" type="xsd:string"/>
<xsd:element name="invalidURI" nillable="true" type="xsd:boolean"/>
<xsd:element name="malformedData" nillable="true" type="xsd:boolean"/>

<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
```

```
</xsd:complexType>
<xsd:element name="EngineNotFoundException" type="tns:EngineNotFoundException"/>
<xsd:complexType name="EngineNotFoundException">
<xsd:sequence>
<xsd:element name="invalidEngineID" nillable="true" type="xsd:long"/>
<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="modify" type="tns:modify"/>
<xsd:complexType name="modify">
<xsd:sequence>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
<xsd:element minOccurs="0" name="kind" type="xsd:string"/>
<xsd:element minOccurs="0" name="xpath" type="xsd:string"/>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="used-namespaces"
             type="ns0:stringArray"/>

<xsd:element minOccurs="0" name="content" type="xsd:string"/>
<xsd:element minOccurs="0" name="update" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="remove" type="tns:remove"/>
<xsd:complexType name="remove">
<xsd:sequence>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
<xsd:element minOccurs="0" name="update" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="removeResponse" type="tns:removeResponse"/>
<xsd:complexType name="removeResponse">
<xsd:sequence>
<xsd:element name="return" type="xsd:boolean"/>

</xsd:sequence>
</xsd:complexType>
<xsd:element name="create" type="tns:create"/>
<xsd:complexType name="create">
<xsd:sequence>
<xsd:element minOccurs="0" name="file" type="xsd:string"/>
<xsd:element minOccurs="0" name="content" type="xsd:string"/>
<xsd:element minOccurs="0" name="override" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="update" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="createResponse" type="tns:createResponse"/>
<xsd:complexType name="createResponse">
<xsd:sequence>
<xsd:element name="return" type="xsd:boolean"/>
</xsd:sequence>

</xsd:complexType>
</xsd:schema>
  </wsdl:types>
  <wsdl:message name="createResponse">
    <wsdl:part element="tns:createResponse" name="result">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="create">
    <wsdl:part element="tns:create" name="parameters">
    </wsdl:part>

  </wsdl:message>
  <wsdl:message name="EngineNotFoundException">
    <wsdl:part element="tns:EngineNotFoundException" name="EngineNotFoundException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="modify">
    <wsdl:part element="tns:modify" name="parameters">
    </wsdl:part>
  </wsdl:message>
```

```
<wsdl:message name="remove">
  <wsdl:part element="tns:remove" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="InvalidDataException">
  <wsdl:part element="tns:InvalidDataException" name="InvalidDataException">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="removeResponse">

  <wsdl:part element="tns:removeResponse" name="result">
  </wsdl:part>
</wsdl:message>
<wsdl:portType name="AnnotationEditionService">
  <wsdl:operation name="modify">
    <wsdl:input message="tns:modify" name="modify">
  </wsdl:input>
    <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
  </wsdl:fault>

    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
  </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="remove">
    <wsdl:input message="tns:remove" name="remove">
  </wsdl:input>
    <wsdl:output message="tns:removeResponse" name="removeResponse">
  </wsdl:output>
    <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">

  </wsdl:fault>
    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
  </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="create">
    <wsdl:input message="tns:create" name="create">
  </wsdl:input>
    <wsdl:output message="tns:createResponse" name="createResponse">
  </wsdl:output>

    <wsdl:fault message="tns:InvalidDataException" name="InvalidDataException">
  </wsdl:fault>
    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
  </wsdl:fault>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="AnnotationEditionServiceImplServiceSoapBinding"
              type="tns:AnnotationEditionService">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="modify">

    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="modify">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:fault name="InvalidDataException">
      <soap:fault name="InvalidDataException" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>

    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="remove">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="remove">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="removeResponse">
      <soap:body use="literal"/>

    </wsdl:output>
```

```
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="EngineNotFoundException">
        <soap:fault name="EngineNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="create">

      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="create">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="createResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="InvalidDataException">
        <soap:fault name="InvalidDataException" use="literal"/>

      </wsdl:fault>
      <wsdl:fault name="EngineNotFoundException">
        <soap:fault name="EngineNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="AnnotationEditionServiceImplService">
    <wsdl:port binding="tns:AnnotationEditionServiceImplServiceSoapBinding"
               name="AnnotationEditionServiceImplPort">
      <soap:address location="http://argentera.inria.fr/semservices/Annotation"/>

    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

A.4 Retieval service WSDL

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:tns="http://query.semservices.inria.fr/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        name="SparqlQueryServiceImplService"
        targetNamespace="http://query.semservices.inria.fr/">
  <wsdl:types>
<xsd:schema attributeFormDefault="unqualified"
            elementFormDefault="qualified"
            targetNamespace="http://query.semservices.inria.fr/">
<xsd:element name="MalformedQuery" type="tns:MalformedQuery"/>
<xsd:complexType name="MalformedQuery">
<xsd:sequence>
<xsd:element name="serialVersionUID" nillable="true" type="xsd:long"/>
<xsd:element name="message" nillable="true" type="xsd:string"/>
<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="QueryRequestRefused" type="tns:QueryRequestRefused"/>
<xsd:complexType name="QueryRequestRefused">
<xsd:sequence>
<xsd:element name="serialVersionUID" nillable="true" type="xsd:long"/>
<xsd:element name="message" nillable="true" type="xsd:string"/>

<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="EngineNotFoundException" type="tns:EngineNotFoundException"/>
<xsd:complexType name="EngineNotFoundException">
<xsd:sequence>
<xsd:element name="invalidEngineID" nillable="true" type="xsd:long"/>
<xsd:element name="message" nillable="true" type="xsd:string"/>
```

```
</xsd:sequence>
</xsd:complexType>
<xsd:element name="ConceptOrPropertyNotFoundException"
             type="tns:ConceptOrPropertyNotFoundException"/>
<xsd:complexType name="ConceptOrPropertyNotFoundException">
<xsd:sequence>
<xsd:element name="notionURI" nillable="true" type="xsd:string"/>
<xsd:element name="message" nillable="true" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>

<xsd:element name="query" type="tns:query"/>
<xsd:complexType name="query">
<xsd:sequence>
<xsd:element minOccurs="0" name="query-request" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="queryResponse" type="tns:queryResponse"/>
<xsd:complexType name="queryResponse">
<xsd:sequence>
<xsd:element minOccurs="0" name="query-result" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="isValid" type="tns:isValid"/>
<xsd:complexType name="isValid">
<xsd:sequence>
<xsd:element minOccurs="0" name="query-request" type="xsd:string"/>

<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="isValidResponse" type="tns:isValidResponse"/>
<xsd:complexType name="isValidResponse">
<xsd:sequence>
<xsd:element name="is-valid" type="xsd:boolean"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getInstanceNb" type="tns:getInstanceNb"/>
<xsd:complexType name="getInstanceNb">
<xsd:sequence>
<xsd:element minOccurs="0" name="uri" type="xsd:string"/>
<xsd:element minOccurs="0" name="all" type="xsd:boolean"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>

<xsd:element name="getInstanceNbResponse" type="tns:getInstanceNbResponse"/>
<xsd:complexType name="getInstanceNbResponse">
<xsd:sequence>
<xsd:element name="instance-nb" type="xsd:int"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getRootProperties" type="tns:getRootProperties"/>
<xsd:complexType name="getRootProperties">
<xsd:sequence>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getRootPropertiesResponse" type="tns:getRootPropertiesResponse"/>
<xsd:complexType name="getRootPropertiesResponse">
<xsd:sequence>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="roots" type="xsd:string"/>
</xsd:sequence>

</xsd:complexType>
<xsd:element name="getLabels" type="tns:getLabels"/>
<xsd:complexType name="getLabels">
<xsd:sequence>
<xsd:element minOccurs="0" name="uri" type="xsd:string"/>
<xsd:element minOccurs="0" name="lang" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
```

```
</xsd:complexType>
<xsd:element name="getLabelsResponse" type="tns:getLabelsResponse"/>
<xsd:complexType name="getLabelsResponse">
<xsd:sequence>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="labels" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="isProperty" type="tns:isProperty"/>
<xsd:complexType name="isProperty">

<xsd:sequence>
<xsd:element minOccurs="0" name="uri" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="isPropertyResponse" type="tns:isPropertyResponse"/>
<xsd:complexType name="isPropertyResponse">
<xsd:sequence>
<xsd:element name="is-property" type="xsd:boolean"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="isConcept" type="tns:isConcept"/>
<xsd:complexType name="isConcept">
<xsd:sequence>
<xsd:element minOccurs="0" name="uri" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>

</xsd:complexType>
<xsd:element name="isConceptResponse" type="tns:isConceptResponse"/>
<xsd:complexType name="isConceptResponse">
<xsd:sequence>
<xsd:element name="is-concept" type="xsd:boolean"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="validate" type="tns:validate"/>
<xsd:complexType name="validate">
<xsd:sequence>
<xsd:element minOccurs="0" name="query-request" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="validateResponse" type="tns:validateResponse"/>
<xsd:complexType name="validateResponse">
<xsd:sequence>

<xsd:element minOccurs="0" name="errors" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getParents" type="tns:getParents"/>
<xsd:complexType name="getParents">
<xsd:sequence>
<xsd:element minOccurs="0" name="uri" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getParentsResponse" type="tns:getParentsResponse"/>
<xsd:complexType name="getParentsResponse">
<xsd:sequence>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="parents" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getChildren" type="tns:getChildren"/>

<xsd:complexType name="getChildren">
<xsd:sequence>
<xsd:element minOccurs="0" name="uri" type="xsd:string"/>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getChildrenResponse" type="tns:getChildrenResponse"/>
<xsd:complexType name="getChildrenResponse">
<xsd:sequence>
```

```
<xsd:element maxOccurs="unbounded" minOccurs="0" name="children" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getComments" type="tns:getComments"/>
<xsd:complexType name="getComments">
<xsd:sequence>
<xsd:element minOccurs="0" name="uri" type="xsd:string"/>
<xsd:element minOccurs="0" name="lang" type="xsd:string"/>

<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getCommentsResponse" type="tns:getCommentsResponse"/>
<xsd:complexType name="getCommentsResponse">
<xsd:sequence>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="comments" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getRootConcepts" type="tns:getRootConcepts"/>
<xsd:complexType name="getRootConcepts">
<xsd:sequence>
<xsd:element minOccurs="0" name="engine-key" type="xsd:long"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="getRootConceptsResponse" type="tns:getRootConceptsResponse"/>
<xsd:complexType name="getRootConceptsResponse">

<xsd:sequence>
<xsd:element maxOccurs="unbounded" minOccurs="0" name="roots" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
  </wsdl:types>
  <wsdl:message name="validateResponse">
    <wsdl:part element="tns:validateResponse" name="result">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getLabelsResponse">

    <wsdl:part element="tns:getLabelsResponse" name="result">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getLabels">
    <wsdl:part element="tns:getLabels" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getRootProperties">
    <wsdl:part element="tns:getRootProperties" name="parameters">

    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="EngineNotFoundException">
    <wsdl:part element="tns:EngineNotFoundException" name="EngineNotFoundException">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="validate">
    <wsdl:part element="tns:validate" name="parameters">
    </wsdl:part>

  </wsdl:message>
  <wsdl:message name="isValid">
    <wsdl:part element="tns:isValid" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="getParentsResponse">
    <wsdl:part element="tns:getParentsResponse" name="result">
    </wsdl:part>
  </wsdl:message>

  <wsdl:message name="getRootConcepts">
    <wsdl:part element="tns:getRootConcepts" name="parameters">
    </wsdl:part>
  </wsdl:message>
```

```
<wsdl:message name="getChildren">
  <wsdl:part element="tns:getChildren" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="isValidResponse">

  <wsdl:part element="tns:isValidResponse" name="result">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="ConceptOrPropertyNotFoundException">
  <wsdl:part element="tns:ConceptOrPropertyNotFoundException"
             name="ConceptOrPropertyNotFoundException">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="getParents">
  <wsdl:part element="tns:getParents" name="parameters">

  </wsdl:part>
</wsdl:message>
<wsdl:message name="isPropertyResponse">
  <wsdl:part element="tns:isPropertyResponse" name="result">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="getInstanceNbResponse">
  <wsdl:part element="tns:getInstanceNbResponse" name="result">
  </wsdl:part>

</wsdl:message>
<wsdl:message name="getCommentsResponse">
  <wsdl:part element="tns:getCommentsResponse" name="result">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="getRootPropertiesResponse">
  <wsdl:part element="tns:getRootPropertiesResponse" name="result">
  </wsdl:part>
</wsdl:message>

<wsdl:message name="isConcept">
  <wsdl:part element="tns:isConcept" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="getRootConceptsResponse">
  <wsdl:part element="tns:getRootConceptsResponse" name="result">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="getInstanceNb">

  <wsdl:part element="tns:getInstanceNb" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="MalformedQuery">
  <wsdl:part element="tns:MalformedQuery" name="MalformedQuery">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="getComments">
  <wsdl:part element="tns:getComments" name="parameters">

  </wsdl:part>
</wsdl:message>
<wsdl:message name="getChildrenResponse">
  <wsdl:part element="tns:getChildrenResponse" name="result">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="QueryRequestRefused">
  <wsdl:part element="tns:QueryRequestRefused" name="QueryRequestRefused">
  </wsdl:part>

</wsdl:message>
<wsdl:message name="queryResponse">
  <wsdl:part element="tns:queryResponse" name="result">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="isProperty">
```

```
    <wsdl:part element="tns:isProperty" name="parameters">
    </wsdl:part>
  </wsdl:message>

  <wsdl:message name="isConceptResponse">
    <wsdl:part element="tns:isConceptResponse" name="result">
    </wsdl:part>
  </wsdl:message>
  <wsdl:message name="query">
    <wsdl:part element="tns:query" name="parameters">
    </wsdl:part>
  </wsdl:message>
  <wsdl:portType name="SparqlQueryService">

    <wsdl:operation name="query">
      <wsdl:input message="tns:query" name="query">
    </wsdl:input>
      <wsdl:output message="tns:queryResponse" name="queryResponse">
    </wsdl:output>
      <wsdl:fault message="tns:QueryRequestRefused" name="QueryRequestRefused">
    </wsdl:fault>
      <wsdl:fault message="tns:MalformedQuery" name="MalformedQuery">
    </wsdl:fault>

      <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="isValid">
      <wsdl:input message="tns:isValid" name="isValid">
    </wsdl:input>
      <wsdl:output message="tns:isValidResponse" name="isValidResponse">
    </wsdl:output>
      <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">

    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getInstanceNb">
      <wsdl:input message="tns:getInstanceNb" name="getInstanceNb">
    </wsdl:input>
      <wsdl:output message="tns:getInstanceNbResponse" name="getInstanceNbResponse">
    </wsdl:output>
     <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
                 name="ConceptOrPropertyNotFoundException">
    </wsdl:fault>

    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getRootProperties">
      <wsdl:input message="tns:getRootProperties" name="getRootProperties">
    </wsdl:input>
      <wsdl:output message="tns:getRootPropertiesResponse"
                   name="getRootPropertiesResponse">
    </wsdl:output>
      <wsdl:fault message="tns:EngineNotFoundException"
                  name="EngineNotFoundException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getLabels">
      <wsdl:input message="tns:getLabels" name="getLabels">
    </wsdl:input>
      <wsdl:output message="tns:getLabelsResponse" name="getLabelsResponse">
    </wsdl:output>
    <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
                 name="ConceptOrPropertyNotFoundException">
    </wsdl:fault>

    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
    </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="isProperty">
      <wsdl:input message="tns:isProperty" name="isProperty">
    </wsdl:input>
```

```
    <wsdl:output message="tns:isPropertyResponse" name="isPropertyResponse">
  </wsdl:output>
    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">

  </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="isConcept">
    <wsdl:input message="tns:isConcept" name="isConcept">
  </wsdl:input>
    <wsdl:output message="tns:isConceptResponse" name="isConceptResponse">
  </wsdl:output>
    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
  </wsdl:fault>

  </wsdl:operation>
  <wsdl:operation name="validate">
    <wsdl:input message="tns:validate" name="validate">
  </wsdl:input>
    <wsdl:output message="tns:validateResponse" name="validateResponse">
  </wsdl:output>
    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
  </wsdl:fault>
  </wsdl:operation>

  <wsdl:operation name="getParents">
    <wsdl:input message="tns:getParents" name="getParents">
  </wsdl:input>
    <wsdl:output message="tns:getParentsResponse" name="getParentsResponse">
  </wsdl:output>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
  </wsdl:fault>
    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
  </wsdl:fault>

  </wsdl:operation>
  <wsdl:operation name="getChildren">
    <wsdl:input message="tns:getChildren" name="getChildren">
  </wsdl:input>
    <wsdl:output message="tns:getChildrenResponse" name="getChildrenResponse">
  </wsdl:output>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
  </wsdl:fault>
    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">

  </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="getComments">
    <wsdl:input message="tns:getComments" name="getComments">
  </wsdl:input>
    <wsdl:output message="tns:getCommentsResponse" name="getCommentsResponse">
  </wsdl:output>
  <wsdl:fault message="tns:ConceptOrPropertyNotFoundException"
              name="ConceptOrPropertyNotFoundException">
  </wsdl:fault>

    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">
  </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="getRootConcepts">
    <wsdl:input message="tns:getRootConcepts" name="getRootConcepts">
  </wsdl:input>
    <wsdl:output message="tns:getRootConceptsResponse" name="getRootConceptsResponse">
  </wsdl:output>
    <wsdl:fault message="tns:EngineNotFoundException" name="EngineNotFoundException">

  </wsdl:fault>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="SparqlQueryServiceImplServiceSoapBinding"
              type="tns:SparqlQueryService">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
<wsdl:operation name="query">
  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="query">
    <soap:body use="literal"/>

  </wsdl:input>
  <wsdl:output name="queryResponse">
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="QueryRequestRefused">
    <soap:fault name="QueryRequestRefused" use="literal"/>
  </wsdl:fault>
  <wsdl:fault name="MalformedQuery">
    <soap:fault name="MalformedQuery" use="literal"/>

  </wsdl:fault>
  <wsdl:fault name="EngineNotFoundException">
    <soap:fault name="EngineNotFoundException" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getInstanceNb">
  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="getInstanceNb">
    <soap:body use="literal"/>

  </wsdl:input>
  <wsdl:output name="getInstanceNbResponse">
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="ConceptOrPropertyNotFoundException">
    <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
  </wsdl:fault>
  <wsdl:fault name="EngineNotFoundException">
    <soap:fault name="EngineNotFoundException" use="literal"/>

  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="isValid">
  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="isValid">
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="isValidResponse">
    <soap:body use="literal"/>

  </wsdl:output>
  <wsdl:fault name="EngineNotFoundException">
    <soap:fault name="EngineNotFoundException" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getRootProperties">
  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="getRootProperties">
    <soap:body use="literal"/>

  </wsdl:input>
  <wsdl:output name="getRootPropertiesResponse">
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="EngineNotFoundException">
    <soap:fault name="EngineNotFoundException" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
<wsdl:operation name="getLabels">

  <soap:operation soapAction="" style="document"/>
  <wsdl:input name="getLabels">
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output name="getLabelsResponse">
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="ConceptOrPropertyNotFoundException">
```

```
      <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>

    </wsdl:fault>
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="isProperty">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="isProperty">
      <soap:body use="literal"/>

    </wsdl:input>
    <wsdl:output name="isPropertyResponse">
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="isConcept">

    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="isConcept">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="isConceptResponse">
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>

    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="validate">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="validate">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="validateResponse">
      <soap:body use="literal"/>

    </wsdl:output>
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="getParents">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="getParents">
      <soap:body use="literal"/>

    </wsdl:input>
    <wsdl:output name="getParentsResponse">
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="ConceptOrPropertyNotFoundException">
      <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="EngineNotFoundException">
      <soap:fault name="EngineNotFoundException" use="literal"/>

    </wsdl:fault>
  </wsdl:operation>
  <wsdl:operation name="getChildren">
    <soap:operation soapAction="" style="document"/>
    <wsdl:input name="getChildren">
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="getChildrenResponse">
      <soap:body use="literal"/>

    </wsdl:output>
```

```
      <wsdl:fault name="ConceptOrPropertyNotFoundException">
        <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>
      </wsdl:fault>
      <wsdl:fault name="EngineNotFoundException">
        <soap:fault name="EngineNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getComments">

      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="getComments">
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output name="getCommentsResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="ConceptOrPropertyNotFoundException">
        <soap:fault name="ConceptOrPropertyNotFoundException" use="literal"/>

      </wsdl:fault>
      <wsdl:fault name="EngineNotFoundException">
        <soap:fault name="EngineNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
    <wsdl:operation name="getRootConcepts">
      <soap:operation soapAction="" style="document"/>
      <wsdl:input name="getRootConcepts">
        <soap:body use="literal"/>

      </wsdl:input>
      <wsdl:output name="getRootConceptsResponse">
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="EngineNotFoundException">
        <soap:fault name="EngineNotFoundException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="SparqlQueryServiceImplService">
    <wsdl:port binding="tns:SparqlQueryServiceImplServiceSoapBinding"
               name="SparqlQueryServiceImplPort">
      <soap:address location="http://argentera.inria.fr/semservices/Query"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Appendix B

Explanation of the Generis KB file

```
<?php
// inclusion of the Generis API
include("./generis/generis/taoAPI.php");

// uri analysis
$uri = $_SERVER['REQUEST_URI'];
$i = strpos($uri, '?');
$uri_base = $uri;
if ($i !== false) {
  $uri_base = substr($uri, 0, $i);
}
$path = explode('/', $uri_base);

// get the module name (palette database repository)
$module = (isset($path[2]))?$path[2]:'';

// get the query
$query = (isset($_GET['query']))?$_GET['query']:'';
// in case of value 'fetch', returns the XML description of
// element specified in '$r', in case of value 'keywords', returns
// the result of search base on keywords specified on field keywords

$lang = (isset($_GET['lang']))?$_GET['lang']:'';
//$r = (isset($_GET['id']))?$_GET['id']:'';
// get keywords
$keywords = (isset($_GET['keywords']))?$_GET['keywords']:'';


//palette module authentication

// catch error if the module isn't palette
if ($module != 'palette') {
        echo 'Error: unknown kb';
        die();
}

// authenticates to the repository
$session = authenticate(array("palette"), array("palette"),
                        array(""), array($module));

$results = '';
```

```
// in this part, the parameters are tested in order to see
// which service has to be impacted

// if the parameters query and lang are empty
// call of the service exportxmlRDF which allows to export
// all data of the repository to RDF format
if (empty($query) && empty($lang)) {
        // call of the method 'exportxmlRDF' with parameter
// session resulting from authenticate method
        $results = exportxmlRDF($session["pSession"]);
        echo $results;
        exit();
}

// if the parameter lang isn't empty
if (!empty($lang)) {
        // in case of 'fetch' call
        if ($lang == 'fetch') {
                if (!empty($r)) {
              // returns the XML description of element
              // specified in parameter $r
              // call of the method 'getXMLDescription' with
              // parameter session resulting from authenticate
              // method, and the uri of the element for which
              // we want a description
                  $results = getXMLDescription($session["pSession"],
                                               urldecode($r));

                } else {
                        echo 'Error: r parameter is empty';
                }
// case of search based on keywords, returns results
        } else if ($lang == 'keywords') {
                if (!empty($keywords)) {
                // search by keywords
                // call of the method 'fullTextSearch' having as
      // parameters session, resulting from
      // authenticate method, and an array of keywords
                   $results = fullTextSearch($session["pSession"],
                                             explode(',', $keywords));
                } else {
                        echo 'Error: keywords parameter is empty';
                }
        }
} else {
        // sparql query
        // calls the method 'sparql' having as parameters the
// session resulting from method authenticate
// previously called, and the Sparql query
        $results = sparql($session["pSession"],$query);
}
echo $results;
?>
```

Appendix C

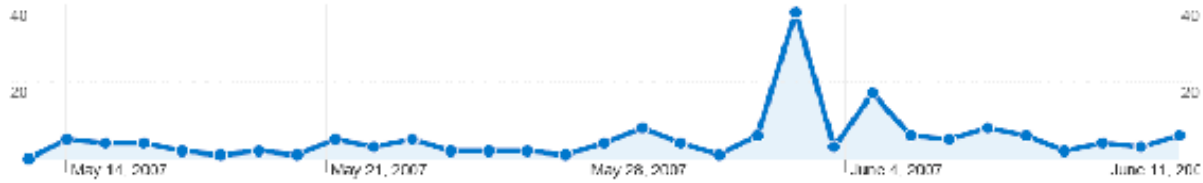
Statistics on the use of SweetWiki by Palette CoPs

C.1 ePrep (12, 13/06/2007)

At the moment, there are 34 persons from the ePrep CoP who are subscribed to Wikiprepas.

In addition to the Web "Main", six WikiWebs have been created, they correspond to the fields of knowledge taught by the members of the CoP (mathematics, physics, chemistry, computer science, English). Another WikiWeb deals with the "great scientists" of these domains of knowledge. There are around 75 pages on these WikiWebs (this number does not include users' homepages and SweetWiki documentation pages). Nevertheless, the great majority of these pages are empty: they have been created but never filled with information, there are approximately only 23 pages that contain information (this number includes the "test pages" created by some users when training to use SweetWiki).

C.1.1 Distribution of the number of visits/sessions for all visitors:160 sessions



The diagram above shows the evolution of the number of visits on Wikiprepas during the observation period. This number oscillates between 1 and 5, increases to 8 just before reaching the pick of 38 sessions opened by 20 persons, on June 2nd 2007 (which represents 23.75% of the total number of sessions during the observation period of one month). This high and sudden frequentation of Wikiprepas is justified by the training organized that day for ePrep members on the use of SweetWiki. After this training, we see that the activity on Wikiprepas has changed: the aspect of the curve differs from what it is from May 13th to June 2nd, now the number of daily sessions oscillates between 5 and 17, however, it tends to decrease.

In addition, we can mention that:

- only 23.75% of the visits are direct (visits from people who clicked a bookmark to come to Wikiprepas or who typed its URL directly into their browser);
- 38.75% are visits from persons who clicked on a link to Wikiprepas provided in another site (e.g. www.eprep.org);

- 37.50% reach Wikiprepas after submitting a query to an engine (e.g. google, msn).

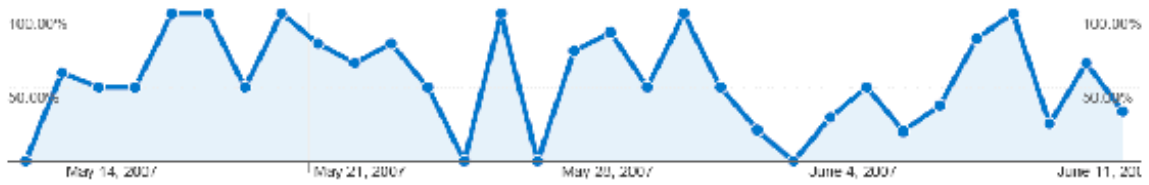
C.1.2 Time on Site for all visitors

The average of the time spent on Wikiprepas by its users during the observation period is 07mn58s.

We notice that this curve has the same aspect that the curve describing the evolution of the visits number on Wikiprepas. Indeed, the time spent on Wikiprepas is very low before June 1st 2007 (average time of 00mn47s) and a pick appears on June 2nd 2007, with an average time of 22mn46s (up to 42mn55s, around 10am, which was the time of the training).

Thus, we conclude to a change in the activity of the CoP ePrep on Wikiprepas after this training, both from the point of view of the number of visits and also the duration of these visits: more visits that last longer (average time of 04mn51s).

C.1.3 Bounce rate



Taking into account the data analyzed previously, we know that an average bounce rate on the entire observation period (45.63%) won't provide further information. Therefore, we distinguish the bounce rate:

- Before June 2nd: 66.13%
- From June 2nd: 32.65%

Which means a bounce rate decrease of 50.62% after the training. This is a good indicator of the interest of Wikiprepas users in its content and usage: before the meeting held on June 2nd, 66.13% of the sessions were limited to one page access, whereas after the meeting, the tendency is reversed.

C.1.4 Top Content

During the observation period, 229 URLs have been visited (at least once). Among these URLs, the homepage of Wikiprepas (`http://argentera.inria.fr/wikiprepas/data/Main/MainHome.jsp`) is, by far, the most frequently visited since it has been viewed 193 times. Then, the "Recent changes" page is often accessed (83 times).

However, the navigation track shows that this page started to be accessed only after the training. Indeed, before the training, neither the faceted navigation using the tags on the

pages, nor the "Recent changes" page nor the "Advanced search" functionality were used by the visitors of Wikiprepas.

In fact, the visitors used to navigate through the wiki via the Webs (especially the Web "Main") and the explicit links provided in the content of pages to access other ones.

After the training, we observe the use of the "Advanced search" functionality and the access to the "Recent changes" page, but still not the use of the tags to navigate.

C.1.5 Conclusion

The analysis made on Wikiprepas usage shows that there is a gap between the scenario defined ([8, D.PAR.03]) and the real use of Wikiprepas. In spite of the number of registered persons on the wiki and the quantity of created pages, Wikiprepas is not very used; therefore, the advantage that can be taken of using its functionalities cannot be observed.

C.2 @pretic (13/06/2007)

At the moment, there are 22 persons of @pretic who are subscribed to SweetWiki. There is only one WikiWeb on the instance of SweetWiki used by @Pretic, it is the Web "Main" which contains 25 pages (this number does not include users' homepages and SweetWiki documentation pages) involving the participation of approximately 11 persons. In these pages, some activities of the members are described and shared, some computer science notions are defined and collaborative work is done.

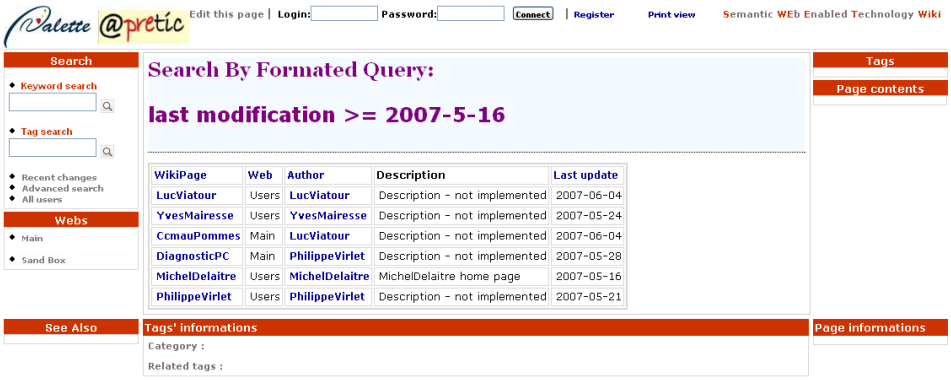
C.2.1 Distribution of the number of visits/sessions for all visitors

The diagram above shows the number of sessions opened on SweetWiki during the observation period. It merely shows a regular but slight activity on the wiki, with between 1 and 7 sessions (opened by 2 to 6 visitors) till June 12th, when 17 visits are made on SweetWiki by 13 visitors. This increase coincides with the presentation of SweetWiki to a restricted group of @pretic members. Concerning the access points to the implementation of SweetWiki that is dedicated to the @pretic CoP:

- 43.40% of the visits are direct;
- 24.53% of the visits are made through an intermediary site which references SweetWiki (e.g. `www.apretic.be`, `www.stecrifa.ulg.ac.be`);
- 32.08% of the visits occur through the use of a query engine that provides a link to SweetWiki (here google).

C.2.2 Time on Site for all visitors

The average of the time spent on SweetWiki for @pretic by its users during the observation period is 03mn30s. This diagram shows the average time spent on SweetWiki by its users from the @pretic CoP. It confirms the low level of use of SweetWiki by this CoP. The dates of May 16th, 24th and 28th, as well as June 4th and 11th contrast with the rest of the values on the diagram. Indeed, they correspond to days when some pages have been created or modified on the wiki, as illustrated in the figure below.



C.2.3 Average Pageviews for all visitors

The diagram above illustrates the average number of pages viewed per visit. As shown, the days when this average is the highest most often coincide with the previously enumerated dates when pages have been created or modified. This can be a clue to conclude that the wiki is yet in a period of trial by few persons who test it by creating pages and navigating through the already existing ones.

C.2.4 Bounce rate

This diagram confirms what has been already observed: in spite of the periods of pages creation or modification, the wiki is not very active and the visits are short and limited to 1 to 3 pages. There's an average of 43.40% of single-page visits.

C.2.5 Top Content

During the observation period, 116 URLs have been visited (at least once). Among these URLs, the homepage of SweetWiki (`http://argentera.inria.fr/swikiapretic/data/Main/MainHome.jsp`) is the most frequently visited since it has been viewed 88 times. Then, the "Recent changes" page is often accessed (76 times). The navigation track shows that the faceted navigation using the tags on the pages as well as "Recent changes" page are used by the visitors of the wiki, whereas the "Advanced search" functionality is not often used by the visitors of the wiki. This is probably due to the fact that, at the moment, the pages are not numerous, and thus, it is very easy for the members who participate and use the wiki to find them in the "Recent changes" listing or by means of the tag-based search (since they make use of the pages tagging functionality).

C.2.6 Conclusion

Even though the scenario defined for the use of SweetWiki by @pretic is not implemented yet, we notice that the slight activity on the wiki is nevertheless characterized by the use of the functionalities that are particular to SweetWiki (such as tagging and tag-based search).

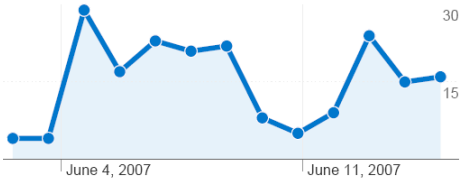
C.3 STE-CRIFA

SweetWiki is used by some researchers of STE-CRIFA to manage the different projects on which they work. As they cannot often exchange and discuss altogether synchronously, they use SweetWiki to work collaboratively, each one providing knowledge and correcting the others. Diverse topics are discussed through SweetWiki, such as the structure of STE-CRIFA Web site, brainstorming on the vocabulary used, the questions or problems met with Palette tools, etc.

At the moment, there are 13 persons who are registered to use SweetWiki. There are two WikiWebs on the instance of SweetWiki dedicated to STE-CRIFA, the Web "Main" which contains 12 pages and the Web "Private" which contains approximately 96 pages, the pages which belong to this Web are only accessible to the authenticated users..

Contrarily to the statistics provided for ePrep and @pretic, the observations of the activities on the wiki for STE-CRIFA extend from June 2nd till June 14th. This is related to the fact that the observations made before this period relied on the activities on the WikiWeb "Main" which is public. However, as noticed, most of the activity takes place in the WikiWeb "Private", and thus, the statistics on the WikiWeb "Main" are not significant of the use of SweetWiki by the members of STE-CRIFA. Therefore, the statistics provided in this analysis illustrate the use of SweetWiki as a whole by the researchers of STE-CRIFA.

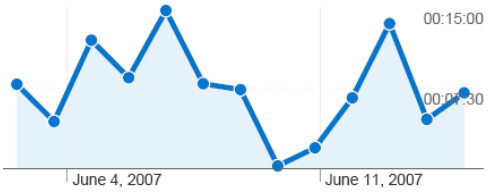
C.3.1 Distribution of the number of visits/sessions for all visitors



The diagram above illustrates the number of sessions opened on SweetWiki during the observation period. We can see that the community is quite active with a total number of 197 sessions: from 4 to 29 visits per day on the wiki (from 3 to 15 distinct visitors, which means that there have sometimes been more visitors than the number of registered persons). Concerning the access points to the STE-CRIFA SweetWiki:

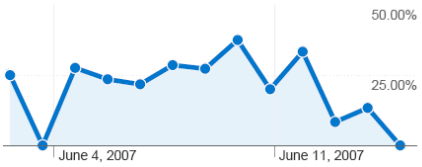
- 65.99% of the visits are direct: direct access to the wiki through its URL;
- 16.24% of the visits are oriented by referring sites such as `www.stecrifa.ulg.ac.be`;
- 17.77% of the visits are resulting from a query submitted on a search engine (in this case: google).

C.3.2 Time on Site for all visitors

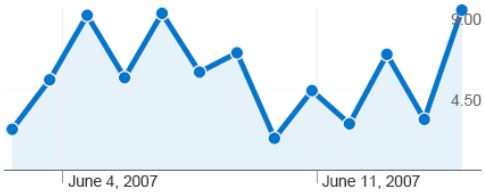


The average time spent on the wiki is equal to 9mn01s. It is only one minute more than the average time spent on Wikiprepas by the ePrep members, which is not a lot. However, contrarily to Wikiprepas, we notice that the average time spent on SweetWiki by the STE-CRIFA members is not due to a sudden and punctual activity on the wiki, but rather to a continuous and regular activity.

C.3.3 Bounce rate



As illustrated in the diagram above, the percentage of sessions that are limited to one page is low, it oscillates between 0 and 37.50% (average of 20.81%). This means that the visitors are quite interested in the content provided by the wiki. As the target of using a wiki is to enable and motivate people to work and learn collaboratively, we can consider that this target is on the right road to be attained. Indeed, regarding the curves (number of visits, time on site, bounce rate) that describe the activity of STE-CRIFA researchers on SweetWiki, we conclude that these users often connect to SweetWiki and, when they do so, they spend time navigating through its content (the sessions are not limited to one-page visits); indeed, as shown in the diagram below which illustrates the distribution of the daily average number of pages visited per session, there is an average number of 6.19 pages visited per session. This means that the visitors are interested in the content they find in SweetWiki, which is the knowledge that is being built by their "colleagues" or that they collaborate to build.



C.3.4 Top Content

During the observation period, 159 URLs have been visited (at least once). Among these URLs, the "Recent changes" page of SweetWiki (`http://argentera.inria.fr/swikiulg/data/Tools/RecentChanges.jsp`) is the most frequently visited since it has been viewed 124 times. Then, several pages belonging to the WikiWeb "Private" are also often accessed. Finally, the homepage of the wiki (`http://argentera.inria.fr/swikiulg/data/Main/MainHome.jsp`) is in the 7th position (viewed 55 times). The navigation track shows that the more accessed WikiWeb is the "Private" Web. The explicit links put on some pages to refer to others are frequently used, whereas the faceted navigation using the tags is not used a lot; on the other hand, the pages are not always tagged. It seems that pages tagging is not systematic for all the users. The "Advanced search" functionality is not often used too. This is probably due to the fact that, at the moment, the wiki is very active, the pages are constantly updated and thus, it is very easy to find a page by viewing the "Recent changes" page, and this is why this page is the most frequently visited.

C.3.5 Conclusion

From the point of view of the "aliveness", there is no doubt on the involvement and participation of the STE-CRIFA researchers subscribed to use SweetWiki. However, there is not a general and full use of the functionalities provided by SweetWiki, maybe because the greatest part of the activity is handled within the "Private" workspace which is dedicated to a restricted group of researchers, and then it is easy for them to find the needed information. But, if this information is to be made accessible to a wider group, it would be worthy to "educate" the users to provide more knowledge by tagging the pages they create/update by reflex. In fact, the number of participants is not the only reason why information should be made more explicit; the information itself, becoming huger and huger, it would be more and harder to access it even with a very restricted group of participants.

Bibliography

[1] B. P. Allen and J. T. Tennis. Building metadata-based navigation using semantic web standards: The dublin core 2003 conference proceedings. In *Proc. of the 2004 Joint ACM/IEEE Conference on Digital Libraries (JCDL'04)*, Tucson, Arizona, USA, June 7-11 2004.

[2] T. Bayes. Studies in the history of probability and statistics: Ix. thomas bayes's essay towards solving a problem in the doctrine of chances. *Biometrika*, 1763.

[3] V. Broughton. Faceted classification as a basis for knowledge organization in a digital environment; the bliss bibliographic classification as a model for vocabulary management and the creation of multi-dimensional knowledge structures. *The new Review of Hypermedia and Multimedia*, pages 67–102, 2001.

[4] M. Buffa and F. Gandon. Wikis et web semantique. In *Proc. of Conference IC 2007*, pages p49–60, Grenoble, France, 2007.

[5] M. Buffa, F. Gandon, and G. Ereteo. *Emerging technologies for semantic web environments: techniques, methods and applications*, chapter A wiki on the semantic web. Fraunhofer Institute for Experimental Software Engineering (IESE), 2007.

[6] P. Cheeseman, M. Self, J. Kelly, J. Stutz, W. Tylor, and D. Freeman. Bayesian classification. In *Seventh National Conference on Artificial Intelligence*, pages 607–611, Saint Paul, Minnesota, 1988.

[7] O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker. Querying the Semantic Web with the CORESE search engine. In *Proc. of the 16th European Conference on Artificial Intelligence (ECAI'2004)*, pages 705–709, Valencia, Spain, 2004. IOS Press.

[8] A. Daele and B. Charlier (eds.). Description of 6 scenarios and of results of 6 validated trials. Deliverable D.PAR.03, Palette FP6-028038, September 2007. Contributors: A. Boukottaya, M. Buffa, M. Charlier, A. Daele, N. Deschryver, A. El Ghali, S. El Helou, L. Esnault, C. Evangelou, A. Giboin, N. Karacapilidis, M. Kunzel, A. Madina, C. Maissen, J. Mikác, A. Milstein, H. Platteaux, R. Peeters, V. Quint, S. Rieppi, M. Saunders, A. Tifous, M. Tzagarakis, E. Vandeput, N. Van de Wiele, I. Vatton, F. Vermeulin and G. Vidou.

[9] W. Dakka, P. G. Ipeirotis, and K. R. Wood. Automatic construction of multifaceted browsing interfaces. In *Proceedings of CIKM'05*, Bremen, Germany, October 31-November 5 2005.

[10] W. Denton. How to make a faceted classification and put it on the web. Technical report, http://www.miskatonic.org, November 2003.

[11] P. V. Dijck. Xfml core - exchangeable faceted metadata language. Specification document of the XFML, 2003.

[12] P. Durville and F. Gandon. Sewese : Semantic web server. In *Proc. of WWW2007 Developers track*, 2007.

[13] A. El Ghali and R. Dieng-Kuntz (eds.). Specification of the CoP-oriented Knowledge Management Tool offering basic CoP-adapted KM services. Deliverable D.KNO.03, Palette FP6-028038, August 2006. *Contributors:* A. El Ghali, R. Dieng-Kuntz, A. Tifous, O. Corby, S. Dehors, C. Evangelou, F. Gandon, A. Giboin, T. Latour, P. Plichart and G. Vidou.

[14] P. Graham. A plan for spam, 2003.

[15] M. Hildrebrand, J. van Ossenbruggen, and L. Hardman. /facet: A browser for heterogeneous semantic web repositories. In *Proc. of the 5th International Semantic Web Conference (ISWC2006)*, volume volume 4273 of Lectures Notes in Computer Science, Athens, GA, USA, November 5-9 2006.

[16] M. Hudon and S. Mas. Analyse des facettes pour la classification des documents institutionnels au gouvernement du quebec. Technical report, Secretariat du Conseil du tresor, Quebec, 2001.

[17] E. Hyvonen, S. Saarela, and K. Viljanen. Ontogator: Combining view- and ontology-based search with semantic browsing. In *Proceedings of XML Finland 2003*, Kuopio, Finland, October 30-31 2003.

[18] M. Mahoui, Z. Miled, A. Godse, H. Kulkarni, and N. Li. Biofacets: Faceted classification for biological information. In *I. C. Society, editor, the 18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*, 2006.

[19] A. Maple. Faceted access: A review of the literature. In *the Music Library Association Annual Meeting*, 10 February 1995.

[20] E. Oren, R. Delbru, and S. Decker. Extending faceted navigation for rdf data. In *Proc. of ISWC'06*, 2006.

[21] J. M. Pierre. Mining knowledge from text collections using automatically generated metadata. In *Fourth International Conference on Practical Aspects of Knowledge Management (PAKM'02)*, 2002.

[22] R. Prieto-Diaz. A faceted approach to building ontologies. In *Proc. of ER2002*, 2002.

[23] S. Ranganathan. *Prolegomena to Library Classification*. Asian Publishing House, Bombay, India, 1967.

[24] A. Seaborne and E. Prud'hommeaux (eds.). SPARQL Query Language for RDF. W3C Candidate Recommendation, 6 April 2006.

[25] G. Smith, M. Czerwinski, B. Meyers, D. Robbins, G. Robertson, and D. S. Tan. Facetmap: A scalable search and browse visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):797–804, September/October 2006.

[26] O. Suominen, K. Viljanen, and E. Hyvnen. User-centric faceted search for semantic portals. In *Proc. of the 4th European Semantic Web Conference (ESWC 2007)*, 2007.

[27] A. Tifous and R. Dieng-Kuntz (eds.). CoP-dependent ontologies. Deliverable D.KNO.02, Palette FP6-028038, March 2006. Contributors: A. Tifous, A. El Ghali, R. Dieng-Kuntz, C. Evangelou, F. Gandon, A. Giboin and G. Vidou.

[28] Y. Tzitzikas, R. Launonen, M. Hakkarainen, P. Kohonen, T. Leppanen, E. Simpanen, H. Tornroos, P. Uusitalo, and P. Vanska. Fastaxon: A system for fast (and faceted) taxonomy design. In *Proceedings of 23th Int. Conf. on Conceptual Modeling, ER'2004*, Shanghai, China, November 2004.

[29] Y. Tzitzikas, N. Spyratos, and P. Constantopoulos. Extended faceted ontologies for web catalogs. Technical Report TR-293, FORTH-ICS, 2001.

[30] Y. Tzitzikas, N. Spyratos, P. Constantopoulos, and A. Analyti. Extended faceted ontologies. In *CAiSE '02: Proceedings of the 14th International Conference on Advanced Information Systems Engineering, Lecture Notes In Computer Science*, volume volume 2348, pages 778–781, London, UK, 2002. Springer-Verlag.

[31] V. Vickery. *Faceted Classification: A Guide to Construction and Use of Special Schemes*. Aslib, 3 Belgrave Square, London, 1960.

[32] W. Yerazunis. The spam-filtering accuracy plateau at 99.9% accuracy and how to get past it. In *MIT Spam Conference*, 2004.