Project no. FP6-028038

Palette

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

Integrated Project
Technology-enhanced learning

D.KNO.03
Specification of the CoP-oriented Knowledge Management Tool offering basic CoP-adapted KM services

Due date of deliverable: July 31, 2006
Actual submission date: August 16, 2006

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 Programme		
Dissemination Level		
R & P	Public	**PU**

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

MODIFICATION CONTROL			
Version	**Date**	**Status**	**Modifications made by**
0.9	05/08/2006	Draft	Adil El Ghali
1.0	06/08/2006	Sent to Reviewers	Rose Dieng-Kuntz
1.1	09/08/2006	Feedback of Reviewer	Manfred Kunzel
1.2	11/08/2006	Feedback of Reviewer	Nikos Karacapilidis
1.3	11/08/2006	Corrections	Adil El Ghali
2.0	12/08/2006	Sent to SC	Rose Dieng-Kuntz

Deliverable manager: Adil El Ghali (INRIA)

List of contributors:
Olivier Corby (INRIA)
Sylvain Dehors (INRIA)
Rose Dieng-Kuntz (INRIA)
Priscille Durville (INRIA)
Adil El Ghali (INRIA)
Christina Evangelou (CTI)
Fabien Gandon (INRIA)
Alain Giboin (INRIA)
Thibault Latour (CRP-HT)
Patrick Plichart (CRP-HT)
Amira Tifous (INRIA)
Géraldine Vidou (CRP-HT)

List of evaluators:
Nikos Karacapilidis
Manfred Kunzel

Summary

This deliverable proposes a preliminary specification of basic KM services interesting for CoPs. It describes their functionalities, their possible interfaces with other services and with the human user; as well as possible uses of these basic KM services by CoPs. Various tools available among the partners, that could offer some of these basic services, are presented. A web service-oriented architecture enabling to offer all such KM services in a modular way with possible interoperability with other Palette services or with some CoP tools, is proposed.

1 Introduction 7

I Building Blocks of KM services 9

2 Knowledge Creation and Annotation 11

2.1 Ontology Creation 11
2.1.1 Definition 11
2.1.2 Functionalities 12
2.1.3 Examples of use in CoPs 14
2.2 Annotation 15
2.2.1 Functionalities 16
2.2.2 Interfaces of the service 17
2.2.3 Examples of use in CoPs 18
2.3 Cooperative Knowledge Creation 19
2.3.1 Functionalities 19
2.3.2 Interfaces of the service 19

3 Knowledge Retrieval and Dissemination 20

3.1 Definition 20
3.2 Knowledge Retrieval 20
3.2.1 Functionalities 20
3.2.2 Interfaces of the service 21
3.2.3 Examples of use in CoPs 22
3.3 Knowledge Dissemination 25
3.3.1 Functionalities 25
3.3.2 Interfaces of the service 25
3.3.3 Examples of use in CoPs 26

4 Knowledge Presentation and Visualization 27

4.1 Functionalities 27
4.2 Interfaces of the service 28
4.3 Examples of use 28
4.3.1 Example of use in CoPs 30

5 Knowledge Evaluation 32

5.1 Functionalities 32
5.2 Interfaces of the service 32
5.3 Examples of use in CoPs 34

6 Knowledge Evolution and Maintenance 37

6.1 Definition 37
6.2 Functionalities 37
6.3 Interface of the service 38
6.3.1 Inputs and Outputs of the service 38
6.4 Examples of use in CoPs 39

II Partners tools offering building blocks services 41

7 Generis4 43

7.1 Architecture 43
7.2 Extension mechanisms 44
7.3 User interfaces and Functional coverage 46
7.4 A use case : "Testing Assisté par Ordinateur" and benchmarking 47
7.5 Generis Deployment 48
7.6 Licence model and documentation 48
7.7 Generis related works in progress 49

8 Corese 50

8.1 Introduction 50
8.1.1 Theoretical Foundations of Corese 50
8.1.2 Corese Ontology Representation Language 53
8.1.3 Corese Query Language 55
8.2 Approximate Semantic Web search 57
8.2.1 Ontological Approximation 57
8.3 Corese Software 57
8.3.1 Architecture 57

9 SeWeSe 60

9.1 Introduction 60
9.2 Architecture 60
9.3 Available components 61
9.4 Links with elementary KM services 63
9.4.1 Ontology editor (Ontology creation) 63
9.4.2 Annotation Editor (Annotation) 64
9.4.3 Clustering (Presentation and Visualization) 65

10 Meat 66

10.1 Introduction 66
10.2 MEAT Components 67

11 Virtual Staff......69

11.1 Introduction......69
11.2 Examples of Queries......72

III Architecture 75

12 Architecture of KM services......77

12.1 Introduction......77
12.2 SOA & SOAP......78
12.3 Architecture......80
12.4 KM system components......81
12.4.1 Services registries......81
12.4.2 Complex services......81
12.4.3 Complex services front-end......82
12.4.4 Widgets......82

IV Use cases and Integration 83

13 Scenarios of usage......85

13.1 The Scenario......85
13.2 Use Cases......86

14 Integration with Mediation services......89

14.1 KM and Mediation interrelation......89
14.2 KM and Mediation services integration scenario......90

15 Conclusion......95

15.1 Integration with CoPs Tools......96
15.2 Further Work......96

Bibliography 101

Chapter 1

Introduction

WP3 aims at offering knowledge management (KM) services for efficient and effective management of the CoP knowledge resources, so as to improve: (i) the access, sharing, and reuse of this knowledge, which can be tacit or explicit, individual or collective, and (ii) the creation of new knowledge. A CoP knowledge resource can be not only a document (report, mail, forum, etc.) materializing knowledge acquired and shared through cooperation between the CoP members but it can also be a person holding tacit knowledge.

Task 3.3 focuses on a CoP-oriented KM tool offering basic CoP-oriented KM services such as knowledge creation and enrichment, knowledge retrieval and dissemination, knowledge presentation and visualization, knowledge evaluation, knowledge evolution and maintenance. As we chose a semantic web-based approach, these KM services will rely on an ontology (describing useful concepts about a CoP, its actors and their competences, its resources such as documents used or produced, its activities, etc.) and on annotation of the CoPs knowledge resources w.r.t. these ontologies.

The meta-models proposed in task 3.1 are useful for understanding a group activity, collaboration, etc. A CoP being a specific kind of such a group, the CoP-dependent ontology to be developed in Task 3.2 will be based on these meta-models. It will consist of CoP-dependent concepts and relations, with which the CoP resources can be annotated. The CoP-oriented KM tool to be specified and developed in Task 3.3 will rely on this CoP-dependent ontology, itself linked to the meta-ontology proposed in Task 3.1.

This deliverable constitutes the preliminary specification of the CoP-oriented KM tool and of the basic KM services it will offer. This deliverable is composed of four parts:

- The first part describes the **building blocks of the KM tool**: the elementary KM services that will be offered. This first part comprises five chapters, each aimed at describing a basic service potentially useful for a CoP. We relied both on the CoP descriptions available in Palette and on the semantic web approach adopted in Palette. For each of the basic services, we specify its intended functionalities, its interfaces with other services (e.g. input and output of the service) as well as its interfaces for interaction with the human user, and we give examples of possible uses in CoPs. The elementary services described in this first part are:

  Knowledge creation and annotation (chapter 2), for enabling a CoP member to create and enrich the CoP ontologies or to annotate the CoP resources by textual annotations and by ontology-based semantic annotations. CoP knowledge such as ontology, annotations or problem-solving cases, can be created cooperatively through collaborative activities.

  - **Knowledge retrieval and dissemination** (chapter 3): for retrieving relevant resources (annotations, people, documents ...) in answer to a user's query or for pushing knowledge towards the user according to his/her profile.
  - **Knowledge presentation and visualization** (chapter 4), for generating friendly graphical interfaces aimed at presenting knowledge to the end-users.
  - **Knowledge evaluation** (chapter 5), for evaluating CoP resources according to a series of evaluation criteria.
  - **Knowledge evolution and maintenance** (chapter 6), for ensuring a coherent evolution of the CoP knowledge resources.
- The second part describes several tools available among the partners, that could offer such elementary services or more complex services corresponding to a combination of such basic services. These tools are:
  - **Generis,** an ontology management tool (chapter 7) could offer ontology creation services,
  - **Corese,** a semantic search engine (chapter 8) could offer knowledge retrieval services,
  - **SeWeSe,** a platform for developing semantic web applications (chapter 9), could offer ontology creation, knowledge annotation and knowledge presentation or visualization services,
  - **MEAT,** an annotation tool based on Natural Language Processing Tools (chapter 10), could offer services of semi-automatic annotation from texts,
  - **Virtual Staff,** a cooperative tool (chapter 11) could offer cooperative problem solving services as a specific kind of knowledge creation services.
- The third part presents the KM tool modular architecture. Since the KM services must be offered not only to human end-users but also to other services, we will rely on a web service-oriented architecture. The components of this architecture are presented in detail.
- The fourth part presents some scenarios of usage and studies about how the KM services specified in WP3 will be interoperable with the mediation services developed in WP4. We give some use cases of invocation of KM services from the mediation tool developed in WP4.

In the conclusion, we will evoke the possible solutions of integration of the proposed KM services with the tools presently used by the CoPs considered in Palette.

Remark:

It must be noticed that this deliverable is only a preliminary version that will be refined after analysis of information on the CoPs to be delivered by WP1.

Part I

Building blocks of KM services

This part describes the **building blocks of the KM tool**: the elementary KM services that will be offered. This first part comprises five chapters, each aimed at describing a basic service potentially useful for a CoP:

- **Knowledge creation and annotation** (chapter 2). The CoP knowledge underlies the CoP ontology, the annotation on the CoP documents or on the CoP itself, and the problem-solving cases. Therefore, we will study ontology creation, annotation (that may be manual, semi-automatic or collaborative) and cooperative creation of CoP knowledge (for example, through collaborative problem solving).
- **Knowledge retrieval and dissemination** (chapter 3) Knowledge retrieval will be guided by the ontology in order to retrieve relevant resources (annotations, people, documents ...) in answer to a user's query, while knowledge dissemination will push relevant knowledge resources towards the user according to his/her profile.
- **Knowledge presentation and visualization** (chapter 4), will enable the generation of friendly graphical interfaces for presenting knowledge to the end-users.
- **Knowledge evaluation** (chapter 5), will enable the evaluation of CoP resources, according to well-defined evaluation criteria.
- **Knowledge evolution and maintenance** (chapter 6) will support a coherent evolution of the CoP knowledge resources.

Chapter 2

Knowledge Creation and Annotation

2.1 Ontology Creation

2.1.1 Definition

The ontology creation service consists of supporting the creation of CoP-dependent ontologies. The ontology creation process follows a number of steps from analyzing the knowledge sources to formalizing the ontology in an ontology description language (such as RDFS, OWL, etc.). At a given stage of this process, the ontology elements are validated by a domain expert; in our case, this task may be carried out through the evaluation service. The Figure 2.1 summarizes this process.

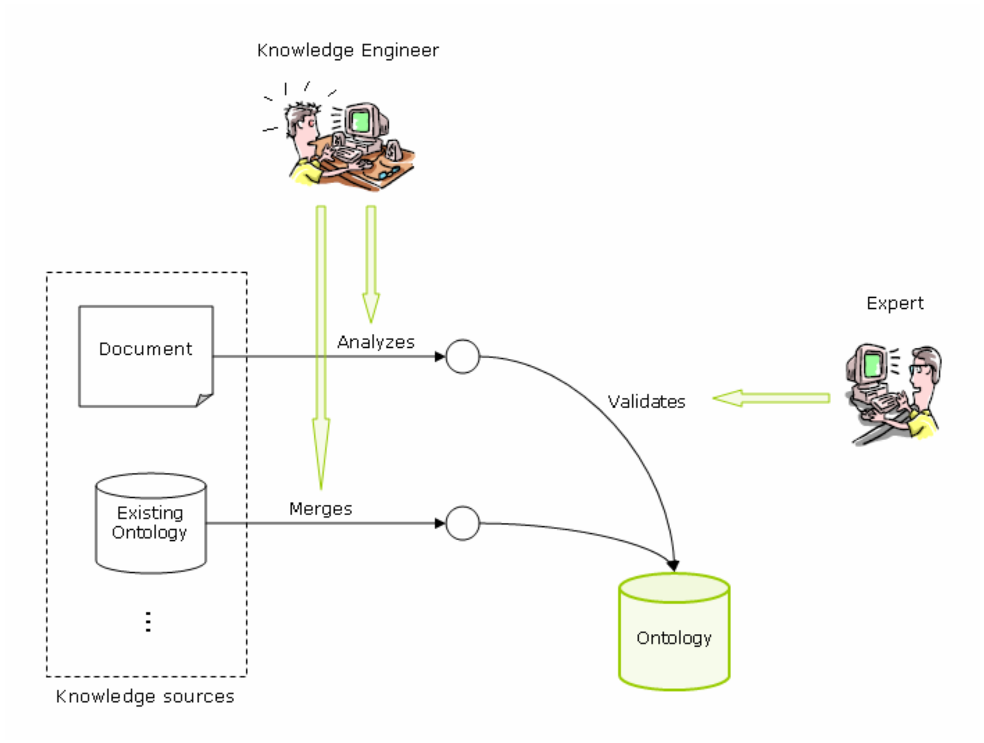


Figure 2.1: Ontology creation process

The details of the creation process may vary according to the methodology adopted, but it minimally consists of:

- identifying the knowledge sources (human sources or textual sources),
- acquiring knowledge from these sources (through knowledge acquisition from humans or through knowledge extraction from texts, using linguistic tools for example),
- determining the terms of the domain (and solving the potential terminological problems that may occur such as synonym terms, homonymy of terms, polysemy, etc),
- determining the concepts and relations to be included in the ontology, with the terms denoting these concepts and relations (possibly in several languages), and with the definitions or explanatory texts documenting these concepts and relations,
- structuring the ontology (e.g. into a hierarchy of concepts and a hierarchy of relations),
- determining possible axioms linking these concepts and relations,
- formalizing the ontology and representing it in the knowledge representation formalism adopted (e.g. RDF(S) language in our case).

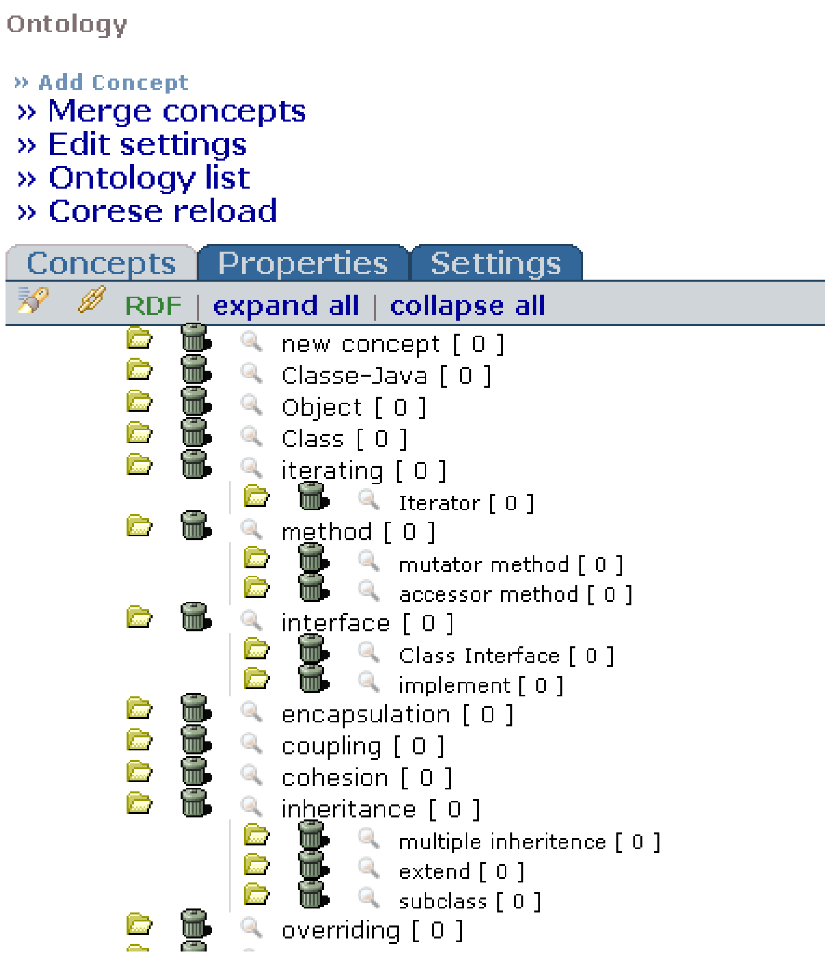
These stages in the process may be undertaken together or separately. And the role of the ontology creation service is to provide support for their achievement.

2.1.2 Functionalities

We may therefore envisage the following functionalities of the service:

- supporting linguistic analysis of knowledge sources, extracting candidate terms (e.g. nominal syntagms and verbal syntagms) from which concepts and relations would be determined,
- ontology edition through friendly interfaces for ontology developer (either an ontologist or a CoP member): the ontology editor would produce a formal ontology and offer ontology visualization services (see Figure 2.2)
- exploiting existing ontologies in order to import them or to integrate several of them into the new ontology being built.

Moreover, the ontology creation service may include or call a verification service for checking the consistency and the coherence of the ontology.


Ontology

» Add Concept
» Merge concepts
» Edit settings
» Ontology list
» Corese reload

Concepts | Properties | Settings

RDF | expand all | collapse all

- new concept [0]
- Classe-Java [0]
- Object [0]
- Class [0]
- iterating [0]
  - Iterator [0]
- method [0]
  - mutator method [0]
  - accessor method [0]
- interface [0]
  - Class Interface [0]
  - implement [0]
- encapsulation [0]
- coupling [0]
- cohesion [0]
- inheritance [0]
  - multiple inheritence [0]
  - extend [0]
  - subclass [0]
- overriding [0]

Figure 2.2: Ontology Editing Interface

Inputs and Outputs of the service

Input

The ontology creation service has as input the knowledge sources used to create the ontology. These knowledge sources can be of different types, such as documents, existing ontologies...

Output

The ontology creation service produces an ontology represented in the knowledge representation formalism chosen (i.e. RDF(S) in our case).

Interfaces to other services

- The Knowledge Creation service may rely on the Knowledge Evaluation service for validating the elements of ontology (i.e. human validation by a human specialist domain of the domain, or verification of the ontology consistency by the system)..
- The Knowledge Evolution service can be used when the creation process uses only ontologies as knowledge sources.
- The Annotation service may appeal to the ontology creation service in the case of manual or semi-automatic annotation, when the user's proposed annotation does not correspond to an element of the ontology. In this case, it may mean that the ontology needs to be enriched by relevant concepts or relations.
- The Knowledge Visualization service for visualizing the ontology with various presentations according to the user.

Interface for Human-Computer interaction

An ontology editor allows the user to introduce concepts and relations but also to have one (or several) global view(s) on the object he/she is about to create. See for instance the ontology editing interface in the Figure 2.2.

Moreover, it must be noted that the ontology builder may not only be an ontologist but a member of the CoP (i.e. he/she may be not specialist in knowledge modeling or in knowledge representation languages). Therefore the interfaces should be friendly enough for to hide the complexity of the ontology and the knowledge representation language.

2.1.3 Examples of use in CoPs

Within the framework of the CoP "Telecom-INT - UX11 Module", the introduction of a new unit to the course (such as Linux kernel programming) which corresponds to a sub-domain (system programming), may require the creation of an ontology of this sub-domain, which could subsequently make possible the annotation of documents introduced in the system in this section of the course. In this case, an ontology of the sub-domain needs to be created and integrated to the existing ontology (via the evolution and maintenance services).

2.2 Annotation

Let us give a few definitions:

Indexation of a textual document consists of locating in this document some words or expressions considered as significant (called terms) in a given context, and of creating a link between these terms and the original text. [Wikipedia]

Metadata is a data about a data (it can be in a paper-based form or in an electronic form).

Annotation is additional information associated to a resource (document, element of document). It can describe as well a comment on the resource, an interpretation on its semantic contents, or metadata (e.g. in the way of the Dublin Core) enabling to index this resource.

Semantic (or Ontological) Annotation is an annotation based on an ontology.

In the context of Semantic Web, annotations must be:

(a) Formal in order to be handled by programs and

(b) Understandable by humans in order to be validated and used by them.

The **Annotation service** to be offered in Palette aims at:

- Allowing the CoP's members to attach opinions, comments or assessments to resources of the CoP;
- Generating corresponding semantic annotations with regards to the CoP-dependent ontology, so as to enable programs to reason on these annotations.

Hence, these cooperative annotations and assessments provide the CoP with more knowledge about the annotated resources and allow to externalize knowledge since the individual comments, made by a member of the CoP on a resource, can thus be shared with the whole CoP.

Therefore, this awareness of the others comments on a resource builds a framework for encouraging interactions among the CoP's members. Then, this Cooperative Annotation service can be considered as a specific kind of "Cooperative Knowledge Creation".

The Annotation service constitutes a support to other services such as "Knowledge Retrieval and Dissemination". In this case, the relevance of a document, for example, can be evaluated according to its semantic annotation.

An annotation is characterized by:

- Its author;
- The addressees of the annotation;
- The form of the annotation (text, graphical, video format, RDF, etc.);

- The annotated resource and its nature (textual document, mail, forum discussion, even an annotation);
- The content of the annotation.

Therefore, the Annotation service must enable to capture and manage these different kinds of information at least.

2.2.1 Functionalities

Manual annotation

An annotation editor can provide forms that can be predefined or dynamically generated, on the basis of the CoP-dependent ontology. This assumes that the elements/parts of the ontology that are relevant for the annotation have already been identified.

Semi-automatic annotation

The service will use natural language processing tools such as term extractor (enabling to recognize in texts terms associated to the ontology concepts) and relation extractor (enabling to recognize in texts relations of the ontology), so as to generate the corresponding annotations. It can use relation patterns to help determine the relevant relations in the resources to be annotated, as well as the arguments of these relations, in order to provide annotations with related to the CoP-dependent ontology.

For validation purposes, the user must be involved in the process of the semi-automatic annotation.

Collaborative Annotation

The Collaborative Annotation service may use natural language processing tools to assist the user in producing free-text annotations, suggesting him the instances according to the other members (their annotations on the resource), the resource to be annotated (for example, it would be relevant to annotate a scientific article according to the domain it deals with; it would be interesting to annotate a discussion on a forum it according to the domain as well as the arguments and positions mentioned).

In the case of collaborative annotation, the Annotation service should support user authentication (in case where annotating would be restricted to particular members of the CoP) and users profiles, thus allowing to know, the identity of the author of each annotation. This might be the starting point for a discussion or a debate between the members, as the members could see the others' annotations and send them a notification or a message so as to suggest them a discussion on the annotated resource/topic. This latter point constitutes a specific kind of "Cooperative Knowledge Creation".

2.2.2 Interfaces of the service

In all cases, the Annotation service needs to have as:

Input

The resources to be annotated;

Output

The obtained annotations (textual annotation plus formal semantic annotation represented in RDF);

Interface to human

The interface for a human (see Figure 2.3) will be a graphical interface consisting of forms to be filled and enabling to indicate the resource to be annotated, as well as to give the additional (previously mentioned) information (the addressees of the annotation, its content, etc.).

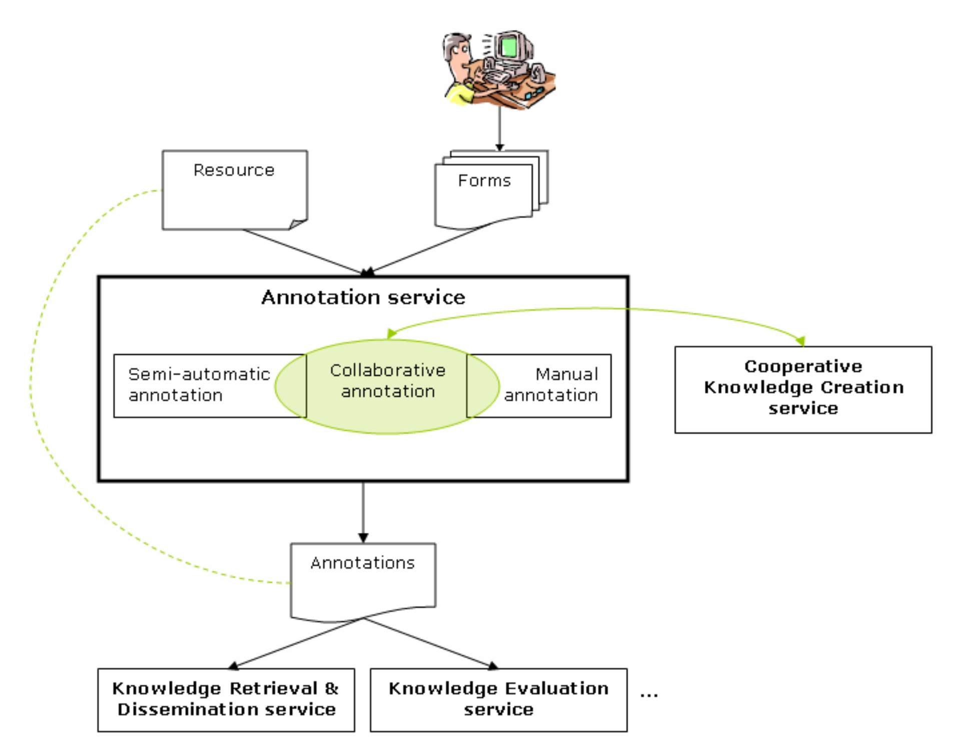

Resource
Forms
Annotation service
Semi-automatic annotation
Collaborative annotation
Manual annotation
Cooperative Knowledge Creation service
Annotations
Knowledge Retrieval & Dissemination service
Knowledge Evaluation service
...

Figure 2.3: Interfaces of the Annotation service

Interfaces to other services

- The annotations created by the Annotation service can be accessed and used by the "Knowledge Retrieval and Dissemination service".
- The "Cooperative Knowledge Creation service": as mentioned above, the "cooperative" Annotation service represents a specific case of this service.
- The "Knowledge Evaluation service" can use the Annotation service in order to get feedback from the CoP members, if a feedback can be expressed through an annotation.
- Other services, such as Inference services or Clustering services could also rely on the annotations generated through the Annotation service.

2.2.3 Examples of use in CoPs

Let's take the "Telecom-INT - UX11 Module" (engineer-students) as an example of CoP (see Table 2.1).

Environment/situation	Making choices about the practical trainings; During the practical trainings: interactions to share knowledge, solve problems, debate, etc.
Practice/ activity	System installation; System administration
Actors	Students; Teachers; Tutors (at the training site, enterprise); Colleagues at the training site.
Resources	A list of the training proposals; Student profiles; Tutor profiles; Teacher profiles; Courses of C language, Unix, etc. Requests for trainings.

Table 2.1: Annotation scenario in "Telecom-INT - UX11 Module"

In this CoP, the Annotation service can be invoked for multiple purposes: the annotation of courses, requests for trainings, training proposals, discussions and debates, etc.

2.3 Cooperative Knowledge Creation

2.3.1 Functionalities

This service aims to support collaborative problem solving. To allow this, the service must enable:

- Identification of the author (user) of a proposal or an argument;
- Support to discussion and argumentation (link with the related Mediation services):
- Filtering according to the user (to provide and keep track on the user's participation throughout the discussion);

This service can be applied to:

- Collective ontology creation;
- Collective annotation creation;
- Collective problem solving or collective case resolution.

2.3.2 Interfaces of the service

Interface to human

The service could provide :

Free discussion support: in this case, there must be a person (with a particular role in the CoP) or a tool that will analyze the content and issues of the discussion;

Structured discussion support, which supports reasoning mechanisms (we can mention in this context the QOC - Question, Option, Criteria - formalism for the decision making process);

White board, that enables the members' cooperation by means of a virtual support that they can manipulate synchronously.

Among these options and supports to discussion, the second one seems to be the most realistic and the most likely to be implemented in the context of the Palette project.

Interfaces to other services

This service produces knowledge resources that will be annotated by the "Annotation service".

Chapter 3

Knowledge Retrieval and Dissemination

3.1 Definition

This service aims at allowing the CoP's members to access resources which constitute an interest, considering their content, annotations and the members' profiles. The Knowledge Retrieval requires the active participation of the members, since they have to formulate and submit a query so as to get the relevant resources in response; while Knowledge Dissemination does not require the members' explicit participation, because it consists of transmitting notifications or resources to the members who may be interested in.

The other difference between Retrieval and Dissemination is that the first provides knowledge in response to ad hoc informational needs, while the second disseminates knowledge which may interest a member, according to his interests/profile (which constitutes his interests, that is his/her stable/long-term needs).

3.2 Knowledge Retrieval

This service is characterized by the following elements:

- The user who performs the search;
- The user's query, which corresponds to the information need;
- The results/answers to the query, in the form of resources or resources annotations.

3.2.1 Functionalities

This service, that offers semantic search, can be decomposed into:

Query formulation, for which the service provides an interface that:

- Facilitates the query formulation by guiding the user, allowing him/her to navigate through the concepts of the CoP-dependent ontology. In addition to the fact that this will provide a semantic search, we can assume that, depending on the part of the ontology that will be used to formulate the query, it will also restrict the search space (for example, the kind of resources to query). For example, if the query is about finding particular competencies, then the retrieval process will consider the CVs of the CoP's members, if available, or their profiles;

- Allows the user to introduce elements of annotations in his query.

Query processing, in which the service uses:

- The ontology and the annotations to determine the list of relevant resources matching the user's query (possibly after an ontology-based reasoning);
- The user's profile to constitute and enrich the list of relevant resources matching his/her query and interests.

Answers/results ranking, in which the service uses:

- The annotations in order to re-rank the relevant resources for a query;
- The user's profile to re-rank the relevant resources with regards to his/her interests.

Answers/results presentation, where the service interacts with the "Knowledge Presentation and Visualization service" (see Chapter 4).

3.2.2 Interfaces of the service

As shown in Figure 3.1, the Knowledge Retrieval service needs to have as:

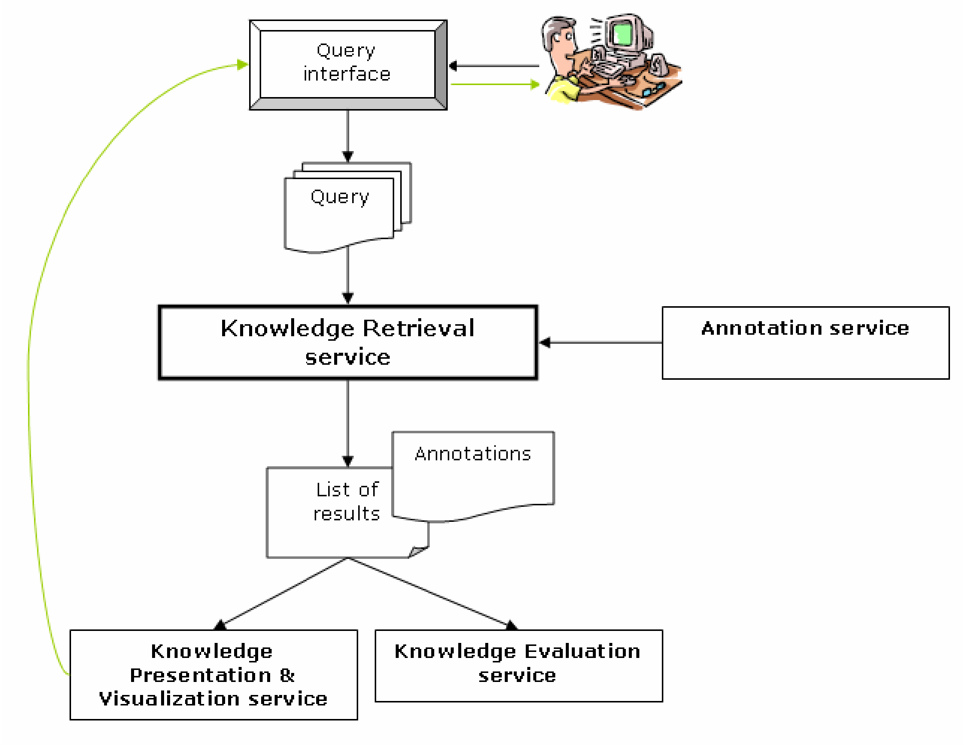

Query interface
Query
Knowledge Retrieval service
Annotation service
Annotations
List of results
Knowledge Presentation & Visualization service
Knowledge Evaluation service

Figure 3.1: Interfaces of the Knowledge Retrieval service

Input

A query with possibly sorting parameters;

Output

The list of resources found as results to the query, together with, their annotations;

Interface to human

An interface to the query formulation (see 3.2.1 - Query formulation);

Interfaces to other services

- As said before, the Knowledge Retrieval service uses the annotations created by the "Annotation service";
- The results provided by the Knowledge Retrieval service can be evaluated through the "Knowledge Evaluation service";
- These results can also be presented through the "Presentation and Visualization service".

Resources used:

- The user's profile;
- The ontology;
- The annotation base;
- The knowledge resources used in the CoP (documents, mails, forums, etc.).

3.2.3 Examples of use in CoPs

Let's consider the "Telecom-INT - UX11 Module" CoP (engineer-students).

Use case 1

In this example, we assume that a member of the CoP is seeking for a practical training. The Knowledge Retrieval service provides him/her with an interface through which he/she formulates his/her query, following the scenario proposed in Table 3.1.

Query/input	A set of subjects, place, a list of the related professions, and eventually the teacher and the tutor.

Resources	The student's profile;
	The ontology;
	The annotation base;
	The trainings proposals.

Table 3.1: Example of Knowledge Retrieval in "Telecom-INT - UX11 Module"

The purpose is to supply the student with training proposals which correspond to his/her query. Therefore, we propose the annotations presented in Table 3.2.

Resources	Semantic annotations on the resources
Trainings proposals	Subject, place, tutor, the related profession, the required competencies.
Students profiles	Static information about the student; A list of courses (which he/she attends); A list of the tutors he/she had, a list of the teachers who supervised him/hers; A list of the professions he/she is interested in.

Table 3.2: Annotations for trainings (following the above example)

Based on these annotations, we propose the following scenario (Figure 3.2) on filtering the training proposals according to the request of the student.

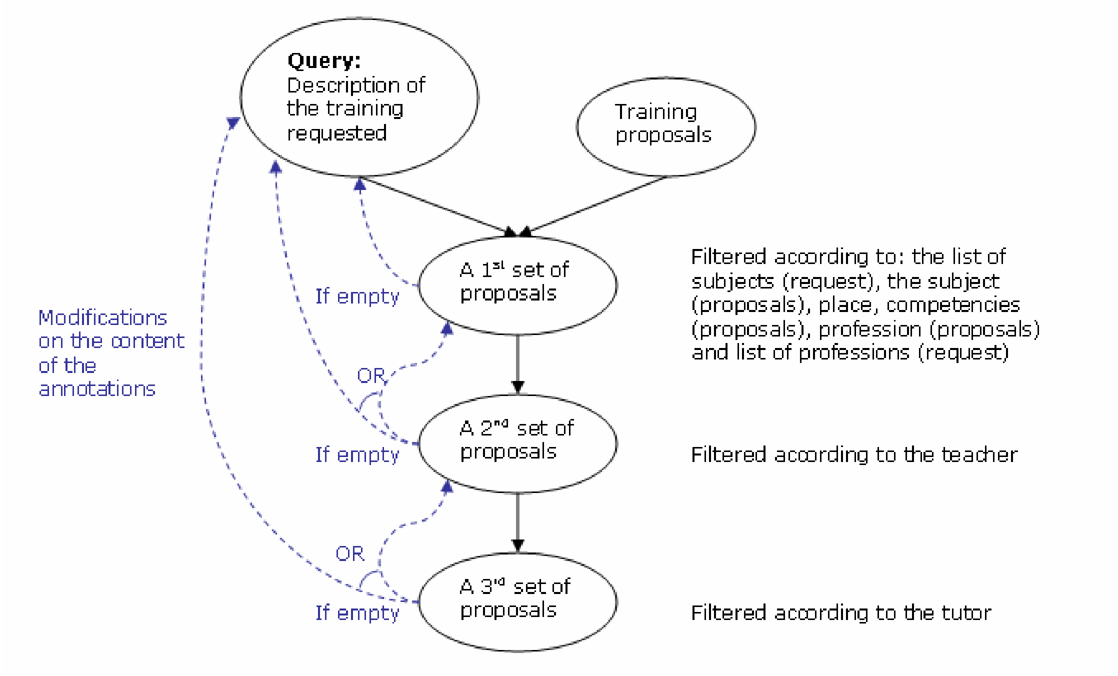

Query:
Description of the training requested
Training proposals
A 1st set of proposals
Filtered according to: the list of subjects (request), the subject (proposals), place, competencies (proposals), profession (proposals) and list of professions (request)
If empty
Modifications on the content of the annotations
OR
A 2nd set of proposals
If empty
Filtered according to the teacher
OR
A 3rd set of proposals
If empty
Filtered according to the tutor

Figure 3.2: Knowledge retrieval scenario

If at a level of filtering, the set of resulting proposals is empty, it means that there is no exact match. In this case, the content of the annotations should be modified (approximated or enlarged). For example, if the first set of proposals is empty, there should be a change on the user's query: there should be an enrichment on the "set of subjects" and the "list of professions" (using the ontology and the annotation base, or requesting the user).

We mention that the validity of annotations should be checked. For example, the “competencies” required for a training (“training proposal”) can be linked to the “subject” of the student’s query.

Use case 2:

In this example, we assume that a member of the CoP is seeking for “relevant surveys about the C programming language”.

Query/input	Survey, programming language: C.

Resources	The student’s profile;
	The ontology;
	The annotation base;
	The courses.

Table 3.3: Example of Knowledge Retrieval in “Telecom-INT - UX11 Module”

Assume that there are three documents: D1, D2 and D3, with the annotations presented in Table 3.4.

Doc	**Annotations with the ontology concepts**	**Textual annotations**	**Semantic annotations**
D1	Programming language: C	Good survey	[Survey:D1]-(evaluated)-[Literal:good]-(subject)-[Programming-language:C]
D2	Programming language: C Survey		[Survey:D2]-(subject)-[Programming-language:C]
D3	Object languages	Prerequisites for a survey about the C language	[Document:D3]-(subject)-[Object-languages]-(prerequisites)-[Survey]-(subject)-[Programming-language:C]

Table 3.4: Annotations of the documents

Without the annotations expressing comments or assessments, the results to the query would be D1 and D2 (with D2 being considered more relevant than D1).

But, considering the semantic annotations of the three documents, which include the comments, and depending on the importance given to the annotations that express assessments and comments, the score of D1 might increase so much that it would become higher than that of D2.

As for D3, if we introduce these annotations, then it should appear in the list of the results to the user’s query. Of course, this document shouldn’t appear at the top level of the list, but as it is somehow relevant to the user’s query, it should be considered as a potential result.

3.3 Knowledge Dissemination

3.3.1 Functionalities

This service uses the annotations and the member's profile to capture and disseminate relevant resources or send notifications to him/her;

It is characterized by:

- The user to which knowledge must be disseminated;
- The resource(s) to be disseminated;
- The moment of the dissemination (e.g. the event that triggers the dissemination).

3.3.2 Interfaces of the service

Input

An event or a particular situation triggering the dissemination.

Output

The list (or a notification or the resource itself) of new or modified resource(s) that match the interests of a particular member (or group of members) and their annotations, together with, the member(s) concerned.

Interfaces to other services

As said before, the Knowledge Dissemination service uses:

- The annotations created by the "Annotation service";
- The resources provided by the Knowledge Dissemination service can be evaluated through the "Knowledge Evaluation service";
- These resources can also be presented through the "Presentation and Visualization service".

Resources used

- The user's profile;
- The ontology;
- The annotation base;
- The new or modified knowledge resources used in the CoP (documents, mails, etc.) together with their annotations (can include its addressees, which constitutes then a dissemination criterion).
- Initially, a CoP member should use an appropriately designed form to describe his/her profile, the topics or resources he/she is interested in, and when he/she wants to receive alerts about new (or modified) resources, or the new resources directly. These elements rather constitute the resources used by the service than its input.

The interfaces of the Knowledge Dissemination service are summarized in Figure 3.3.

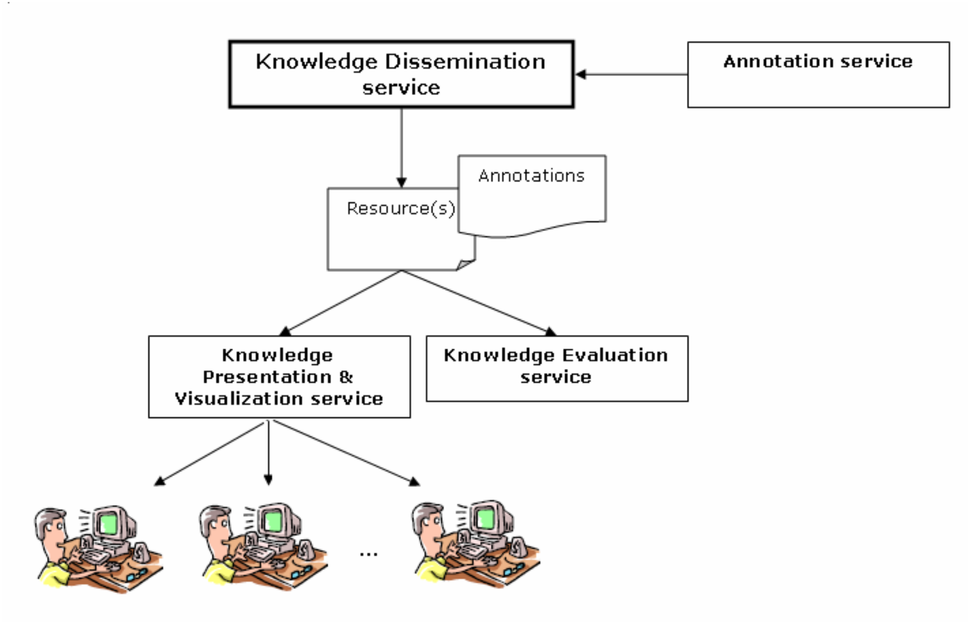

Knowledge Dissemination service
Annotation service
Annotations
Resource(s)
Knowledge Presentation & Visualization service
Knowledge Evaluation service
...

Figure 3.3: Interfaces of the Knowledge Dissemination service

3.3.3 Examples of use in CoPs

In the case of the “Telecom-INT - UX11 Module” CoP (engineer-students), if a new training proposal is posted, then all the CoP’s members which are interested in its subject, its tutor, its related profession or its required competencies (these elements annotate the training proposal), should be made aware of it, so that they can apply for it. The content of the proposal, which is used to index it, can also be used to capture the relevant terms and concepts describing the training proposal.

Chapter 4

Knowledge Presentation and Visualization

The Presentation and Visualization service is in charge of presenting the knowledge to end users. This presentation can take different forms depending on the types of knowledge and resources involved.

4.1 Functionalities

When a user receives the result of a query or browses a knowledge base (the annotations base or a resources base) the interface requirements depend on:

- the type of search the user is performing (e.g. looking for a particular item vs. looking for an overview of the contributions in a domain);
- the type of task that triggered the search (e.g. learning on a subject, writing a report, comparing opinions);
- the type of items that is being searched (e.g. searching for documents, for persons, etc.);
- the profile of the user (e.g. expert of a domain, random web surfer);
- the device of visualization (e.g. on a computer, on a PDA, on a cellular phone);
- etc.

Therefore, more generally speaking and depending on the context, the visualization of a piece of knowledge will require different types of interfaces providing different views, different interaction means and different customization capabilities.

The idea then is to rely on an open pool of visualization services and widgets among which users can choose the most appropriate one. Some of these services may be extremely simple (e.g. generate an HTML table of the answers to a query) and some may be more complex (e.g. generate a graphical view of the classes of answers to a query) and require a post-processing of the results to be finally visualized (e.g. clustering algorithm).

4.2 Interfaces of the service

Input

The inputs will most probably vary with the complexity of the service and some of them will be fetched by the service itself depending on the data it has to display (e.g. ontologies referenced in a result). We can distinguish three types of inputs:

(a) the content to be visualized: RDF chunk or XML SPARQL Binding to be displayed

(b) the specific parameters of the service (e.g. preferred language, user profile, etc.)

(c) the resources needed to generate the view and that will be fetched at runtime, depending on the actual content to be visualized (e.g. ontologies, pictures, etc.).

Output

The output is a web-based interface rendering the result.

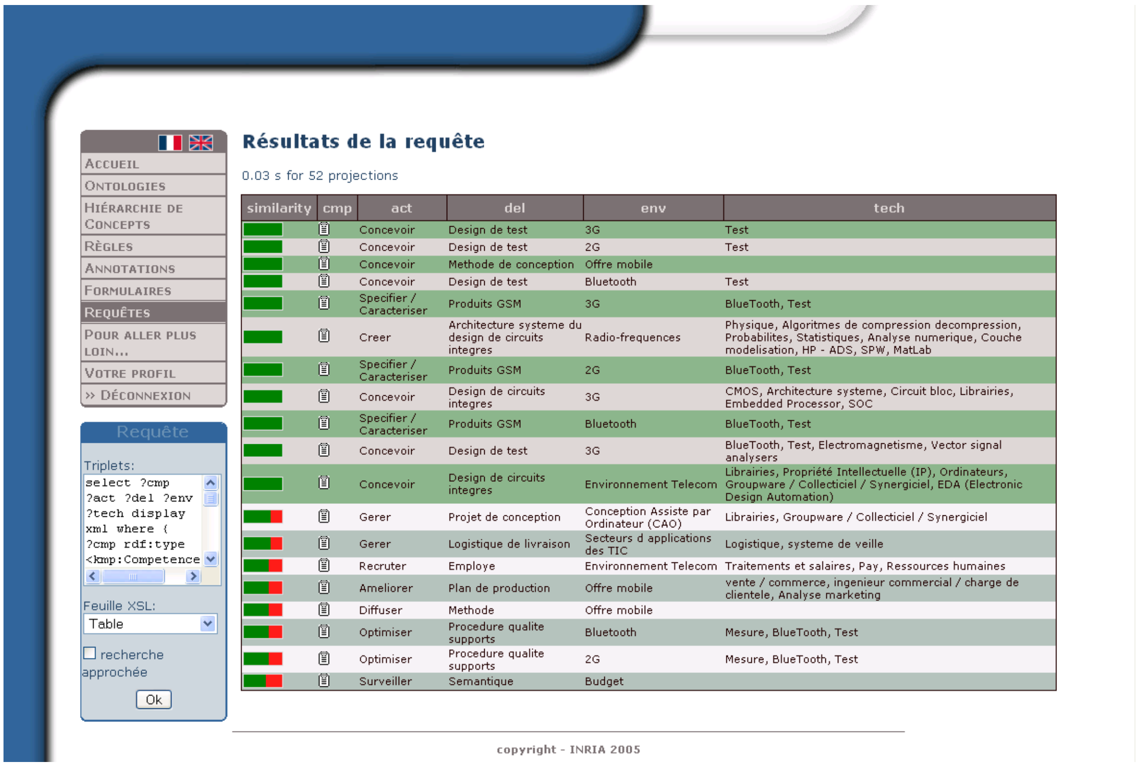
4.3 Examples of use

The Visualization service provides CoPs users with an interface to access the resources handled by the KM system. In this respect, it is the access point to knowledge in the system. The users can access to these resources/knowledge items either by **(i)** nequesting the knowledge base, or by **(ii)** navigating in it.

Here we give a few examples of different services applied to display a result of a query. These examples have been implemented using SeWeSe (see Chapter 9).

Query results presentation

The example shown in Figure 4.1 presents the result of a query in an HTML table. Another possible presentation is to give a graphical view of the results of a query, as depicted in Figure 4.2.



```
select ?cmp
?act ?del ?env
?tech display
xml where (
?cmp rdf:type
<kmp:Competence
```

Résultats de la requête

0.03 s for 52 projections

similarity	cmp	act	del	env	tech
🟩	📋	Concevoir	Design de test	3G	Test
🟩	📋	Concevoir	Design de test	2G	Test
🟩	📋	Concevoir	Methode de conception	Offre mobile	
🟩	📋	Concevoir	Design de test	Bluetooth	Test
🟩	📋	Specifier / Caracteriser	Produits GSM	3G	BlueTooth, Test
🟩	📋	Creer	Architecture systeme du design de circuits integres	Radio-frequences	Physique, Algoritmes de compression decompression, Probabilites, Statistiques, Analyse numerique, Couche modelisation, HP - ADS, SPW, MatLab
🟩	📋	Specifier / Caracteriser	Produits GSM	2G	BlueTooth, Test
🟩	📋	Concevoir	Design de circuits integres	3G	CMOS, Architecture systeme, Circuit bloc, Librairies, Embedded Processor, SOC
🟩	📋	Specifier / Caracteriser	Produits GSM	Bluetooth	BlueTooth, Test
🟩	📋	Concevoir	Design de test	3G	BlueTooth, Test, Electromagnetisme, Vector signal analysers
🟩	📋	Concevoir	Design de circuits integres	Environnement Telecom	Librairies, Propriété Intellectuelle (IP), Ordinateurs, Groupware / Collecticiel / Synergiciel, EDA (Electronic Design Automation)
🟩🟥	📋	Gerer	Projet de conception	Conception Assiste par Ordinateur (CAO)	Librairies, Groupware / Collecticiel / Synergiciel
🟩🟥	📋	Gerer	Logistique de livraison	Secteurs d applications des TIC	Logistique, systeme de veille
🟩🟥	📋	Recruter	Employe	Environnement Telecom	Traitements et salaires, Pay, Ressources humaines
🟩🟥	📋	Ameliorer	Plan de production	Offre mobile	vente / commerce, ingenieur commercial / charge de clientele, Analyse marketing
🟩🟥	📋	Diffuser	Methode	Offre mobile	
🟩🟥	📋	Optimiser	Procedure qualite supports	Bluetooth	Mesure, BlueTooth, Test
🟩🟥	📋	Optimiser	Procedure qualite supports	2G	Mesure, BlueTooth, Test
🟩🟥	📋	Surveiller	Semantique	Budget	

copyright - INRIA 2005

Figure 4.1: Result of a query

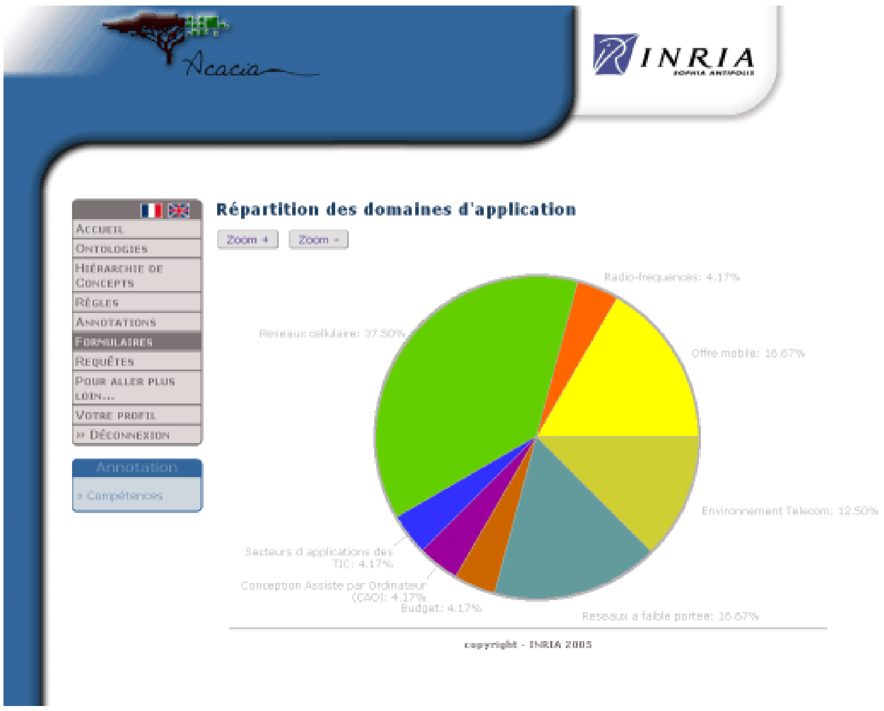
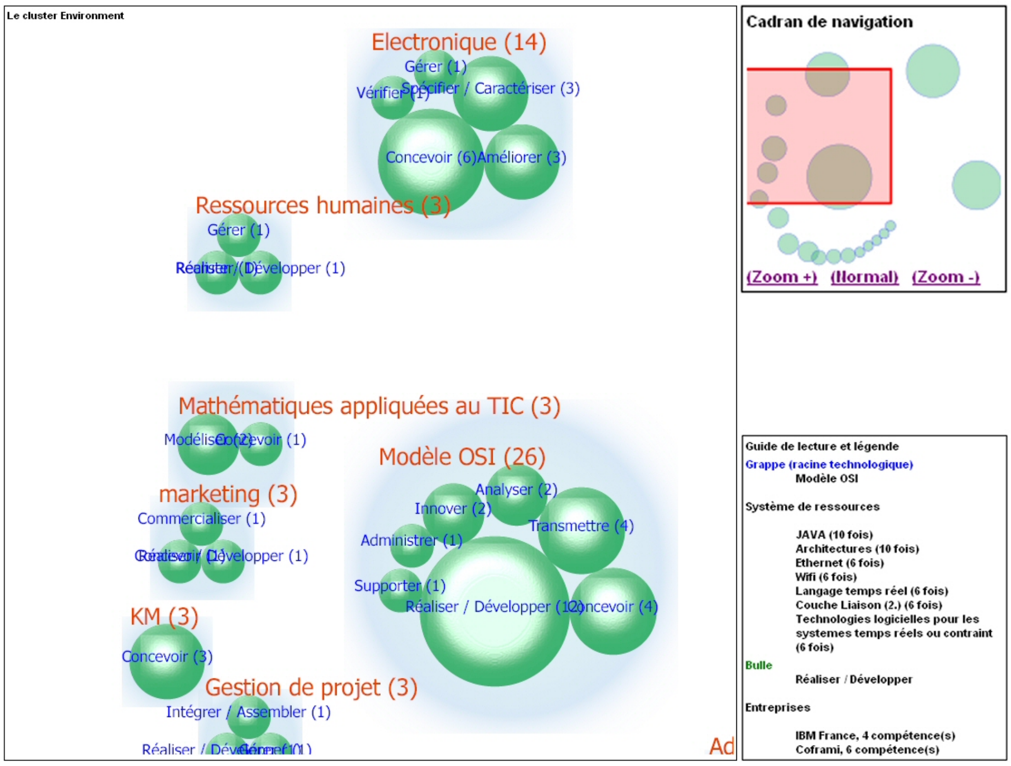


Figure 4.2: Graphical view of the results of a query

Navigation

Another possible scenario for the visualization service is the navigation in the annotations base. For example if the user wants to search visually in all the existing resources, rather than formulate a specific query. The service should be able to present a general view of these resources, using some complex visualization operations, such as clustering illustrated (see Figure 4.3). It could enable to zoom on some part of the resources.



Guide de lecture et légende

Grappe (racine technologique)

Modèle OSI

Système de ressources

JAVA (10 fois)
Architectures (10 fois)
Ethernet (6 fois)
Wifi (6 fois)
Langage temps réel (6 fois)
Couche Liaison (2.) (6 fois)
Technologies logicielles pour les systemes temps réels ou contraint (6 fois)

Bulle

Réaliser / Développer

Entreprises

IBM France, 4 compétence(s)
Coframi, 6 compétence(s)

Figure 4.3: Example of clustering

4.3.1 Example of use in CoPs

In the case of the "Telecom-INT - UX11 Module" CoP (engineer-students), the visualization service is involved as shown in Figure 4.4.

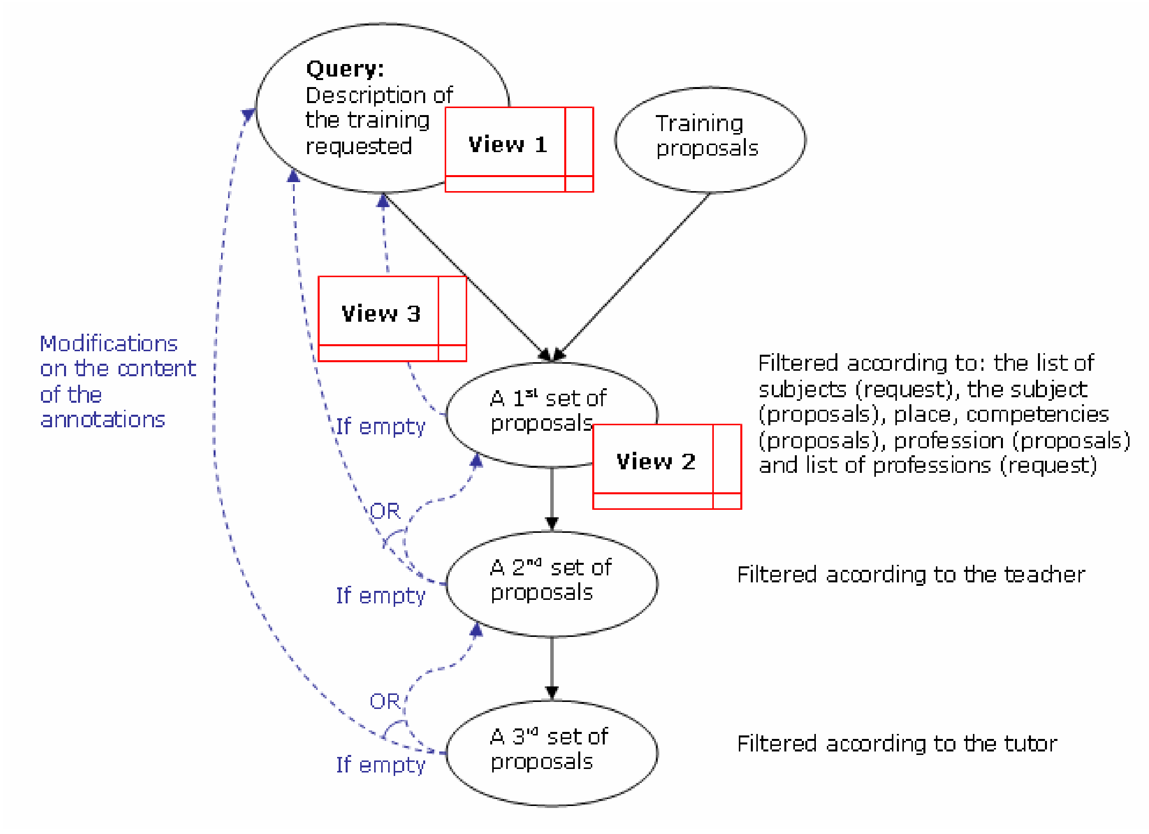

Query:
Description of the training requested
View 1
Training proposals
View 3
Modifications on the content of the annotations
A 1st set of proposals
View 2
Filtered according to: the list of subjects (request), the subject (proposals), place, competencies (proposals), profession (proposals) and list of professions (request)
If empty
OR
A 2nd set of proposals
Filtered according to the teacher
If empty
OR
A 3rd set of proposals
Filtered according to the tutor
If empty

Figure 4.4: Knowledge visualization scenario

Chapter 5

Knowledge Evaluation

The purpose of building a Knowledge Evaluation service in Palette is to provide support for the evaluation of CoP tools and resources, such as the CoP-dependent ontology, the annotations, the documents used or produced by the CoP (e.g. the lessons learnt), etc.

The Knowledge Evaluation service is characterized by:

- The resource to be evaluated;
- The tool to be evaluated;
- The type of evaluation performed (automatic, semi-automatic, manual);
- The evaluation criteria (for a resource: correctness, usefulness, quality, etc.; for a service: performance criteria, such as efficiency);
- The evaluation grid through which the evaluation is made explicit.

5.1 Functionalities

Given a particular resource, this service aims to provide an evaluation grid to be filled either by:

- CoP member(s) who give their feedback concerning a resource: this corresponds to the manual evaluation;
- An automated module of the service itself: in this case, the evaluation grid is filled according to the knowledge conveyed by the CoP-dependent ontology and the annotations, and using reasoning mechanisms. This corresponds to the automatic evaluation;
- Semi-automatically: in this case, the Evaluation service uses its reasoning mechanisms to produce an evaluation, which will be then checked for validation or completion by a particular member of the CoP.

5.2 Interfaces of the service

In all cases, the Knowledge Evaluation service needs to have as:

Input

- The resource to be evaluated or the necessary information to evaluate a tool (depending on what is to be evaluated);
- The evaluation grid, which contains the evaluation criteria;

- The information provided by CoP members, in case they contribute to the evaluation (their feedback).

Output

The evaluation grid, which may be filled with:

- A binary information (yes/no, true/false);
- A qualitative result: formal annotations or free-text generated according to the CoP-dependent ontology and annotations;
- A quantitative result based on evaluation scores or statistics.

This service may also offer mechanisms to provide a support to comparative evaluation between multiple resources.

Interface to human

In the case of manual or semi-automatic evaluation, it will be a graphical interface enabling one to fill, validate or complete the evaluation grid (with evaluation feedbacks or evaluation criteria). It will also enable a CoP member to indicate the resource to be evaluated.

This interface may also consist of a form or a questionnaire to be filled by the members and then interpreted by the service.

Interface to other services

The Knowledge Evaluation service uses the knowledge provided by:

- **“Annotation service”** when validating the annotations made on a resource: to check whether the annotation is made with respect to the CoP-dependent ontology (automatic evaluation) or whether the resource is completely annotated. If not, the Knowledge Evaluation service identifies the missing annotations to be made (semi-automatic evaluation);
- **“Knowledge Creation service”** in this case, the evaluation concerns the CoP-dependent ontology, to be then evaluated by the Knowledge Evaluation service which will check for the coherence and the validity of its structure;
- **“Knowledge Retrieval and Dissemination service”** for diverse purposes, such as:
    - the evaluation of the retrieval efficiency according to the time of response to the user’s query, for example;
    - a preliminary evaluation of the resources retrieved (and presented to the user who submitted the query), based on the user’s profile, to determine his search preferences and the relevance threshold to respect when retrieving resources for him;
    - the evaluation of the retrieved or disseminated resources, using the Annotation service to create the annotations expressing the relevance judgment of the user who submitted the query (case of

the retrieval) or of the users to whom the resources are disseminated (case of the dissemination);

The interfaces of the Evolution service are summarized in the Figure 5.1.

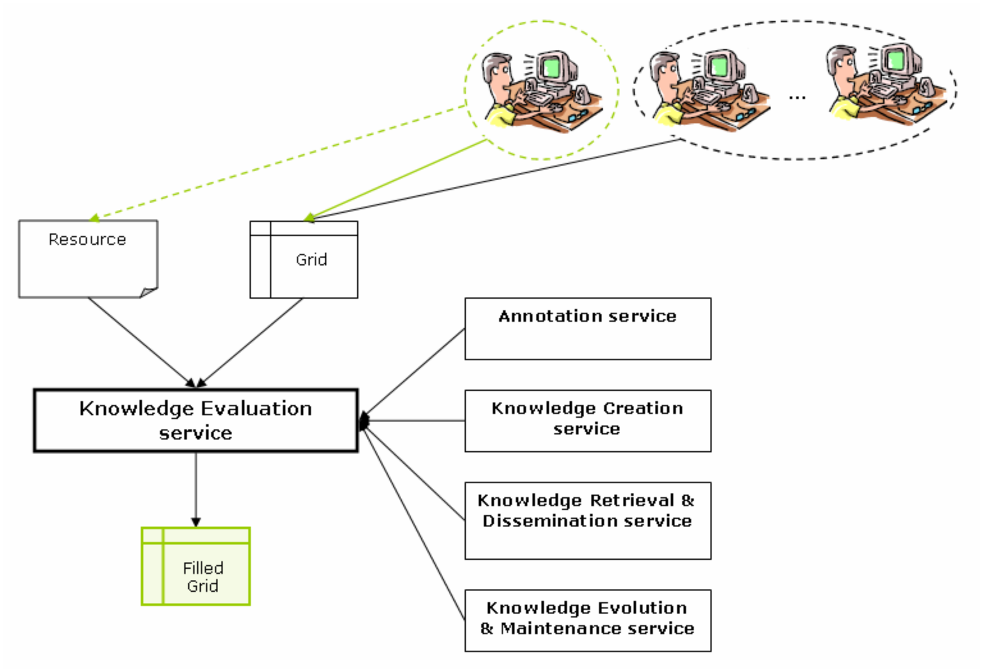

Resource
Grid
Annotation service
Knowledge Evaluation service
Knowledge Creation service
Knowledge Retrieval & Dissemination service
Knowledge Evolution & Maintenance service
Filled Grid
...

Figure 5.1: Evaluation service interfaces

5.3 Examples of use in CoPs

Let's take, as an example, the context of "lessons learnt" evaluation in a CoP. We describe, in the following, the scenario of evaluation with respect to the lessons learnt model developed in the *"D.KNO.01 CoP-independent meta-ontologies and support ontologies"* deliverable.

Figure 5.2 is the part of the lessons learnt model that we refer to.

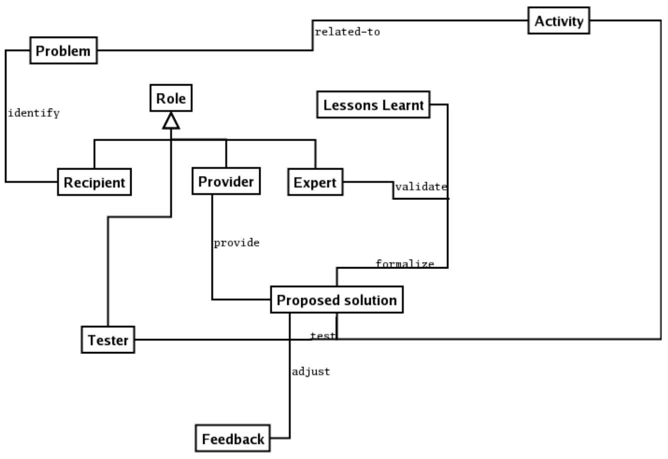


Figure 5.2: Lessons learnt model

The Tester proceeds to the experimentation of the proposed solutions to the problem and gives his/her feedback;

The Expert is in charge of assessing the proposed solutions, using his expertise on the domain and, at the same time, taking into account the feedback of the Testers.

The Knowledge Evaluation service plays the role of the Expert (see Table 5.1). More specifically, it:

- Fills the evaluation grid using the Testers' feedbacks;
- Completes this grid using its own expertise (the ontology, the annotations).

Inputs	Proposed solution(s); - Testers' feedback.
Resources used	Problem description; - Domain elements of expertise (ontology, annotations); - Evaluation grid.
Outputs	Qualitative and quantitative evaluation (e.g. the percentage of Testers who give a good feedback on the lesson learnt); - Automatic binary evaluation (yes/no, true/false: e.g. the lesson learnt is added to the lessons learnt repository).

Table 5.1: Example of Knowledge Evaluation of lesson-learnt.

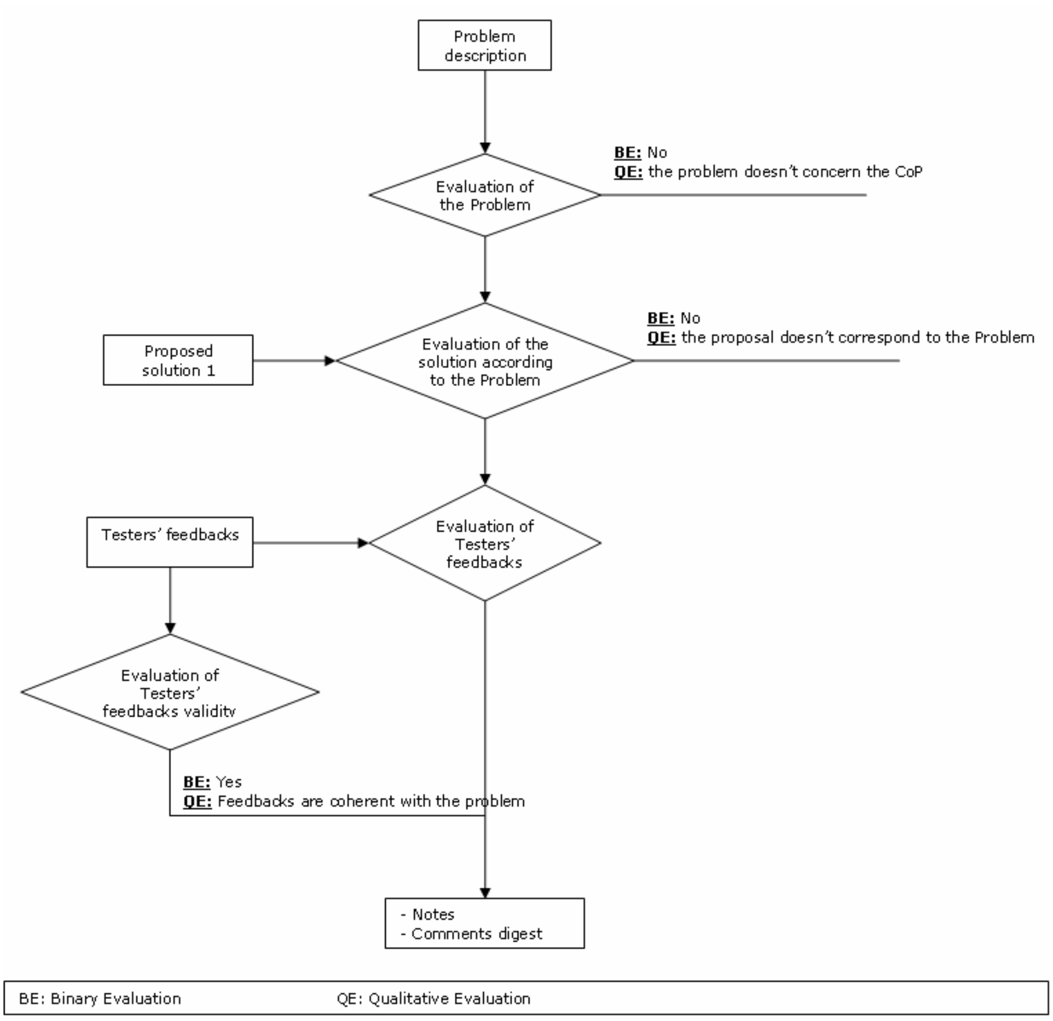

Problem description
Evaluation of the Problem
BE: No
QE: the problem doesn't concern the CoP
Proposed solution 1
Evaluation of the solution according to the Problem
BE: No
QE: the proposal doesn't correspond to the Problem
Testers' feedbacks
Evaluation of Testers' feedbacks
Evaluation of Testers' feedbacks validity
BE: Yes
QE: Feedbacks are coherent with the problem
- Notes
- Comments digest
BE: Binary Evaluation
QE: Qualitative Evaluation

Figure 5.3: Knowledge evaluation

The "Evaluation of the Testers' feedbacks" depends on the Tester: the evaluation note or score he/she puts on the proposed solution may be weighted according to the level of domain expertise of the Tester. This assumption depends on the considered CoP and its organization. Figure 5.3 show a detailed scenario of Knowledge evaluation.

Chapter 6

Knowledge Evolution and Maintenance

6.1 Definition

Knowledge evolution service consists of supporting the consistency of the CoP memory in case of evolution of CoP's knowledge resource: new document or document modified, new version of the ontology, new annotations or annotation modified, etc.

6.2 Functionalities

Regarding functionalities, we can envisage the following:

- Processing of the ontology evolution: in particular support to the CoP ontology manager in order to re-establish consistency of annotations influenced by the changed ontology; notification to the CoP members possibly concerned by this change according to their user profile;
- Processing of the annotation base evolution: in particular detection of possible inconsistencies generated from this change; notification to the CoP members possibly concerned by this change according to their user profile;
- Processing of the evolution (e.g. creation, modification, removal) of a document (e.g. report, mail, element of discussion in a forum) considered as a knowledge resource of the CoP: in particular warning about possible influence on its annotations; notification to the CoP members possibly concerned by this change according to their user profile;
- Processing of the CoP members evolution (i.e. the introduction, change of role, or removal of a CoP member, constitution of subgroups in the CoP, evolution of competences of a member, etc): influence on the annotations about these members, notification to the CoP members possibly concerned by this change according to their user profile.

6.3 Interface of the service

6.3.1 Inputs and Outputs of the service

Input

The element that evolved (e.g. ontology, annotation, document, CoP actor, etc.), and the description of its change (for example, its past state and its new state)

Output

The new state of the set of knowledge resources once the coherence restored, and the CoP members alerted of this evolution.

Interface in Human-Computer interaction

An editor for showing to the user the consequences of a change on the CoP knowledge resource and for helping to process the evolution, in particular in case of several alternatives to re-establish the coherence of the knowledge resources base (annotations, ontologies, documents, etc.)

Interfaces to other service

- The Knowledge Evolution service uses the annotations created by the "Annotation service", in particular the annotations on CoP members and their user profiles describing their interest in some resources;
- It could call the Knowledge Dissemination service in order to alert the CoP members influenced by the evolution of a given CoP knowledge resource (which can in turn be evaluated through the Knowledge Evaluation service);
- The support offered by the Knowledge Evolution service to the user can rely on the "Knowledge Retrieval" service in order to access the resources influenced by a change and it can rely on the "Presentation and Visualization service".

Resources used

- The user's profile;
- The ontology;
- The annotation base.

Figure 6.1 summarizes the interfaces of the evolution service.

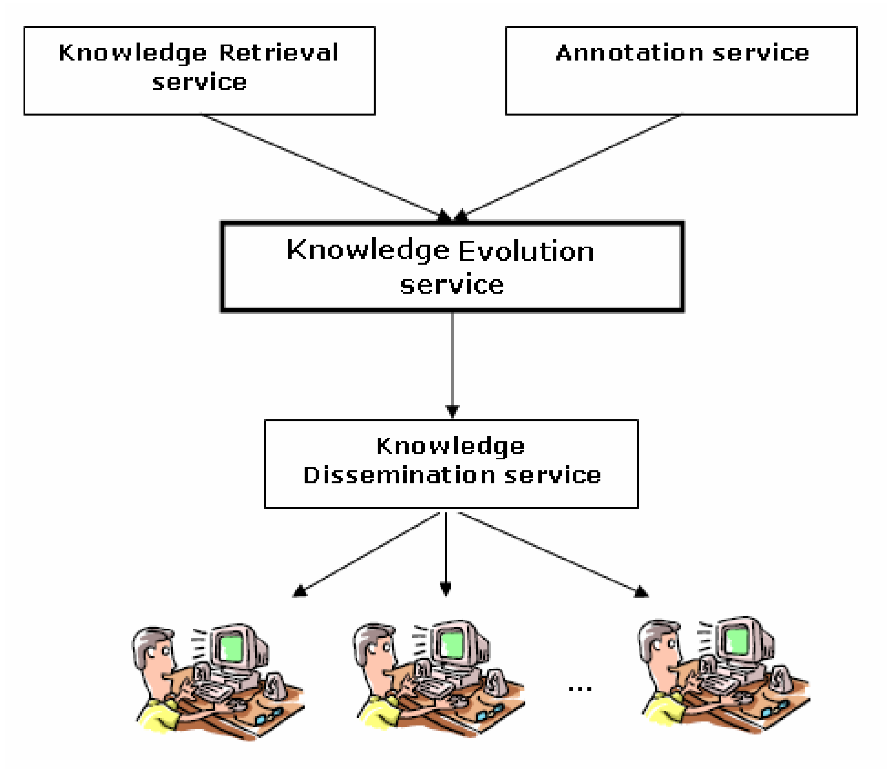

Knowledge Retrieval service
Annotation service
Knowledge Evolution service
Knowledge Dissemination service
...

Figure 6.1: Knowledge evolution interfaces

6.4 Examples of use in CoPs

In the context of "Telecom-INT - UX11 Module" (engineer-students), a new training proposal, a change of a student's tutor, the introduction of a teacher in charge of a new course with new documents dedicated to this course, and the modification of annotations on student profiles after the results of an exam, are examples of events that could trigger the knowledge evolution module.

Part II

Partners Tools offering building blocks services

This second part describes several tools available among the Palette partners. These tools which could offer such elementary services or more complex services (corresponding to a combination of the services mentioned above), are:

Generis, an ontology management tool (chapter 7) could offer ontology creation services,

Corese, a semantic search engine (chapter 8) could offer knowledge retrieval services,

SeWeSe, a platform for developing semantic web applications (chapter 9), could offer ontology creation, knowledge annotation and knowledge presentation or visualization services,

MEAT, a memory of experiments that contains an annotation tool (MeatAnnot) based on Natural Language Processing Tools (chapter 10), could offer services of semi-automatic annotation from texts,

Virtual Staff, a cooperative tool (chapter 11) could offer cooperative problem solving services as a specific kind of knowledge creation services.

Some of these tools are generic (e.g. Generis, Corese, SeWeSe), while others are so far dedicated to biomedical applications (MEAT, Virtual Staff) but could be adapted for other domains.

Chapter 7

Generis[4]

Generis is an ontology management tool which enables 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[1] and RDFS[2] 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 provided consisting in 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.

7.1 Architecture

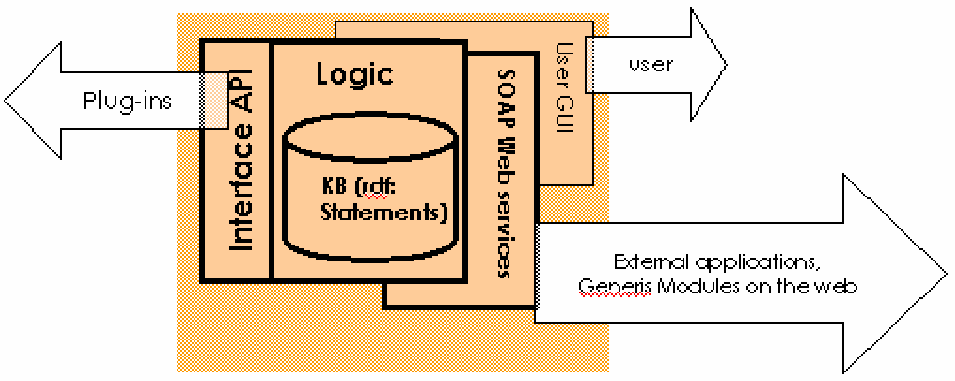


Figure 7.1: Generis Architecture

[1] Resource Description framework (http://www.w3.org/RDF/)

[2] Resource Description Framework Schema (http://www.w3.org/rdf-schema/)

Generis is built according to a traditional 3-tiers architecture (see Figure 7.1s) with a persistence layer, an application layer implemented in PHP[3] , and an interface layer (also in PHP). The interface layer consists in a Graphical User Interface (GUI), an Application Programming Interface (API), and a series of Web Services (WS). The database can be theoretically any relational database, using an abstraction layer. In practice, we built the kernel using MySQL while PostgreSQL and ORACLE has been tested and found compatible without problem. The application layer implements all the necessary data management methods, *i.e.* the RDF and RDFS associated methods plus some extra methods such as the structured search functionality, the communication services, and the administration of the kernel and user management. A basic GUI is proposed with the kernel enabling the manipulation of the model and the data, as well as the administration of the module. The kernel also proposes a series of services enabling the creation of more specific interfaces (either using directly the API when the interface is implemented in PHP, or using the WS via the SOAP[4] or XML-RPC[5] protocols when interfaces are written in JAVA or Flash).

7.2 Extension mechanisms

Several extensions mechanisms were implemented into Generis to face specific needs according to its use context. It is possible to specialize a Generis node in a specific domain by defining a model and to restrict the user to manage resources according to this higher level model (these resources should be protected because they fundamentally define the nature of the module). Nevertheless, it is still possible for the user to define his own lower level model to manage his resources. Such specialization of Generis can be completed by adding plug-ins which will provide the user with functionalities specific to the domain.

Plug-ins can be added to Generis to enable custom meta-data process automation or custom presentation to end user. Plug-ins use the API of Generis to retrieve, or store data. User-defined plug-ins can be integrated in any of the existing modules; provided that the specialized resources and their associated model where the plug-in relies on have also been imported in the module where the plug-in is installed. A plug-in can use all the methods present in the Kernel API. Hence, a plug-in can exploit the communication facilities and implement automatic treatment of data originating from different modules. This is possible only if the two following conditions are fulfilled: i) the local module where the plug-in is installed must be a valid subscriber of the external modules where the data

[3] PHP Home Page: (http://www.php.net)
[4] Simple Object Access Protocol (http://www.w3.org/TR/soap/)
[5] XML Remote Procedure Call (http://www.xmlrpc.com)

come from; ii) the specialized resources and ontologies the plug-in relies on must be present (at least the necessary part with respect to the treatment logics) in the distant modules. Plug-in installation is extremely simple. The PHP file containing the valid plug-in code must simply be placed in the ad hoc directory.

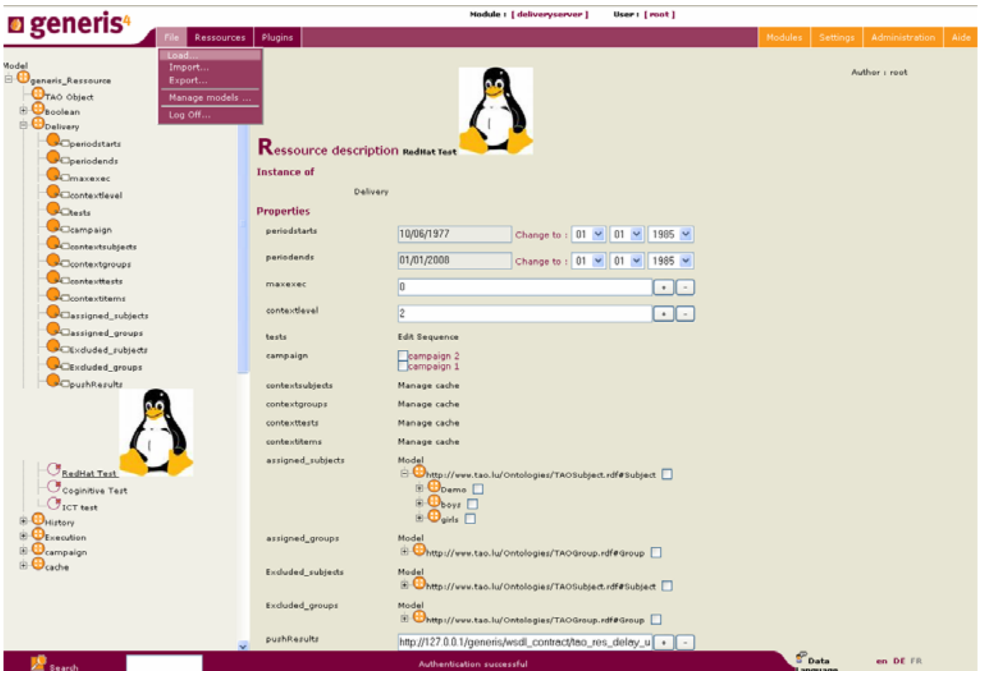
A plug-in is not necessarily bounded to a graphical interface or to a specific area such as the one used to display the resource structure for instance. However, most plug-ins use the GUI working area either for interactivity or for parameterization. Plug-ins can be as complex as little applications with many functions accessible through drop-down menus. Plug-in specific menus defined in the plug-in manifest will automatically appear in the first GUI area under the menu bar of the kernel general functions.

Currently, the default standard interface of a module includes three standard plug-ins. The first plug-in is used to access the resources. It updates the display of resource structure in the corresponding area of the interface, and uses the working area to execute interactively its basic functionalities: display resource content, manipulate resource structure, edit resources, search resources, create associations with other local or external resources (resources present in subscribed modules). The second standard plug-in enables the user personal data management (ID and password, access mask to be automatically applied on resources created by the user, e-mail, affiliation, etc.). The third standard plug-in is only accessible by the administrator and proposes several functions such as user, module subscriber, and module subscription management. The management of users is strictly local within a module. They are identified by an ID and a password and interact with the GUI and can, depending on their access privilege and the user group they belong to, read or create resource with different levels of detail). The module subscription mechanism does not allow altering distant (or non-local) resources.

The graphical aspect of the GUI can be modified using skins implemented as a CSS[6] file. These CSS should be implemented by the users and copied in the appropriate platform directory. Currently, there are no GUI functions enabling the dynamic skin management directly by the user. This should be added in further versions, as well as a basic CSS editor to help a user creating his own skin.

[6] Cascading Style Sheet (http://www.w3.org/Style/CSS/)

7.3 User interfaces and Functional coverage



Author : root

Ressource description RedHat Test

Instance of

Delivery

Properties

periodstarts	10/06/1977 Change to : 01 01 1985
periodends	01/01/2008 Change to : 01 01 1985
maxexec	0
contextlevel	2
tests	Edit Sequence
campaign	campaign 2 campaign 1
contextsubjects	Manage cache
contextgroups	Manage cache
contexttests	Manage cache
contextitems	Manage cache
assigned_subjects	Model http://www.tao.lu/Ontologies/TAOSubject.rdf#Subject Demo boys girls
assigned_groups	Model http://www.tao.lu/Ontologies/TAOGroup.rdf#Group
Excluded_subjects	Model http://www.tao.lu/Ontologies/TAOSubject.rdf#Subject
Excluded_groups	Model http://www.tao.lu/Ontologies/TAOGroup.rdf#Group
pushResults	http://127.0.0.1/generis/wsdl_contract/tao_res_delay_u

Figure 7.2: Annotation editor

The basic PHP/DHTML interface provided with the kernel is divided in four distinct areas. The first gathers the general functionalities and information: export, import, load, logout, connected user and access to functionalities (data manipulation). The second includes the data language management functionalities. Languages are managed on two levels: the language of the interface (as a general function), and the language of the resources. This later is directly managed with the ontology using language attribute associated to each resource. The management of the interface languages is made using a configuration file (Language Package) containing the label of the different terms appearing on the display. All alphabets and character sets can be used since the character encoding is UTF-8. Any number of language packages can be defined or personalized by the users. The two remaining areas are dedicated to the resource management. The third area consists in a graphical display of the resource structure stored in the database and the queries to obtain detailed information on these resources. By default, the resource structure is displayed as a tree to the user but can be easily changed or modified. Currently the tree is implemented in Javascript and runs on the client side. This implementation should soon disappear to eliminate problems when displaying very large structure and database contents. The fourth area is the working area where forms associated to resources management and queried data are displayed. Contextualized functionalities, i.e.

functionalities that only appear on the screen depending on the type of resource currently being focused on in the working area, are also displayed in this area. In some circumstances, some other windows (pop-ups) can be used temporarily.

Finally, a set of utilities is proposed to the user to facilitate the access to and the management of modules the user has a subscription to (they can be local or distant). To do so, user can access to an Internet portal with personal account where the different modules the user has an access to can be registered. This enables the user to navigate from one module to another without entering several times the different user ID's and passwords for each single module. In the personal module management portal, the user can select the modules he wants direct access to during the session. The selected list will then appear in a special "Module" menu in the GUI menu bar (see Figure 7.2).

7.4 A use case : "Testing Assisté par Ordinateur" (TAO) and benchmarking

TAO is the French acronym for computer based assessment. TAO is a project performed in collaboration between the Centre de Recherche Public Henri Tudor and the University of Luxembourg. The main objective is to provide teachers, HR managers, pedagogues, etc. with a versatile and distributed platform to manage e-testing resources (tests creation, subjects management, large scale test delivery, results management) and, also, to deliver on line test to the subjects. For this purpose Generis was specialized into six kinds of different modules. Different plug-ins were implemented for e-testing specific needs (Test and Item authoring tool, results presentation). Due to the large scale test delivery requirements, several benchmarks and Generis optimizations were performed on the Generis kernel. (for instance, ests had to be delivered to more than 300 subjects in one second).

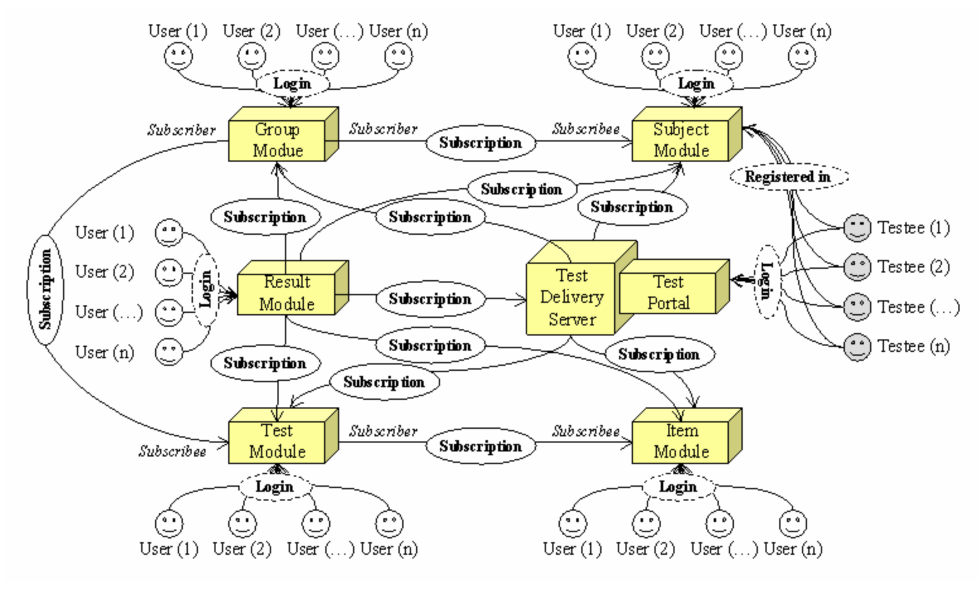


Figure 7.3: Computer based assessment

Generis was also used in different projects of the CRP Henri Tudor for many purposes like knowledge management, e-learning resource management (see Figure 7.3), leading to the development of new plug-ins or Generis improvements.

7.5 Generis Deployment

The installation procedure is very simple and non intrusive. Indeed, all the required services and third party applications such as the web server (Apache), the database management system and the PHP engine are installed in a unique directory without any modification to the host platform configuration. Uninstalling the kernel is as straightforward as deleting a directory and its content. However it is still possible to install Generis on its own web server configuration.

7.6 License model and documentation

Generis is an open source product however it is not yet released under the GPL[7] license, expected to be releases it soon. But at this time, we prefer to finish the documentation completely before releasing it to an open source community. Nevertheless, it's still possible to simply agree on a non disclosure document.

Main components of the source code are documented (API), a user guide in the state of draft is available and a programmer's guide should be written soon.

[7] GNU Public License: `http://www.gnu.org/licenses/licenses.html`

7.7 Generis related works in progress

Currently, search in a resource is only possible in the local repository (local module). The possibility to make more extended search in all the external modules that have granted access to the local module through valid subscriptions is under study. In the same fashion, there are currently no possibilities to discover automatically Generis modules that are installed on the web. This makes the localization of existing helpful modules rather cumbersome without maintaining a central repository of installed modules together with their descriptions. The possibility to create such directories or to exploit fully the P2P network formed by the subscription scheme must be studied. The later implies that an administrator should be able to enable or disable the public accessibility of the URL of its own subscriptions (or as subset of) to a third party module. The anonymous resource querying by another module or simply by requesting the RDF-like URL of the module (in this case the system returns an RDF file with all public resources) without passing through the identification protocol must be added.

Furthermore, new modeling facilities will be added to Generis which will enable the user to define constraints on its model or to define inference rules. Such rules will allow implicit knowledge to be created in the knowledge base.

Chapter 8

Corese

8.1 Introduction

Corese [Corby et al., 2004, Corby et al., 2006] is an ontology-based semantic search engine for the Semantic Web that implements such a matching function using the projection operator defined in the Conceptual Graphs (CG) formalism [Sowa, 1984]. Its general principle is presented in Figure 8.1.

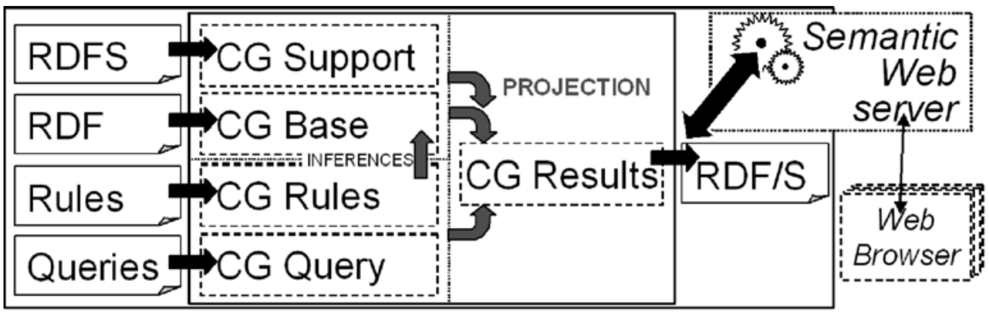


Figure 8.1: Corese general principle

8.1.1 Theoretical Foundations of Corese

The Corese engine internally works on conceptual graphs. When matching a query with an annotation, according to a shared ontology, these are translated in the conceptual graph model [Sowa, 1984, Chein et al., 1998]. Through this translation, Corese takes advantage of the existing work of this knowledge representation community, in particular the results on operators and reasoning capabilities of the Conceptual Graphs formalism.

Conceptual Graph (CG) and RDF(S) models share many common features and a mapping can easily be established between RDF(S) and a large subset of the CG model. An in-depth comparison of both models has been the starting point of the development of Corese [Corby et al., 2000, Delteil et al., 2001].

Both models distinguish between ontological knowledge and assertional knowledge. In both models, the assertional knowledge is positive, conjunctive and existential and it is represented by directed labeled bipartite graphs. In Corese, an RDF graph G representing an annotation or a query is thus translated into a conceptual graph CG.

Regarding the ontological knowledge, the class (resp. property) hierarchy in a RDF Schema corresponds to the concept (resp. relation) type hierarchy in a CG support. RDF properties are declared as first class entities like RDFS classes, in just the same way that relation types are declared independently of concept types in a CG support. This is this common handling of properties that makes relevant the mapping of RDFS and CG models. In particular, it can be opposed to object-oriented language, where properties are defined inside classes.

There are some differences between the RDF(S) and CG models in their handling of classes and properties. However they can be quite easily handled when mapping both models. Mainly, the RDF data model supports multi-instantiation whereas the CG model does not and a RDF property declaration may specify several constraints for the domain (resp. range) whereas in the CG model, a relation type declaration specifies a single constraint for the domain (resp. range). However, the declaration of a resource as an instance of several classes in RDF can be translated in the CG model by generating the concept type corresponding to the most general specialization of the concept types translating these classes. Similarly, the multiple domain (resp. range) constraints of an RDF property can be translated into a single domain (resp. range) constraint of a CG relation type by generating the concept type corresponding to the most general specialization of the concept types constraining the domain (resp. range) of the property.

As a result, the management of RDF(S) through conceptual graphs consists in compiling the type hierarchies of the CG support, the association of a compiled type to each resource, and, finally, the use of the projection operation of the CG model as the keystone of an optimized query processing based on compiled type hierarchies.

This projection operation is the basis of reasoning in the conceptual graph model. A conceptual graph G_1 logically implies a conceptual graph G_2 iff it is a specialization of G_2 (noted $G_1 \leq G_2$). A graph G_1 is a specialization of G_2 iff there exists a projection G_2 of into G_1 such that each concept or relation node of G_2 is projected on a node of G_1 whose type is the same as the type of the corresponding node of G_2 or a specialization of it, according to the concept type hierarchy and the relation type hierarchy.

Formally, let us define a CG as a labeled bipartite graph $G=(C,R,E,l)$ where C and R are the sets of its concept nodes and of its relation nodes, E is the set of its edges and l is a mapping which labels each relation node r of R by a relation type *type(r)* of the relation type hierarchy $\top_r$ and each concept node c of C by a couple *(type(c), ref(c))* where *type(c)* is a concept type of the concept type hierarchy $\top_c$ and *ref(c)* is an individual marker or the generic referent $*$. The projection operation is then defined as follows

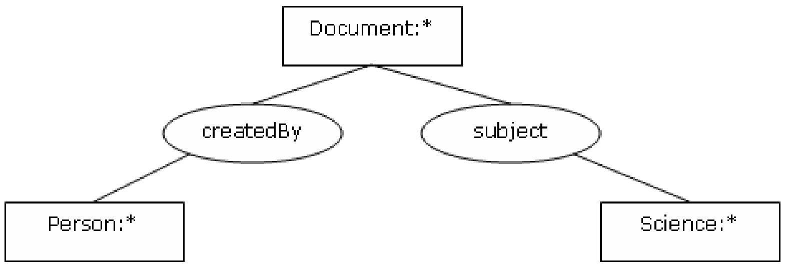
[Chein et al., 1998]: A projection from a CG $G=(C_G,R_G,E_G,l_G)$ to a CG $H=(C_H,R_H,E_H,l_H)$ is a mapping Π from to and from to which:

- preserves adjacency and order on edges: $\forall rc \in E_G$, $\Pi(r)\,\Pi(c) \in E_H$, and if c is the i^{th} neighbor of r in G then $\Pi(c)$ is the i^{th} neighbor of $\Pi(r)$ in H;

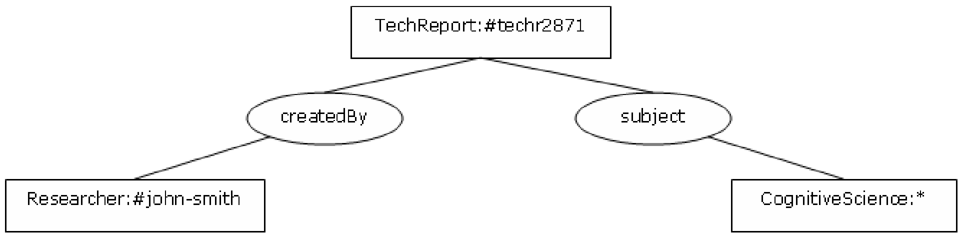
- may decrease labels: $\forall x \in C_G \cup R_G$, $l_H(\Pi(x)) \leq l_G(x)$.

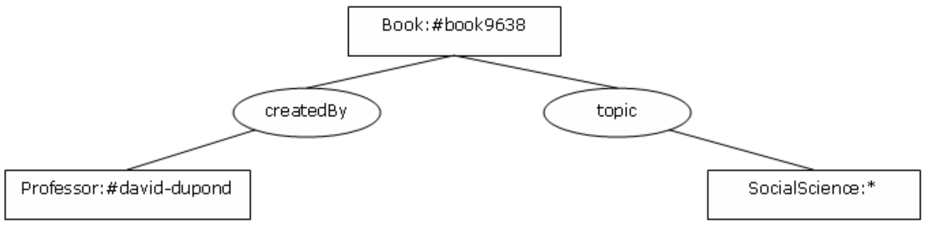
A query is thus processed in the Corese engine by projecting the corresponding conceptual graph into the conceptual graphs translated from RDF(S). The retrieved web resources are those for which there exists a projection of the query graph into their annotation graphs.

For example the following query graph enables us to search for documents about science and their authors.



When processing this query, Corese retrieves a technical report of a researcher about cognitive science and a book of a professor about social science: these documents are annotated with the following graphs upon which there exists a projection of the query graph.





The node `[Document:*]` of the query graph is projected upon `[TechReport:techr2871]` in the first graph and upon `[Book:book9638]` in the second, the types `TechReport` and `Book` being subclasses of `Document` in the ontology shared by these annotation graphs and the query graph, and the uri doc1 and doc2 specializing the generic referent *; the node [Person:*] is projected upon [Researcher:john-smith] and [Professor:david-dupond], their types being subclasses of Person and the uri john-smith and david-dupond specializing the generic referent *; the node [Science:*] is projected upon [CognitiveScience:*] and [SocialScience:*], their types being subclasses of Science; the node (createdBy) is projected upon the node of the same type in both graphs; and the node (subject) is projected upon the node of the same type in the first graph and upon the node (topic) in the second, topic being a subtype of subject in the ontology.

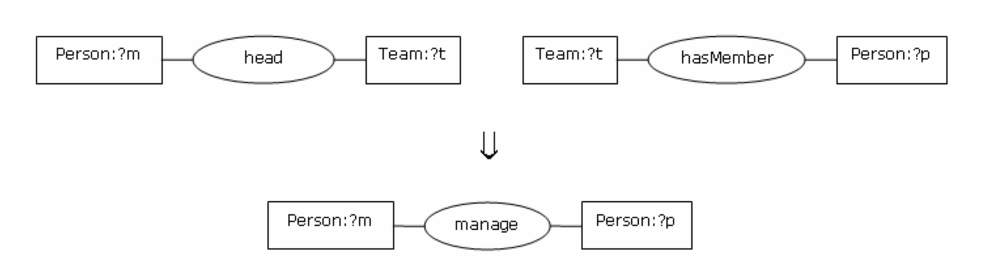
8.1.2 Corese Ontology Representation Language

The first ontology representation language of Corese was RDFS. It has progressively been extended to handle some major features of OWL Lite. Our choice of RDFS is mainly historical: the first implementations of Corese with RDF(S) preceded the emergence of OWL. However the different projects in which Corese has been experimented have shown us that the expressivity of RDF(S) is sufficient in many applications - if extended with inference rules and approximation in the query language. We think that OWL Lite features are quite sufficient to handle most knowledge representation problems encountered in Semantic Web applications. Corese provides OWL value restrictions, intersection, subclass and algebraic properties such as transitivity, symmetry and inverse. It also provides the annotation, versioning and ontology OWL statements. Corese does not yet provide cardinality restrictions, property and class equivalences, owl:sameAs and loops in subsumption hierarchy.

These extensions to OWL features are based on domain axioms which are taken into account when matching a query with an annotation [Corby and Faron-Zucker, 2002]. We have proposed an RDF Rule extension to RDF and Corese integrates an inference engine based on forward chaining production rules. The rules are applied once the annotations are loaded and before the query processing occurs: the annotation graphs are enriched before the query graph is projected. This is the key to the scalability of Corese to the web application in which we have used it.

The production rules of Corese implement conceptual graph rules [Salvat, 1998]: a rule $G_1 \Rightarrow G_2$ is a pair of lambda abstractions (λx_1, … λx_n G_1, λx_1, …, λx_n G_2) where the are co-reference links between generic concepts of and corresponding generic concepts of that play the role of rule variables.

For instance, the following CG rule states that if a person ? m is head of a team ? t which has a person ? p as a member, then ? m manages ? p :



A rule $G_1 \Rightarrow G_2$ applies to a graph G if there exists a projection π from to G, i.e. G contains a specialization of G_1. The resulting graph is built by joining G and G_2 while merging each $\pi(x_i)$ in G with the corresponding x_i in G_2. Joining the graphs may lead to specialize the types of some concepts, to create relations between concepts and to create new individual concepts (i.e. concepts without variable).

The Corese rule language is based on the triple model of RDF. The syntax of a rule is the following:

```
<cos:rule>
 <cos:if>
   a triple pattern
 </cos:if>
 <cos:then>
    a triple pattern
 </cos:then>
 </cos:rule>
```

where `cos` is the prefix for the Corese namespace and where the triples correspond to RDF statements whose conjunction is translated into a conceptual graph.

For instance, the CG rule above is the translation of the following Corese rule:

```
<cos:rule>
 <cos:if>
    ?m rdf:type s:Person
    ?m s:head ?t
    ?t rdf:type s:Team
    ?t s:hasMember ?p
    ?p rdf:type s:Person
 </cos:if>
<cos:then>
    ?m s:manage ?p
</cos:then>
</cos:rule>
```

This triple syntax is shared with the Corese query language, which is further described in the next section.

8.1.3 Corese Query Language

The Corese query language is built upon the SPARQL query language : a query is either a triple or a boolean combination of triples. For instance the following query retrieves all the persons (line 1) with their names (line 2) who are authors (line 3) of a thesis (line 4), and it returns their thesis title (line 5):

```
(1) ?p rdf:type kmp:Person
(2) ?p kmp:name ?n
(3) ?p kmp:author ?doc
(4) ?doc rdf:type kmp:Thesis
(5) ?doc kmp:Title ?t
```

The first element of a Corese triple is either a variable or a resource qualified name (an XML qname); the third element is either a variable, a value or a resource qname; the second element is either a property qname, a variable or a comparison operator. Class and property names are thus qnames whose namespaces are either standard and denoted by predefined prefixes (rdf, rdfs, xsd, owl and cos for the Corese namespace) or user-defined prefixes denoting namespaces, as shown in the following example.

```
dc as http://purl.org/dc/elements/1.1/
```

Variable names begin with a question mark. Values are typed with the XSD datatypes: numerical values, `xsd:string`, `xsd:boolean` and `xsd:date`. The language of the value of a literal can be specified by using the @ operator and based on the specification of `xml:lang`. For instance, in the following example, we constrain the title to be in English.

```
?doc kmp:Title ?t@en
```

The comparison operators for equality and difference (`=, ! =`), ordering (`<, <=, >, >=`) and string inclusion and exclusion (~, ! ~) enable us to compare a variable with a value or with another variable. For instance in the following example, we constrain the title so that it must include the word ’web’.

```
filter ( ?t ~ "web" )
```

Type comparators enable us to specify constraints on some types in a query: strict specialization (`<:`), specialization or same type (`<=:`), same type (`=:`), generalization or same type (`>=:`), strict generalization (`>:`).

For instance, by using the `<:` operator in the following example, we constrain the document to be a strict specialization of a thesis (e.g. a PhD thesis, a MSc thesis, etc.).

```
filter ( ?doc <: kmp:Thesis )
```

By default, a list of triples is a conjunction. The `union` operator is also available and brackets enable us to combine conjunctions and disjunctions in a query. Corese handles such queries by putting them in disjunctive normal form, processing each conjunctive sub-query and juxtaposing all the results.

Let us note that the Corese query language supports ontological reasoning by querying ontologies just like annotations, since RDF Schemas are RDF data. For instance, the following query retrieves all the properties whose `domain` is a subclass of the `kmp:Document` concept.

```
?p rdf:type rdf:Property
?p rdfs:domain ?c
?c rdfs:subClassOf kmp:Document
```

Some SQL-like operators extend the core Corese query language to improve the presentation of the retrieved answers:

- By default, the matching of all the variables occurring in a query are returned from the retrieved annotations. A `select` operator allows to select the only variables whose matching are desired in the answers.
  For instance, in the following example, we select only the title of the document and the name of its author.

```
select ?t ?n where
```

- A `group` operator corresponding to the SQL `group-by` allows to group the retrieved answers according to one or more concepts instead of listing separately answers about the same concept(s) (in case an annotation is answering a query several times).
  For instance, when querying for documents on a specific subject and written by an author, a `group` on the *document* variable will avoid that a document written by several authors appears several times, once for each of its authors. By default, a `group` is applied to the first variable of a query.

- A `count` operator, combined with a `group` allows the counting of the (different) documents retrieved. For instance, to mention the number of documents written on each subject, `count` is applied to the *document* variable and `group` to the *subject* one.

8.2 Approximate Semantic Web search

We have extended the core query language of Corese to address the problem of possible mismatch between end-user and ontological concepts. Corese is able to cope with queries for which there is no exact answer by approximating the semantics of the query, its structure, or both.

8.2.1 Ontological Approximation

The first principle of the Corese semantic approximation is to evaluate semantic distances between classes in the ontology. Based on this ontological distance, Corese not only retrieves web resources whose annotations are *specializations* of the query, it also retrieves those whose annotations are *semantically close*.

8.3 Corese Software

8.3.1 Architecture

Corese is a software tool developed in Java. A stand-alone version is publicly available under the INRIA license at `http://www.inria.fr/acacia/corese` including Java packages, a documented API and a Swing query GUI. A Corese semantic web server has also been developed. It is designed according to a 3-tier architecture (Figure 8.2): the presentation layer responsible for the presentation of data, the application layer implementing the application business logic and the persistent layer managing the persistence of the application data.

Presentation Layer

This layer, also called the Corese web server, is responsible for generating the content to be presented in the users' browser (ontology views and browsing controls, query edition interfaces, annotation forms, answer presentation, etc.). This part relies on a model-view-controller architecture to handle HTTP requests from the client (users' Web browser) and generate HTTP responses fed by the Corese services as appropriate (i.e. upper modules of the Business Logic Layer: Query Parser, CG-to-RDF

Pretty Printer) and formatted using XSLT or JSP templates. This layer is implemented by a set of servlets and provides the front-end of what we call a Semantic Web Server: an HTTP server able to solve semantic web queries submitted through HTTP requests; able to provide JSP tags to include semantic web processing and rendered results in web pages; able to provide XSLT extensions to include semantic web functions in XPath expressions, thus improving RDF/XML transformation capabilities; able to provide a form description language to dynamically build forms using queries for instance to populate the different choices of a drop-down box.

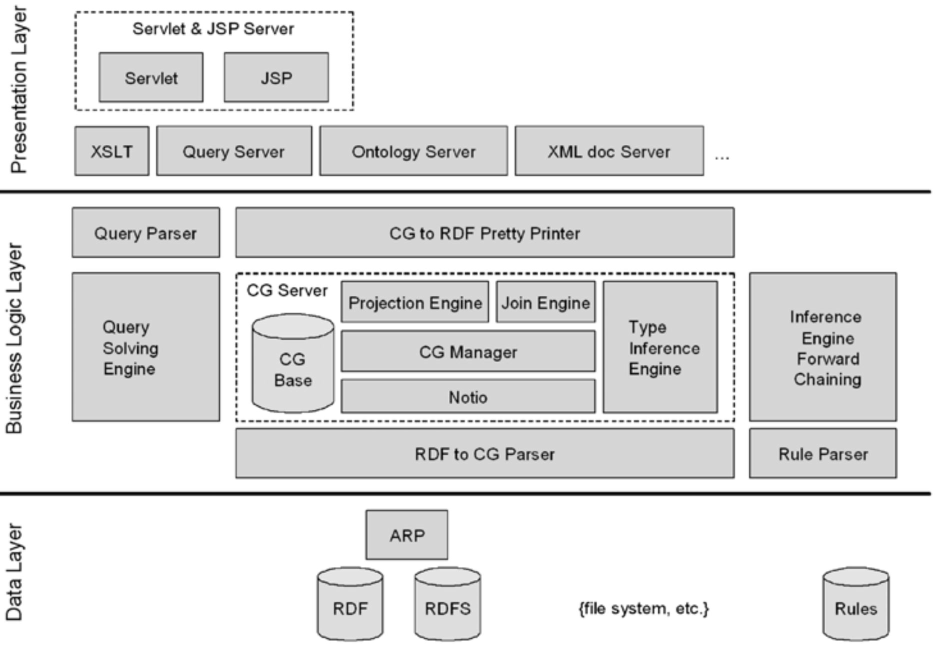


Figure 8.2: Corese 3-tier architecture

Application Layer

The Corese application layer (or Business Logic Layer) is a platform that implements three main services accessible through a well defined API: a Conceptual Graph server (whose CG manager is based on the Notio API [Southey and Linders, 1999]), a Query engine and an Inference Engine handling forward chaining rules. A set of parsers transforms RDF to CG, Rules to CG Rules and Queries to CG graphs to be projected on the base. The core CG server implements the management of the CG base, the projection and join operators and type inference on the type lattices. A CG-to-RDF pretty-printer allows us to produce any result in RDF/XML syntax. This layer is also an independent package and API that can be downloaded and used by developers to add semantic web capabilities to their applications.

Data Layer

It comprises the RDF(S) data (ontology and annotations) accessed by means of the ARP [HP,] parser which produces triple events interpreted and translated by the RDF-to-CG Parser. In addition rules are saved in separate files and parsed by the Rule Parser of the Business Logic Layer.

Chapter 9

SeWeSe

9.1 Introduction

All semantic web applications using a semantic engine (like Corese) provide common functionalities that can be factorized into a semantic web application development platform. This is what the SeWeSe platform does. The goal of such a platform is to provide reusable, configurable and extensible components in order to reduce the amount of time spent to develop new semantic web applications and to allow these applications to focus on their domain specificities.

SeWeSe is built upon Corese engine and provides a set of functionalities like generation of interfaces for queries, edition and navigation, and for the management of the transverse functions of a portal (presentation, internationalization, security, etc.). An ontology editor, a generic annotation editor and a basic rule editor are parts of the SeWeSe platform.

9.2 Architecture

Concerning its architecture, SeWeSe relies on Tomcat: and provides a set of filters, servlets, JSP tags and libraries as well as some templates to build new applications.

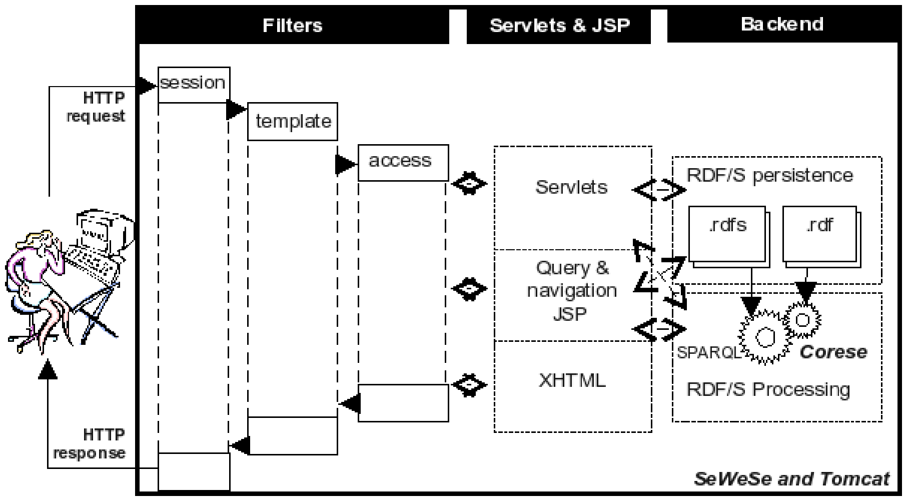


Figure 9.1: SeWeSe architecture

SeWeSe is a Java 1.5 application, using the XSLT engine Xalan 2.7.0, the logging library Log4J 1.2.12. It can be deployed under Tomcat 5.5.x and uses JSP 2.0 and the standard JSP Tag library 1.1.2. Relying on the tomcat architecture, SeWeSe provides several mechanisms to customize and extend its functionalities. For instance filters can be added to implement pre and post processing and the existing filter handling sessions can be parameterized to extract and memorize characteristics of the user for the whole session (e.g. her interests).

9.3 Available components

The architecture of SeWeSe is organized around four main types of components:

Toolkits that group back-end interfaces (e.g. accesses to Corese) and transversal functionalities (e.g. applying an XSLT style-sheet) . These toolkits have documented API and their instance is placed in the context of the web application so that developers of applications based on SeWeSe can reuse them directly.

JSP tags are extensions of the standard JSP library to provide tags handling recurrent task in developing a semantic web application (e.g. submit a query) . These tags can be used by developers of applications based on SeWeSe anywhere in their JSP.

Filters provide mechanisms to implement transversal processing on requests and response without duplicating code or multiplying references (e.g. template filter adding header and trailer to every page served) . Filters provided by SeWeSe are customizable and any new filter can be added.

Servlets implement responses requiring important processing (e.g. modifying an existing ontology) . Existing servlets can be called by developers of applications based on SeWeSe and new one can be added.

A number of toolkits are available to develop new applications, among which:

- concurrentToolkit: provides functionalities to handle concurrent accesses to the resources of the application
- xsltToolkit: to apply efficiently XSLT style-sheets (caching mechanisms)
- domToolkit : to manipulate elements of an XML document
- I18nToolkit: to handle internationalization
- engineToolkit: to submit queries to the semantic search engine (e.g. SPARQL queries to Corese)

- ontologyToolkit: to manage the ontologies
- notionToolkit: to handle objects representing RDF resources
- annotationToolkit: to manage RDF annotations

A number of JSP tags are predefined:

- tags to lock resources to manage concurrency
- tags to provide graphic widgets (e.g. calendar to select a date, dialog pop-ups, tool-tips) ;
- tags to locally (i.e. in a page) or transversally (i.e. in external files) handle internationalization;
- tags to display and navigate in all or sub-parts of RDFS schemas;
- tags to create new annotations in the knowledge base;
- tags to modify existing annotations;
- tags to include the result of a query and specify the style-sheet to format it;
- tags to handle security and access control;

SeWeSe also includes a number of administration tags (e.g. to reset the server).

Using these tags and an additional complete set of tools and models, SeWeSe also support the generation of ontology-based forms. These forms can be used to create, modify or query the annotations.

A number of browser-independent javascript libraries are available, among which:

- A library of usual functions for HTML interfaces (e.g. to validate forms, to hide or show an HTML DIV, to modify a form as it is filled, etc.);
- A library to use AJAX (e.g. for auto-completion) ;
- A library to support graphical components;
- A WYSIWYG HTML editor.

Several filters are shipped with SeWeSe:

- The session filter provides standard login and session management functionalities plus a declarative way to handle the data about a user that are persistent through the session (e.g. name, access rights, department, language, etc.);
- The template filter provides a simple way to apply a template to a set of pages (e.g. include headers and trailers) ;

- The Access filters allow to set restrictions on access rights for targeted resources using a set of extensible profiles.

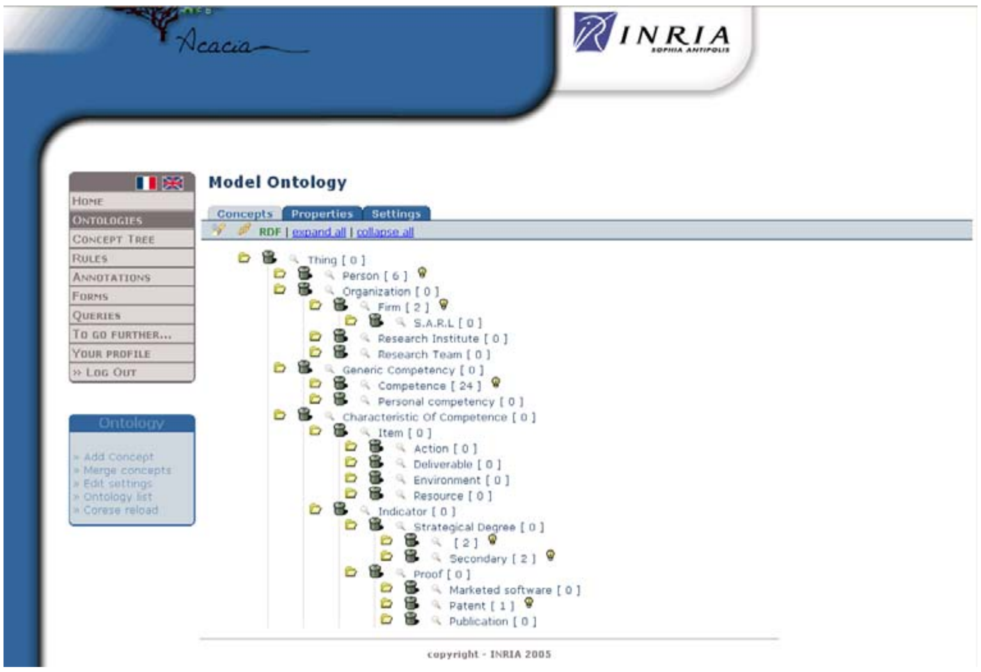
9.4 Links with elementary KM services

In addition to the development bricks stated above, SeWeSe comes with a customizable web-based ontology editor, a simple rule base editor and a generic annotation editor that can be used for development or administration purposes and that can be reused in dedicated editors.

9.4.1 Ontology editor (Ontology creation)

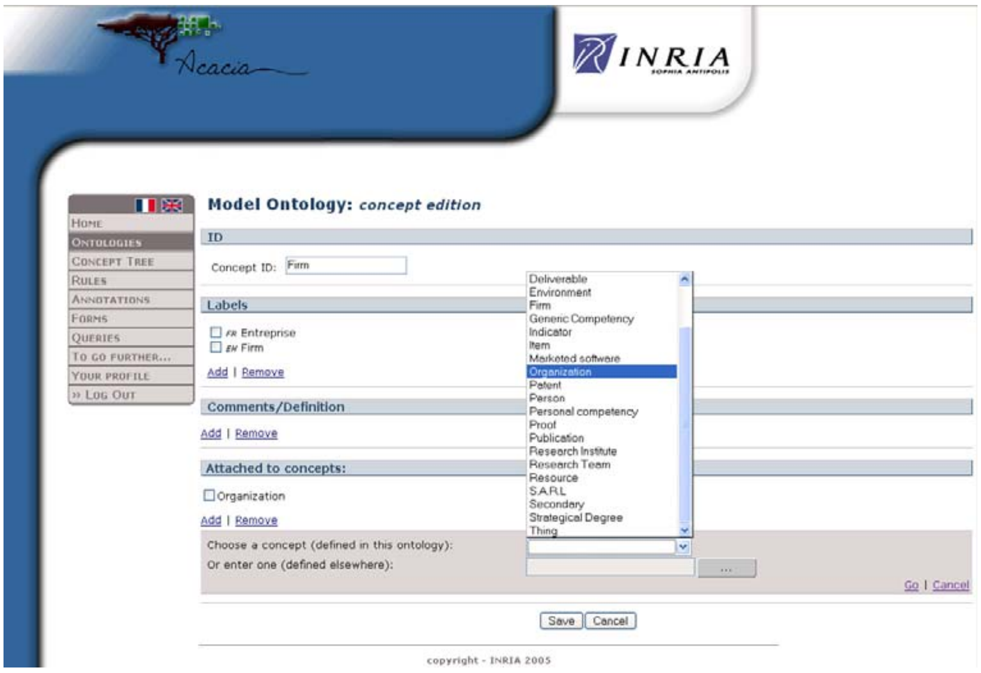
The ontology editor (see Figure 9.2) allows users to navigate into the application ontology by visualizing existing concepts and properties (and how often they are used in the application). Different views can be displayed: a hierarchical one, a flat one and a filtered one.

Each concept or property (see Figure 9.3) can be modified. And it is possible to add new concepts and new properties. In addition, the editor allows us to merge several concepts into a new one.



```
Thing [ 0 ]
    Person [ 6 ]
    Organization [ 0 ]
        Firm [ 2 ]
            S.A.R.L [ 0 ]
        Research Institute [ 0 ]
        Research Team [ 0 ]
    Generic Competency [ 0 ]
        Competence [ 24 ]
        Personal competency [ 0 ]
    Characteristic Of Competence [ 0 ]
        Item [ 0 ]
            Action [ 0 ]
            Deliverable [ 0 ]
            Environment [ 0 ]
            Resource [ 0 ]
        Indicator [ 0 ]
            Strategical Degree [ 0 ]
                [ 2 ]
                Secondary [ 2 ]
            Proof [ 0 ]
                Marketed software [ 0 ]
                Patent [ 1 ]
                Publication [ 0 ]
```

Figure 9.2: Ontology editor: hierarchical view of ontology



Model Ontology: *concept edition*

ID

Concept ID: Firm

Labels

FR Entreprise
EN Firm

Add | Remove

Comments/Definition

Add | Remove

Attached to concepts:

Organization

Add | Remove

Choose a concept (defined in this ontology):
Or enter one (defined elsewhere):

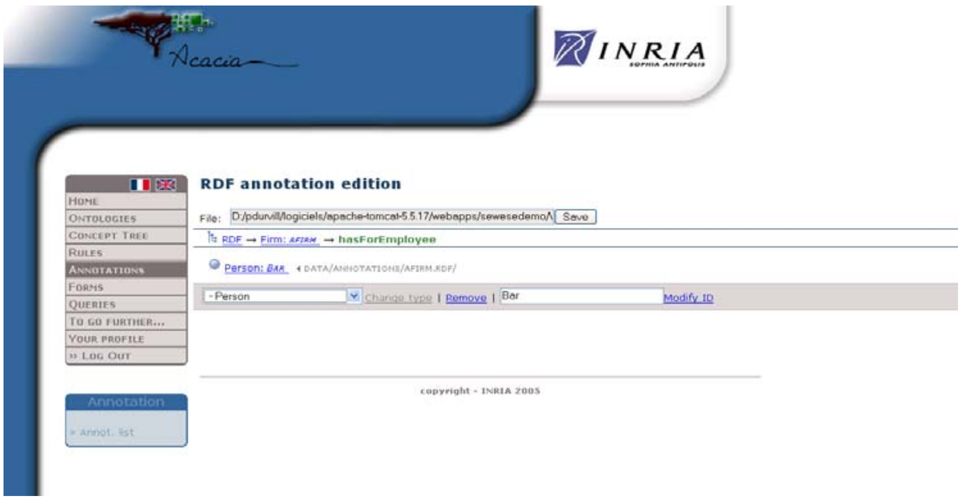
copyright - INRIA 2005

Figure 9.3: Ontology editor: concept edition

It is possible to customize the ontology editor in order to get a graphical interface dedicated to an administrator that will manage the application ontology (or a part of the application ontology) evolution. This management tasks will be, for example, to add new concepts that have been suggested by the application users, to merge concepts, to get the most popular concepts (the concept that are frequently used) and so on.

9.4.2 Annotation Editor (Annotation)

The annotation editor (see Figure 9.4) allows users to edit annotations in a generic way. It is possible to create new instances and add relations between existing instances, to modify a relation between two instances, to modify an instance ID and update this ID elsewhere, to remove relations and instances. Because the editor relies on the RDF/RDFS structure and not on the interpretation of the application annotations and ontologies, the editor user never minds the domain of working. He will work with instances and relations between them directly.



RDF annotation edition

File: D:/pdurvill/logiciels/apache-tomcat-5.5.17/webapps/sewesedemo/\

RDF → Firm: AFIRM → hasForEmployee

Person: BAR ◂ DATA/ANNOTATIONS/AFIRM.RDF/

- Person | Bar

copyright - INRIA 2005

Figure 9.4: Annotation editor: view of an annotation

9.4.3 Clustering (Presentation and Visualization)

SeWeSe allows us to display global views of the used concepts and their repartitions i.e. if the added value of a set of answers to a query is no in an individual answer but in the repartition of the answers among different classes, then one can use the subsumption tree as a dendrogram to cluster answers at a chosen level of details. The result is the ability to control the precision/specialisation of the vocabulary used to answer your query. The Figure 9.5 display the repartition of instances retrieved by a query over a more or less specific set of classes.

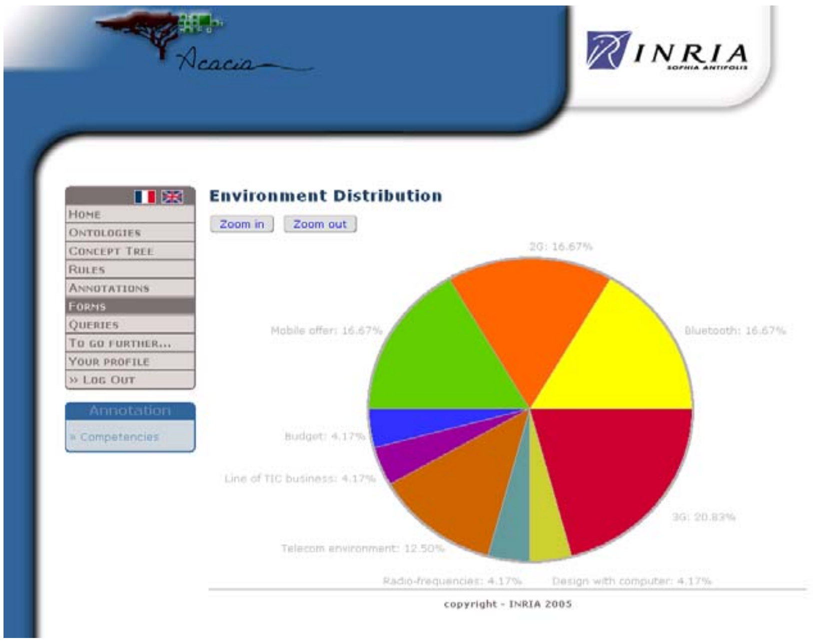


Figure 9.5: Clustering view

Chapter 10

Meat

10.1 Introduction

The MEAT project [Khelif et al., 2006, Khelif et al., 2005b, Khelif et al., 2005a] aims at building a memory of experiments in the DNA micro-array domain, and at supporting biologists in their interpretation and validation of the results of their experiments, through analysis of semantically annotated Medline scientific articles (see Table 10.1).

System	MEAT, a memory of experiments of biologists on DNA micro-arrays
Context	Memory of DNA-micro-array experiments
Domain	Bio-medical
Company	IPMC
Semantic Web Approach	External, open web
Resources	Scientific articles useful for interpretation or validation of results of DNA micro-array experiments
Information sources	Human experts
Ontology	UMLS ontology (i.e. the semantic network of UMLS) that contains 134 concept types and 54 relations, and is linked to millions of terms via UMLS meta-thesaurus.
Expert validation	Validation of extracted relations and of generated annotations, by biologists of IPMC
Typical user query Used reasoning	- “ Find all the articles asserting a given (resp. any) relation between a given biological entity (gene, protein, ...) and another biological entity ” - “ Find all the articles asserting a given (resp. any) relation between a given gene and a given (resp. any) disease ” Classic projection
Corese functions used End-user evaluation	Corese new query language - Use of rules - Use of approximate reasoning Evaluation by biologists of IPMC
Services offered	- Automatic extraction of relations and term from texts - Automatic generation of RDF annotations

Table 10.1: Summary of MEAT project

10.2 MEAT Components

The MEAT system (see Figure 10.1) comprises:

- the **MeatOnto** ontology composed of:
  - UMLS semantic network considered as a general ontology for the bio-medical domain: the UMLS hierarchy of semantic types can be regarded as a hierarchy of concept types and the terms of the meta-thesaurus can be considered as instances of these concept types.
  - The MGED ontology proposed by Microarray Gene Expression Data Group to describe DNA experiments,
  - DocOnto, an ontology for describing metadata on documents.
- **MeatAnnot** that offers a service of annotation of a text with respect to an existing ontology (here the UMLS ontology) .
- **MeatSearch** that offers a search service based on the Corese semantic search engine, with dedicated interfaces for visualizing graphically the annotations satisfying the user's query.

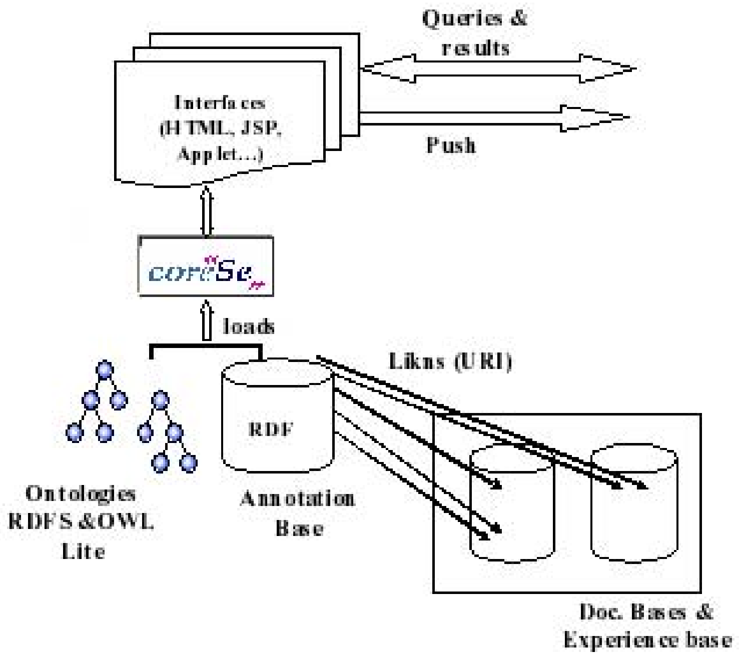

Queries & results
Interfaces (HTML, JSP, Applet...)
Push
coreSe
loads
Likns (URI)
RDF
Ontologies RDFS &OWL Lite
Annotation Base
Doc. Bases & Experience base

Figure 10.1: Architecture of MEAT system

More precisely, the MeatAnnot system relies on analysis of scientific articles through NLP tools (GATE modules, TreeTagger, RASP) in order to generate automatically RDF annotations (not only concepts but also relations among concepts). See Figure 10.2 for a general view on MeatAnnot methodology.

Based on an analysis of occurrences of relations in a corpus of biological texts, a relation detection grammar is offered for detecting UMLS relations such as (`interacts_with, expressed_in, has_role` ...) in the texts. Then, in the sentences where relations were detected, MeatAnnot relies on UMLS Knowledge Server in order to recognize terms corresponding to UMLS concepts and constituting the arguments linked by the relation detected. Then MeatAnnot generates an RDF annotation that is validated by the biologist and then stored. The annotations base is then used by Corese semantic search engine for retrieving the articles possibly relevant for answering the biologist's query and supporting him/her in the interpretation of a DNA micro-array experiment.

The MEAT project illustrates the reuse of an existing ontology and the use of linguistic tools for generating RDF annotations. Provided that some adaptations are performed, MeatAnnot could be used with a CoP-dependent ontology and could thus offer to CoPs such a service of semi-automatic annotation from texts.

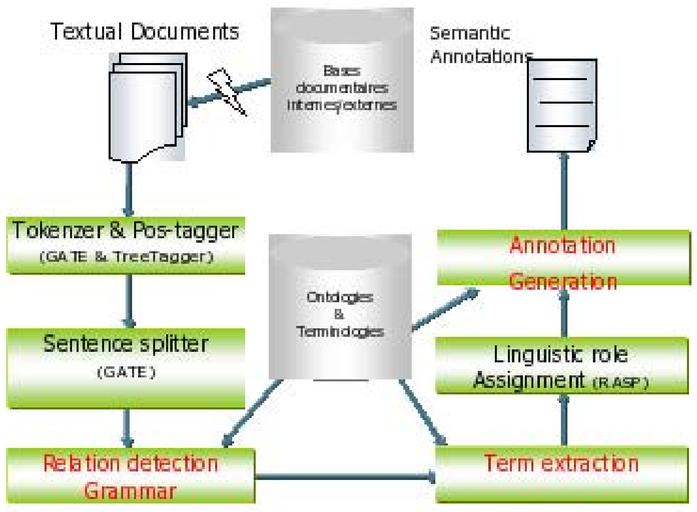

Textual Documents
Bases documentaires internes/externes
Semantic Annotations
Tokenzer & Pos-tagger (GATE & TreeTagger)
Sentence splitter (GATE)
Relation detection Grammar
Ontologies & Terminologies
Annotation Generation
Linguistic role Assignment (RASP)
Term extraction

Figure 10.2: MeatAnnot Methodology for semi-automatic annotation of texts

Chapter 11

Virtual Staff

11.1 Introduction

The Life Line project [Dieng-Kuntz et al., 2006, Dieng-Kuntz et al., 2004] aims at developing a knowledge management tool for a health care network (see Table 11.1).

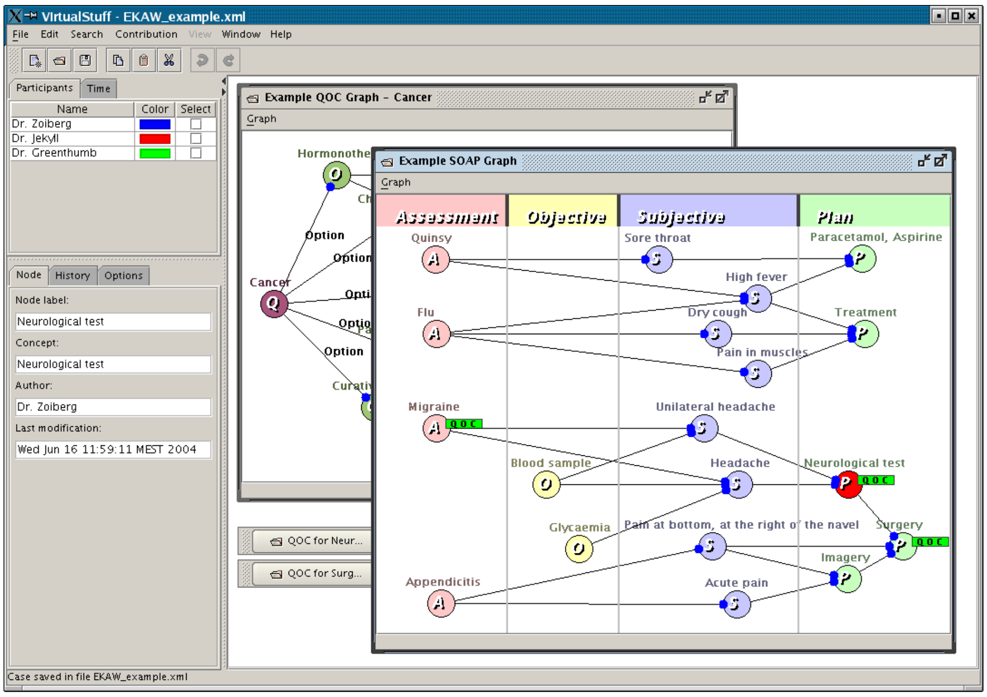
System	A Virtual staff in the framework of the "Ligne de Vie" (Life Line) project
Context or scenario	Support to cooperative reasoning of members of a health care network
Domain	Medicine
Company	Nautilus, a society specialised in marketing medical software
Scope of Semantic Web Approach	Medical semantic Web among several distributed health partners
Resources	Medical documents such as patient records, guide of best practices
Information sources	A medical database that we translated automatically into RDF(S)
Ontology	Nautilus ontology comprising 26432 concepts and 13 relations
Expert validation	Validation by our industrial partner Nautilus
Typical User query Used reasoning	"Find the past sessions of virtual staff where a given therapy was chosen for the patient and what were the arguments in favour of his solution" "Find a past session of virtual staff where the patient suffered from a given symptom and what was the disease diagnosed and the therapy protocol decided" Classic projection
CORESE functions used	CORESE past query language
End-user evaluation	Evaluation by our industrial partner
Services	Support to cooperative problem solving + Integration of an ontology with SOAP and QOC graphs

Table 11.1: Summary of Life Line Project

The "Virtual Staff" [Dieng-Kuntz et al., 2006, Ruzicka et al., 2004] allows the members of a healthcare network to visualize their collective reasoning when formulating diagnosis assumptions or when making decisions of therapeutic procedures. This application corresponds to an organisational semantic Web dedicated to a medical community cooperating in the context of a health care network.

In the Virtual Staff, the dependencies between the various diagnostic and therapeutic hypotheses are represented through a graph using the concepts defined in the Nautilus ontology. The doctor reasons by linking the health problems to the symptoms, the clinical signs and the observations in order to propose care procedures. The Virtual Staff thus relies on the SOAP model (Subjective, Objective, Assessment, Plan) [Weed, 1971], used by the medical community. In this model:

- the S nodes describe current symptoms and clinical signs of the patient,
- the O nodes describe analyses or observations of the physician,
- the A nodes correspond to the diseases or health problems of the patient,
- and the P nodes correspond to the procedures or action plans set up in order to solve the health problems.



Name	Color	Select
Dr. Zoiberg	🟦	☐
Dr. Jekyll	🟥	☐
Dr. Greenthumb	🟩	☐

Node label: Neurological test

Concept: Neurological test

Author: Dr. Zoiberg

Last modification: Wed Jun 16 11:59:11 MEST 2004

Case saved in file EKAW_example.xml

Figure 11.1: Interface of the Virtual Staff

Sometimes, the doctor may need to visualize (see Figure 11.1) all the possible solutions and the arguments in their favour or against them. The QOC model (Question Options Criteria) [Moussavi, 1999], used by CSCW community for support to decision-making, can then be useful. In this model, a question Q corresponds to a problem to solve. To solve the question Q, several Options are possible, with, for each option, the criteria in its favour and the criteria against it: each option is thus connected positively or negatively to criteria. Two types of questions are possible for the Virtual Staff:

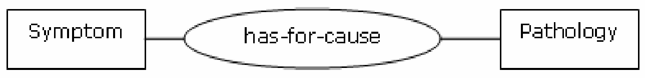
- *Diagnosis of a pathology:* Which pathology explains the clinical signs of the patient?
- *Search of a prescription:* Which action plan will enable to treat the diagnosed pathology?

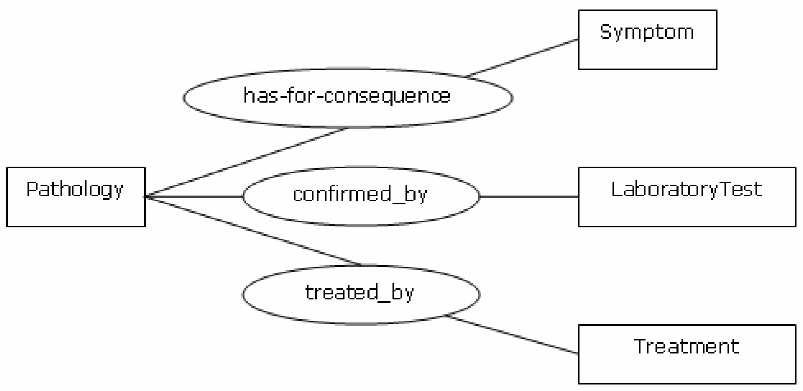
In the Virtual Staff, SOAP graphs enable to visualize the medical record and in phase of decision, QOC graphs enable to choose between pathologies or between action plans. Using the Nautilus ontology, the system can propose a list of possible concept types to help the users to build SOAP and QOC graphs. Table 11.2 indicates the concept types among the subtypes of which each category of node must be chosen.

Node Category	Possible concept types
S node in a SOAP graph	Symptom PathologicalAgent ForeignBody
O node in a SOAP graph	LaboratoryTest
A node in a SOAP graph	Malformation Pathology PsychoProblem
P node in a SOAP graph	Treatment DiagnosticGesture.
Option in a QOC graph	Pathology Treatment
Criterion in a QOC graph	Symptom LaboratoryTest Pathology Treatment

Table 11.2: Nodes of Virtual Staff graphs and ontology concept types

The arcs between the nodes correspond to relations among concepts:

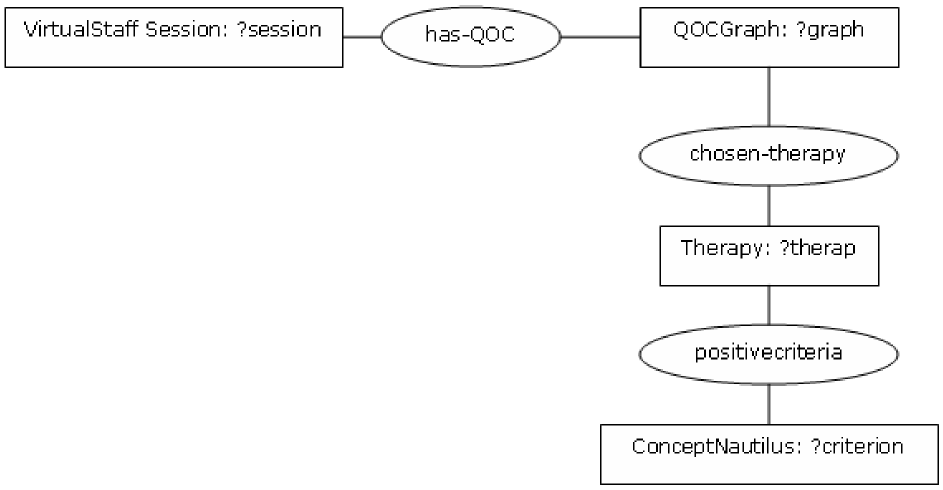




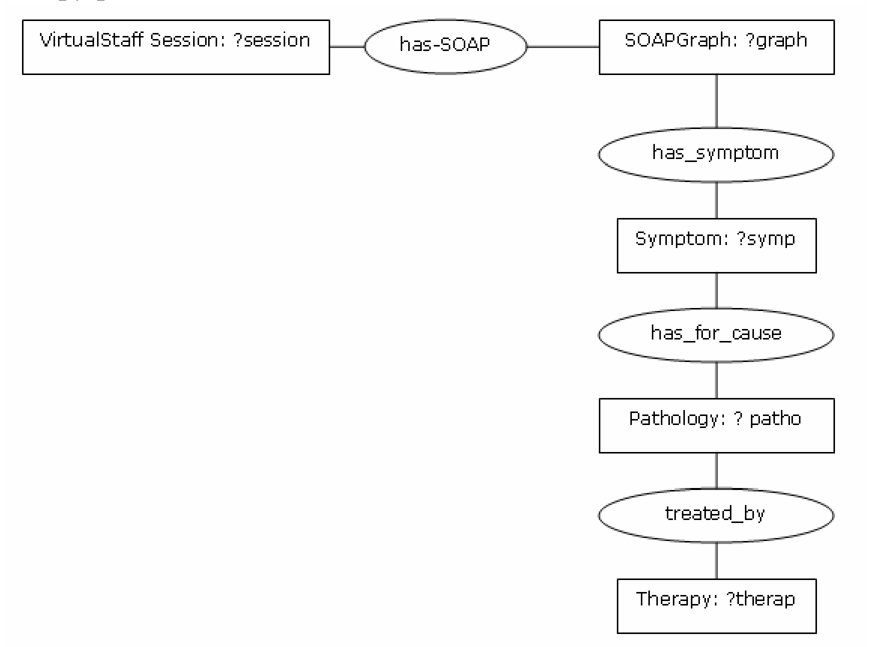
The arcs between the nodes of a QOC tree can be interpreted by «Question has-solution Option» or by «Option has-positive-criteria Criterion» or by «Option has-negative-criteria Criterion».

11.2 Examples of Queries

- *"Find the past sessions of virtual staff where a given therapy was chosen for the patient and what were the arguments in favour of this solution "*



- *"Find a past session of virtual staff where the patient suffered from a given symptom and what was the disease diagnosed and the therapy protocol decided"*



The Virtual Staff could be extended to cooperative problem solving for CoPs, not especially in medical domain: a correspondence could be made between the SOAP model (resp. the QOC models) and the CoP domain concepts.

For example, if the CoP members perform cooperatively a diagnosis on an artefact:

- the S nodes would correspond to the possible symptoms of faulty system,
- the O nodes to the possible observations on the system,
- the A nodes to the possible diagnoses,
- the P nodes to the possible plans or procedures for repairing the problem diagnosed.

By the same way, if the CoP members must decide between several possible diagnoses or between several repair procedures, they can build a QOC graph where:

- The Q nodes correspond to the issue to be solved
- The O nodes correspond to the possible options (i.e. the possible diagnoses) ,
- The C nodes associated to a given option correspond to the possible positive criteria in favor of this option and the possible negative criteria against this option.

A session of the Virtual Staff can be seen as a specific case of problem solving (see Figure 11.2). Through Corese, a kind of case-based reasoning allows to retrieve past cases satisfying some constraints;

- Find a past case where a given option had been chosen and give the positive criteria in favor of this option,
- Find a past case where the choice was between a given option and another one,
- For a given kind of issue, find all the options proposed by a member of s given class of users.
- Find the participant whose proposed options were the most often finally chosen
- Etc

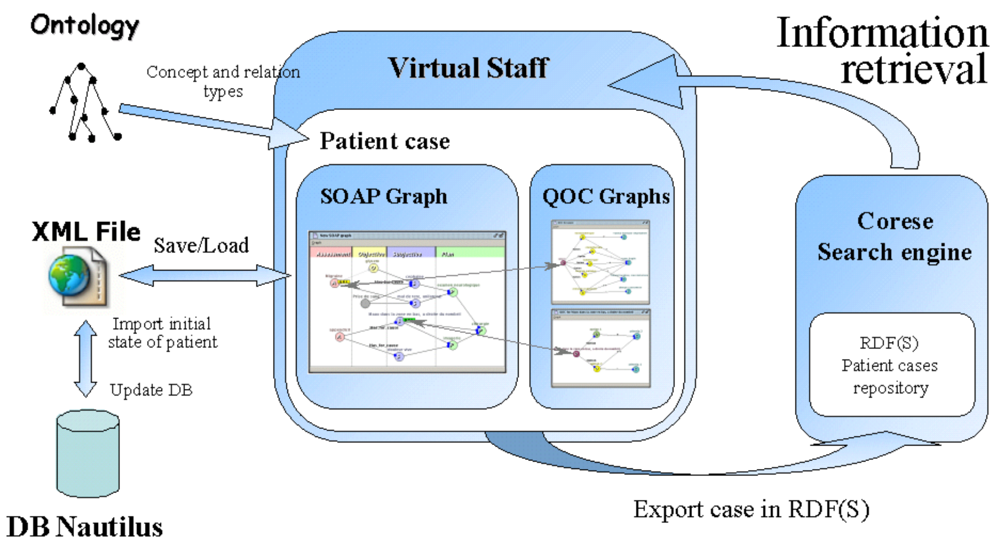


Figure 11.2: Architecture of Virtual Staff

As a conclusion, the Virtual Staff could be useful for cooperative problem solving in a CoP needing to visualize such a collective resolution and to access to past cases.

Part III

Architecture

This third part presents the KM tool modular architecture. Since the KM services must be offered not only to human end-users but also to other services, we will rely on a web service-oriented architecture. In chapter 12, after the description of the principles of a service-oriented architecture and some W3C standards dedicated to web services, the components of the KM tool architecture are detailed.

Chapter 12

Architecture of KM services

12.1 Introduction

The Knowledge management services in Palette must be offered to CoPs members to help them in their activities inside the CoPs. To achieve this objective, KM services can be provided as tools that provide a number of functionalities, but they should also be accessible by other services or tools. This observation was the starting point of the conception of the architecture of KM system and of the definition of elementary KM services (Part I) that we designed to be the building blocks of the KM system. Since the needs of CoPs cover a large spectrum. We try to identify and specify the elementary KM services in order to be able to use them for developing more specific and CoPs oriented tools.

Web services[8] offer the possibility to implement such architecture. A generic architecture of the KM system has been already proposed in Palette's DoW (see Figure 12.1)

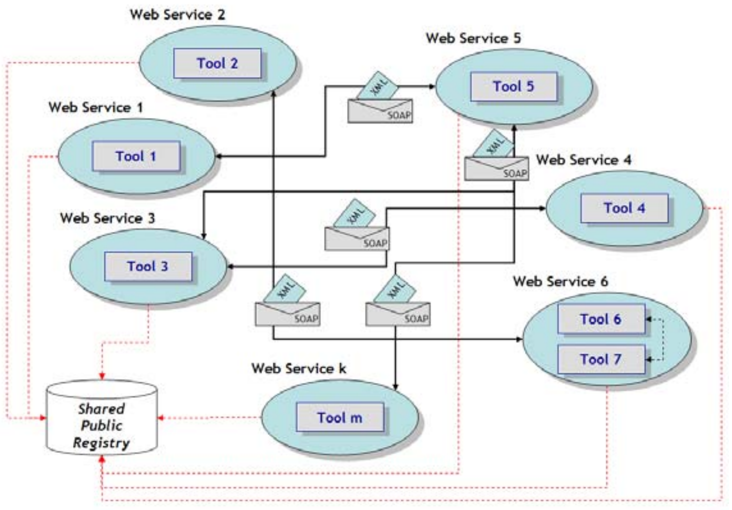


Figure 12.1: Web services architecture

[8] According to the W3C a Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface that is described in a machine-processable format. Other systems interact with the Web service in a manner prescribed by its *interface* using messages.

The architecture of the KM services we present here is a realization of such a generic architecture. It comprises a kernel of services (the KM elementary services described in Part I) and define a new type of services, namely a **complex web services** that will provide more complex functionalities. These complex services can be used by CoPs members *directly* through a human-computer interface or *indirectly* when they are used by other services or tools. In order to meet these requirements we need to use standards that help us:

- to improve integration;
- to facilitate the reuse of developed tools and services;
- to establish standardized data representation;
- to allow organizational agility.

We relay on a the SOA (Service Oriented-Architecture) and on the SOAP (Simple Object Access Protocol) standard for communication between services.

This chapter starts by presenting SOA [Krafzig et al., 2005] and SOAP [Snell and Tidwell, 2001]. Then we give an overview of the architecture. Finally we present some details of the different components that it comprises.

12.2 SOA & SOAP

Service Oriented Architecture

Service Oriented Architecture was first proposed by [Schulte and Natis, 2003]. They specified SOA as "a style of multi-tier computing that helps organizations share logic and data among multiple applications and usage modes."

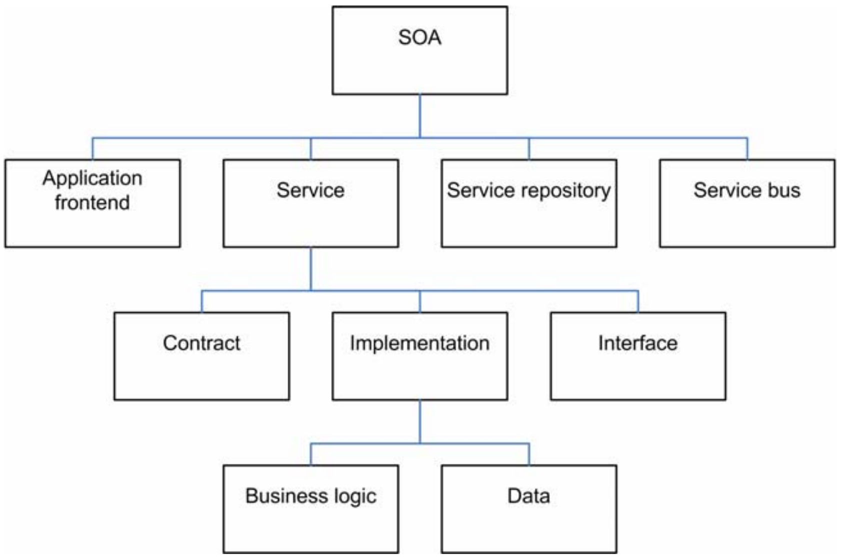


Figure 12.2: Elements of SOA [Krafzig et al., 2005]

SOA is usually based on Web services standards (e.g., using SOAP or REST) that have gained broad industry acceptance. These standards (also referred to as Web service specifications) also provide greater interoperability and some protection from lock-in to proprietary vendor software. This characteristic answers an important requirement of the Palette project which is to provide open-source tools for CoPs.

SOA can also be regarded as a style of Information Systems architecture that enables the creation of applications that are built by combining loosely coupled and interoperable services.

The elements of a SOA are presented in Figure 12.2. We can see that a service in SOA is defined by a contract, an interface and an implementation. In Part I of this deliverable, we defined the *contract* (functionalities) and the *interface* of the elementary services of Palette. We also gave indications about the *data* used for their *implementations*.

Simple Object Access Protocol

SOAP is a web service W3C standard which provides a means of communication between applications running on different operating systems, with different technologies and programming languages. It is a protocol for exchanging XML-based messages over a computer network, normally using HTTP.

In the example below[9], a `GetStockPrice` request is sent to a server. The request has a `StockName` parameter, and a Price parameter will be returned in response. The namespace for the function is defined in http://www.example.org/stock address.

The SOAP request:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>

</soap:Envelope>
```

[9] From W3 Schools SOAP Tutorial.

A SOAP response:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>

</soap:Envelope>
```

12.3 Architecture

The Knowledge Management system of Palette will comprise:

A **kernel** which is a set of (i) *elementary services*, providing the basic operations of knowledge management, for instance: ontology creation, annotation, cooperative knowledge creation, retrieval, dissemination, visualization and presentation, evaluation, and evolution. And a (ii) *services registries* provide the ability to register, discover, and manage Web services in kernel and the complex services component (see section 12.4.1).

Complex services which are a composition of elementary services (see section 12.4.2) implemented to achieve some CoPs tasks. The set of available complex services can be viewed as a virtual KM platform that can be instantiated for each CoP or for each CoP user.

A **front-end** that describes the way the user access the KM services and can be of two types:

- A **stand-alone front-end:** When a service can be used directly by CoPs, we need to develop a front-end to access (see section 12.4.3);
- **Widgets:** in case that KM services that will be integrated with existing tools (see section 12.4.4).

Resources including ontologies, annotations, user profiles, etc.

The figure 12.4 presents a general view of this architecture.

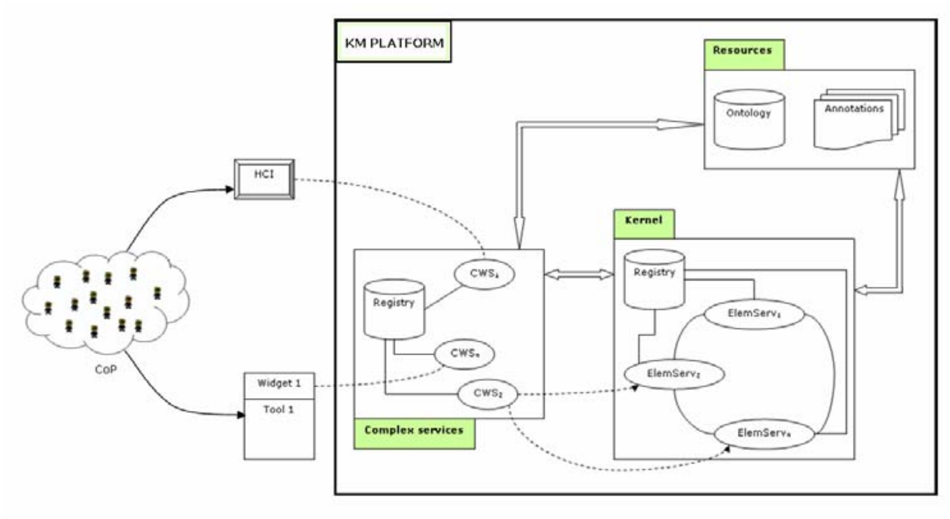


Figure 12.4: Palette KM services architecture

12.4 KM system components

12.4.1 Services registries

The web services offered to the CoPs may cover a large scope of functionalities. The number of web services that may be developed over the elementary services offered by Palette may grow, if the KM system is used by more CoPs. Centralized facilities for access and control of service metadata and artifacts is critical. A service registry provides these capabilities and is a key infrastructural component and cornerstone for SOA deployments.

12.4.2 Complex services

Defining the abovementioned complex services will allow us to adapt quite easily the services provided to a CoP, according to its needs and function of the tasks to be accomplished by its members. This solution is flexible enough to allow several manners of providing services to the users. We may easily integrate these services to existing tools via a SOAP interface, we may group them in a platform which meets certain needs of the CoPs, and which may be customized according to the user's needs and constraints. An example of the integration of complex services in a platform can be seen in SweetWiki[10] [Buffa, 2006]

These complex services will resort to elementary services. For instance, the complex service of running a query in the KB corresponds to

[10] http://www-sop.inria.fr/acacia/soft/sweetwiki.html

a number of quite simple tasks in a CoP: finding the resources that match the user's request and presenting the results in the most adequate manner for the recipient. This service will use elementary search and visualization services. The front-end of this service will be made up of a component corresponding to the request formulation and one corresponding to results visualization; both of them need to be completely transparent for the user. For these two components and associated tasks, the service resorts, respectively, to the retrieval service and the visualization/presentation service.

Complex services which correspond to a single elementary service constitute a particular case of complex services. In this case, the service the user has access to is a mere instantiation of the corresponding elementary service.

These complex services will be either:

- integrated directly, as front-ends (HCI in Figure 12.4) with which the user can communicate directly
- or as widgets in the existing tools.

12.4.3 Complex services front-end

Independent front-ends are user (web) interfaces for complex services. They must be defined according toof the CoP members' needs and they have to use the ontology that is dependent on the CoP.

12.4.4 Widgets

Widgets are quite elementary tools which meet particular user needs, when using a tool. They bring a series of improvements to the tool without changing their fundamental architectures. The services associated to widgets must be conceived in such a manner that satisfies this constraint. Their interfaces differ from independent front-ends in that they must integrate an existing tool.

An example of widget for which CoPs have shown a lot of interest is a complex service of navigation/modification/creation of annotations, on resources mentioned in a discussion (a forum or a chat). This widget must be integrated to the chat or web forum tool. It will identify the resources which annotations are present in the system and allow the users to visualize these annotations, create annotations on the resources they introduce or modify resource annotations.

Remark

The development of these interfaces (in most cases) is in our point of view outside the scope of WP3 and should be discussed with people from WP5.

Part IV

Use cases and Integration

Part IV presents some scenarios of use (chapter 13) and studies how the KM services thus specified in WP3 will be interoperable with the mediation services developed in WP4 (chapter 14). We thus give some use cases of invocation of KM services from the mediation tool developed in WP4. The conclusion (chapter 15) recapitulates the possible solutions of integration of the proposed KM services with the tools presently used by the CoPs considered in Palette.

Chapter 13

Scenarios of use

This chapter presents a set of potential use cases in order to illustrate the various probable ways of using the abovementioned KM services. More specifically, the following paragraphs present a set of user-oriented scenarios for using the “Knowledge Creation and Annotation”, “Knowledge Retrieval and Dissemination”, “Knowledge Presentation and Visualization”, “Knowledge Evaluation”, and “Knowledge Evolution and Maintenance” services. The proposed KM services are to be properly designed so as to support the various needs of various CoPs in different contexts. In the following, we provide a comprehensive description of a potential scenario of use, which comprises all the abovementioned services. In other words, the proposed scenario consists of multiple use cases, each one corresponding to a particular KM service.

Environment/situation	Selection of name and packaging for a new product
Practice/activity	Group thinking and decision making
Actors	CoP members, i.e. marketing managers, executives, advertising and marketing experts, packaging designers, etc
Resources	A list of names and packaging descriptions proposals and related marketing aims A list of existing products’ names and packaging descriptions (of the same category) Product description Product target group(s) profile(s) (e.g. age, country) Related market statistical reports of preferences Collaboration tools (e.g. brainstorming tools, decision making tools, whiteboards, e-mail, forum, etc)

Table 13.1: The investigated Scenario

13.1 The Scenario

A potential Scenario to be addressed assumes that a CoP of managers has been established to help decision making about the name and packaging of a new product to be released in the market. Table 13.1 presents the environment/situation, practice/activity, actors and resources related to this Scenario.

13.2 Use Cases

CoP's members may utilize the provided Knowledge Management services as described in the following use cases. It is clearly noted that the following use cases are potential, indicative, and by no means exhaustive.

As far as the "Knowledge Creation and Annotation" service is concerned, a set of potential use cases for the CoP members (in the context of the above scenario) is described in Tables 13.2 and 13.3. These knowledge creation and annotation use cases also apply to the development of the CoP related ontologies.

Knowledge creation from :	Description
Environment	Creation of environment descriptions repository
Activities	Creation of good and bad practices repository (this service is closely related to the knowledge evaluation service). Creation of related activities repository (e.g. e-mails, discussion boards)
Actors	Creation of user profiles (e.g. static information, domain of expertise, preferences, activity records, etc)
Resources	Creation of resources repository (e.g. reports, white papers, statistical reports, market analysis, etc)

Table 13.2: Use cases of knowledge creation service

Annotations on :	Description
List of names proposals	Annotations regarding name/Packaging description, proposal creator, aims
List of existing names	Annotations regarding name/Packaging description, owner company
Product description	Annotations regarding the shape, colour, weight, volume, fragile/non fragile, conservation conditions, recycling options
Product target group(s) profile	Annotations regarding static information about target group(s) (e.g. average age, income, places of purchase, country, etc.)
Decision making activities	Annotations regarding the activity's minutes, such as actor, impact, reactions, etc Annotations regarding other decision making related activities, carried out through e-mails, brainstorming sessions, whiteboards or forums

Table 13.3: Use cases of annotation service

As far as the “Knowledge Retrieval and Dissemination” services is concerned, potential use cases for the CoP members are described in Tables 13.4 and 13.5, respectively.

Retrieval of :	Description
Practice/activity	Retrieve related past cases so as to avoid bad practices Retrieve past decisions so as to evaluate past choices (this service is related to the knowledge evaluation service)
Actors	Retrieve knowledge regarding the users, such as their domain of expertise
Resources	Retrieve knowledge resources such as reports, white papers, statistical reports, market analysis, etc

Table 13.4: Use cases of knowledge retrieval service

Dissemination of :	Description
Resources	Dissemination of documents such as minutes, white papers, reports, supporting material, etc
Actor profiles	Dissemination of actor profiles for expertise management Dissemination of actor profiles to be considered for the development of adaptive and/or awareness services
Activity records	Dissemination of the activity records towards the shaping of user profiles (this is related to the “Knowledge Creation and Annotation” service) and the evolution of the decision making activity

Table 13.5: Use cases of knowledge dissemination service

As far as the “Knowledge Presentation and Visualization” service is concerned, potential use cases are described in Table 13.6.

Presentation and Visualization of :	Description
Activity	Presentation and visualization of the users activities (e.g. name/packaging proposal, argument supporting a proposal, etc) Presentation and visualization of the users interactions Presentation and visualization of the activity’s outcomes (e.g. decision or inability to reach a decision, consensus, veto, etc)
Resources	Presentation and visualization of resources (see Table 1) according to their significance, validity, etc (this is related to the “Knowledge Evaluation” service)
User profiles	Presentation and visualization of user profiles

Table 13.6: Use cases of presentation and visualization service

As far as the “Knowledge Evaluation” service is concerned, potential use cases are described in Table 13.7:

Evaluation of :	Description
Name/packaging proposals and related arguments	Evaluation of the knowledge expressed in the proposals and arguments supporting (or speaking against) them
Product target group profiles	Evaluation of product target group profiles validity
User profiles	Evaluation of user profiles validity
Decision making activity	Evaluation of the decision making activity towards the shaping of good or bad practices Evaluation of the decision making activity towards the shaping of good or bad decisions

Table 13.7: Use cases of the evaluation service

As far as the “Knowledge Evolution and Maintenance” services are concerned, potential use cases are described in Table 13.8:

Evolution and maintenance of :	Description
User profiles	Evolution and maintenance of user profiles given the fact that these are dynamic. This could be also related to the user activities and use of Palette services
Resources	Evolution and maintenance of resources (see Table 13.1), in cooperation with the “Knowledge Creation and Annotation” service

Table 13.8: Use cases of the evolution and maintenance service

Chapter 14

Integration with Mediation services

According to the Palette's DoW (see pp. 30-31), one of the core aims of our work is to achieve interoperability among the three different sets of Palette services, i.e. information services, knowledge management services, and mediation services. Acknowledging that one of the most dominant practices of CoPs requiring mediation support is decision making, the integration of KM and Mediation services in particular is considered as a very important issue towards the satisfaction of CoP collaboration and communication needs. In this chapter, we justify the above argument and we present a scenario of KM and Mediation services integration.

14.1 KM and Mediation interrelation

Empirical evidence shows that collaborative decision making is an interplay between social and knowledge processes [Schwarz, 2003]. The social perspective concerns the mechanisms of coordination in the expression and discussion of views, on which the effects of power structures and group-thinking play a significant role [Ackermann et al., 2004]. These mechanisms are in turn responsible for the dynamics of convergence or divergence of opinions as a CoP deals with an issue. On the other hand, the conduct of argumentative discourses between decision makers results in the pooling of group members' individual knowledge and expertise [Karan et al., 1996]. As argumentative discourses evolve, the stakeholders' knowledge is usually clustered around specific ideas, solutions and views, while the whole collaboration process can result in knowledge exchange and reconstruction [Evangelou and Karacapilidis, 2005]. At the same time, such discourses, when appropriately structured and maintained, may stimulate active participation and encourage knowledge sharing. Their final outcome is usually a decision, considered as "piece of knowledge indicating the nature of an action commitment" [Holsapple and Whinston, 1996]. When decisions are the result of appropriate (e.g. argumentation-based) reasoning and evaluation mechanisms, the decision making process also constitutes new knowledge.

Taking the above remarks into account, it becomes clear that the collection and processing of knowledge is ubiquitous in a decision making environment. As characteristically stated in [Li and Lai, 2005], the efficiency and effectiveness of decision making is strongly related to the appropriate exploitation of all possible organizational knowledge resources. But, in common practice, issues such as information/knowledge loss or distortion, as well as suboptimal decision making (i.e. not enough issues and alternatives are explored) during argumentative discourses, are primary problems of productivity loss. In addition to that, we argue that the existing organizational knowledge can be made explicit through the decision making process *per se*. Similarly, new state-of-the-art knowledge can be also created and formally represented. As derives, the proper integration of Knowledge Management and Mediation services can be of great value for CoPs in such contexts.

14.2 KM and Mediation services integration scenario

A potential scenario to be addressed in the context of Palette assumes that a CoP has been established to help managers enhance their organizational competitive position in the market. Activities such as the above, definitely advance learning in a CoP. CoP' members may exploit Mediation services and register themselves as participants in the associated collaboration to be conducted. Table 14.1 presents the environment/situation, practice/activity, actors and resources related to this scenario.

Environment/situation	CoP members collaboration towards learning
Practice/activity	Mediation, communication, brainstorming, collaboration, argumentation and decision making
Actors	CoP members External actors, moderators (optional)
Resources	CoP documents (local or remote) External sources (e.g. internet) CoPe_it !

Table 14.1: The investigated scenario (CoP mediation activity)

CoPe_it! (see http://copeit.cti.gr) is a web-based prototype that supports argumentative collaboration. It has been developed within the Palette context to provide CoPs with the desired Mediation Services (the final version of CoPe_it! is expected to be ready in M18). Discourses being held in CoPs can be considered as social processes, and as such, they often result in the formation of groups whose knowledge is clustered around specific views of the problem. Following an integrated approach, CoPe_it! provides CoPs members engaged in such discourses with the appropriate means to collaborate towards the solution of diverse issues. In addition to providing a platform for group reflection and capturing of organizational memory, CoPe_it! augments teamwork in terms of knowledge elicitation, sharing and construction, thus enhancing the quality of the overall process and building a collective memory of a CoP. This is due to its structured language for discussion and its mechanism for the evaluation of alternatives. Taking into account the input provided by the individual members of a CoP, CoPe_it! constructs an illustrative discourse-based knowledge graph that is composed of the ideas expressed so far, as well as their supporting documents. Moreover, through the integrated reasoning mechanisms, discussants are continuously informed about the status of each discourse item asserted so far and may reflect further on them according to their beliefs and interests on the outcome of the discussion. In addition, CoPe_it! aids group sense-making and mutual understanding through the collaborative identification and evaluation of diverse opinions. Furthermore, CoPe_it! provides a shared web-based workspace for storing and retrieving the messages and documents of the participants. The knowledge base of CoPe_it! maintains all the above items (messages and documents), which may be considered, appropriately processed and transformed, or even reused in future discussions. Discourse items (e.g. goal, alternatives, and arguments in favor or against) posted by CoP members in CoPe_it! can be considered as knowledge (or knowledge to be retrieved from) items.

As derives from the above, CoPe_it! already provides a set of KM services. Nevertheless, it could be integrated with more specialized KM services in order to further support the CoP members' needs. Figure 14.1 presents a possible Scenario of invoking KM services from CoPe_it!.

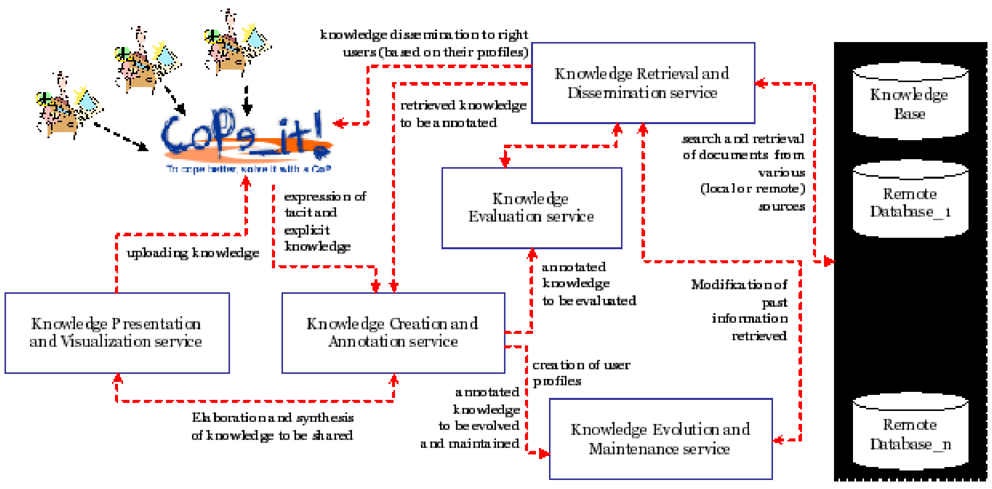

knowledge dissemination to right users (based on their profiles)
Knowledge Retrieval and Dissemination service
retrieved knowledge to be annotated
CoPe_it!
To cope better, solve it with a CoP
search and retrieval of documents from various (local or remote) sources
Knowledge Base
Remote Database_1
Knowledge Evaluation service
expression of tacit and explicit knowledge
uploading knowledge
annotated knowledge to be evaluated
Modification of past information retrieved
Knowledge Presentation and Visualization service
Knowledge Creation and Annotation service
creation of user profiles
annotated knowledge to be evolved and maintained
Elaboration and synthesis of knowledge to be shared
Knowledge Evolution and Maintenance service
Remote Database_n

Figure 14.1: A potential scenario for invoking KM services from CoPe_it!

As shown in figure 14.1, CoPe_it! may communicate with the "Knowledge Creation and Annotation" service that handles among others the creation of members' profiles. Whenever a user wants to contribute to an ongoing collaboration, CoPe_it! invokes the "Knowledge

Representation and Visualization Tool" that handles the expression of tacit and explicit knowledge. This last tool may communicate with the "Knowledge Retrieval and Dissemination" services which in turn are able to carry out activities related to searching and retrieving of related documents from diverse (local or remote) knowledge and data bases. Then, one may exploit the features and functionalities provided by either the "Knowledge Evaluation" or the "Knowledge Evolution and Maintenance" services, which enable the elaboration and synthesis of knowledge to be shared among the members of the CoP. It is through the "Knowledge Retrieval and Dissemination" service that knowledge is uploaded to CoPe_it! , which in turn may handle issues of dissemination of knowledge to the appropriate CoP' members (e.g. based on their rights and responsibilities). Furthermore, CoPe_it! may invoke the "Knowledge Creation and Annotation" service that facilitates the annotation of already expressed knowledge (before storage), according to the CoP-dependent ontology models. Another potential use case concerns the evaluation of the knowledge expressed (e.g. in terms of validity, etc.) from each user through the "Knowledge Evaluation" service.

Table 14.2 summarizes the abovementioned and presents additional use cases of invoking KM services from CoPe_it!. It is noted again here that the use cases presented in this section are potential, indicative, and by no means exhaustive.

KM Service	Description
Knowledge Annotation and Creation	Creation of user profiles Creation of new knowledge (e.g. by combining previous pieces of knowledge, by reconsidering past cases, etc.) Annotation of tacit and explicit knowledge expressed during argumentation (e.g. creator, relative knowledge items, etc)
Knowledge Presentation and Visualization	Presentation and visualization of tacit and explicit knowledge Presentation and visualization of argumentation (includes diverse discourse items) Presentation and visualization of collaboration results Presentation and visualization of decision making Presentation and visualization of conflicts, alliances, etc. Presentation of connections between the items and their creators
Knowledge Retrieval and Dissemination	Search and retrieval of related documents from diverse (local or remote) knowledge and data bases Dissemination of knowledge according to users expertise, needs, interests Knowledge retrieval from the connections between the items and their creators
Knowledge Evaluation	Evaluation of the tacit and explicit knowledge expressed during argumentation Evaluation of retrieved items validity Evaluation of users' credibility (according to his past actions, position, authority etc.) Evaluation of retrieved items relevance to issue under consideration
Knowledge Evolution and Maintenance	Elaboration and synthesis of knowledge to be shared among the members of the CoP (appropriate handling of expressed tacit and explicit knowledge) Evolution and maintenance of user profiles Evolution and maintenance of retrieved and new knowledge Synthesis (and maintenance) of knowledge expressed during argumentative collaboration Synthesis (and maintenance) of retrieved knowledge

Table 14.2: Use cases of invoking KM services from CoPe_it!

Chapter 15

Conclusion

This deliverable proposed a preliminary specification of basic KM services interesting for CoPs:

- Individual or collaborative knowledge creation services with support to ontology creation, and annotation;
- Knowledge retrieval and dissemination;
- Knowledge presentation and visualization;
- Knowledge evaluation;
- Knowledge evolution and maintenance.

We described their functionalities, their possible interfaces with other services and their interfaces for interaction with the human user. Based on the available descriptions of CoPs (for example, the CoP "Telecom-INT - UX11 Module"), we gave examples of possible uses of these basic KM services by CoPs.
We also described various tools available among the partners: Generis, Corese, SeWeSe, MEAT, Virtual Staff. Provided that some extensions and/or adaptations will be developed, in order for them to be usable in the context of CoPs, these tools could offer some of these services.
We also proposed a web service-oriented architecture enabling to offer all such KM services in a modular way, with interfaces both towards end-users but also towards other KM services or even towards other Palette services (such as Mediation services).

The interoperability of this architecture is illustrated by examples of use cases of invocation of KM services from the mediation tool developed in WP4. Another interoperability issue that needs to be investigated is how to integrate the proposed KM services with the tools presently used by the CoPs considered in Palette.

15.1 Integration with CoPs Tools

Several solutions may be envisaged in the interaction between CoPs tools and KM services:

1. We may integrate new services to the existing tools. We will attempt to integrate them as web services by modifying the existing tools (keeping in mind the strong constraints related to legal and technical issues).
2. We may keep the existing tools as such, but render them accessible as web services (a SOAP Layer may have to be developed). This will allow us to use proprietary software provided that they have an available API. In this case, we will develop a portal allowing the integration of several applications (ours and the other existing applications). If a relative unification of approaches is necessary, a service ensuring the connection between the different applications will be vital (probably employing the meta-data repository around which all services gravitate). The SOA architecture permits a user to log on to a portal (possibly installed and configured by a CoP administrator) and to choose the services (existing applications or Palette services) which interest him from a given available set of applications and services. He may therefore recognize tools he is already familiar with as part of a vaster application.
3. We may ignore the tools already used, but try to preserve all functionalities for the CoPs. Then we have the choice between (a) developing the services implementing the functionalities of the eliminated tools and (b) replacing these tools by other Open-Source tools which we may adapt. In the latter case, the implementation effort might obviously be enormous and the risk of and rejection by the CoPs is very strong.
4. Last, we may ignore the tools already used, but simply propose our new tools offering new functions in comparison with their previous tools and let the members of the CoPs use both their previous tools and our new tools according to their needs.

As we need more information both from the Palette's CoPs (mainly through WP1), we cannot take yet a decision on which solution will better match the needs of Palette, since we aim at offering tools both generic enough and usable by real CoPs. The decision will be taken in cooperation with WP5, the Palette's work package dedicated to integration.

15.2 Further Work

As a further work, we will:

- Study thoroughly the deliverables of WP1 and the descriptions of the CoPs,
- Make explicit our methodology for building the CoP-dependent ontologies (Task 3.2);
- Refine the KM services specified in this deliverable (Task 3.3) and make explicit the concepts needed to be included in the CoP-dependent ontologies, on which the foreseen KM services would rely;
- Build the CoP-dependent ontologies according to this methodology, using the CoP-available information, and being guided by the meta-models defined in Task 3.1;
- Develop the KM services (with adaptation of the partners' existing tools when needed).

Table of authors

Introduction	*Rose Dieng-Kuntz & Adil El Ghali*
Knowledge Creation & Annotation	*Amira Tifous, Rose Dieng-Kuntz & Adil El Ghali*
Knowledge Retrieval & Dissemination	*Amira Tifous & Rose Dieng-Kuntz*
Knowledge Presentation & Visualization	*Adil El Ghali & Fabien Gandon*
Knowledge Evaluation	*Sylvain Dehors, Adil El Ghali, Alain Giboin & Amira Tifous*
Knowledge Evolution & Maintenance	*Rose Dieng-Kuntz*
Generis	*Thibault Latour, Patrick Plichart & Geraldine Vidou*
Corese	*Olivier Corby*
SeWeSe	*Fabien Gandon & Priscille Durville*
Meat	*Rose Dieng-Kuntz*
Virtual Staff	*Rose Dieng-Kuntz*
Architecture of KM services	*Adil El Ghali*
Scenarios of usage	*Christina Evangelou*
Integration with Mediation services	*Christina Evangelou*
Conclusion	*Rose Dieng-Kuntz & Adil El Ghali*

Bibliography

[Ackermann et al., 2004] Ackermann, F., Eden, C., and Brown, I. (2004). *The Practice of Making Strategy : A Step-by-Step Guide*. Sage Publications Inc.

[Buffa, 2006] Buffa, M. (2006). Sweetwiki: Semantic web enabled technologies in wiki. In *Proc. of WikiSym 2006*.

[Chein et al., 1998] Chein, M., Mugnier, M., and Simonet, G. (1998). Nested graphs: A graph-based knowledge representation model with FOL semantics. In *Proc. of the 6th International Conference on Principles of Knowledge Representation and Reasoning, KR'98*, pages 524–534, Trento, Italy.

[Corby et al., 2000] Corby, O., Dieng, R., and Hébert, C. (2000). A conceptual graph model for W3C resource description framework. In *Proc. of the 8th International Conference on Conceptual Structures, ICCS'00*, number 1867 in LNCS, pages 468–482, Darmstadt, Germany. Springer-Verlag.

[Corby et al., 2004] Corby, O., Dieng-Kuntz, R., and Faron-Zucker, C. (2004). Querying the semantic web with the CORESE search engine. In de Mantaras, R. L. and Saitta, L., editors, *Proc. of the 16th European Conference on Artificial Intelligence (ECAI'2004)*, pages 705–709, Valencia. IOS Press.

[Corby et al., 2006] Corby, O., Dieng-Kuntz, R., Faron-Zucker, C., and Gandon, F. (2006). Searching the semantic web: Approximate query processing based on ontologies. *IEEE Intelligent Systems Journal*, 21(1).

[Corby and Faron-Zucker, 2002] Corby, O. and Faron-Zucker, C. (2002). Corese: A corporate semantic web engine, in proc. of the workshop on real world rdf and semantic web applications. In *Proc. of 11th International World Wide Web Conference, WWW2002*, Honolulu, Hawai, USA.

[Delteil et al., 2001] Delteil, A., Faron, C., and Dieng, R. (2001). Extensions of RDFS based on the conceptual graph model. In *Proc. of the 9th International Conference on Conceptual Structure, ICCS2001*, number 2120 in LNAI, pages 275–289, Stanford, CA, USA. Springer-Verlag.

[Dieng-Kuntz et al., 2004] Dieng-Kuntz, R., Minier, D., Corby, F., Ruzicka, M., Corby, O., Alamarguy, L., and Luong, P.-H. (2004). Medical ontology and virtual staff for a health network. In *Proc. of EKAW'2004*, pages 187–202, Whittlebury Hall, UK.

[Dieng-Kuntz et al., 2006] Dieng-Kuntz, R., Minier, D., Ruzicka, M., Corby, F., Corby, O., and Alamarguy, L. (2006). Building and using a medical ontology for knowledge management and cooperative work in a health care network. *Computers in Biology and Medicine*, 36(Issue 7-8):871–892.

[Evangelou and Karacapilidis, 2005] Evangelou, C. and Karacapilidis, N. (2005). Knowledge-based strategy development: An integrated approach. In *Proceedings of I-KNOW'05*, pages 4–11, Graz, Austria.

[Holsapple and Whinston, 1996] Holsapple, C. and Whinston, A. (1996). *Decision Support Systems: A Knowledge Based Approach*. West Publishing Company.

[HP,] HP. Arp from hp. http://www.hpl.hp.com/semweb/arp.htm.

[Karan et al., 1996] Karan, V., Kerr, D., Murthy, U., and Vinze, A. (1996). Information technology support for collaborative decision making in auditing: An experimental investigation. *Decision Support Systems*, 16:181–194.

[Khelif et al., 2005a] Khelif, K., Dieng-Kuntz, R., and Barbry, P. (2005a). MEAT: An experiment memory for the biochip domain. In *Proc. of the Third International Conference on Knowledge Capture KCAP'05 (Poster)*, Banff, Canada.

[Khelif et al., 2005b] Khelif, K., Dieng-kuntz, R., and Barbry, P. (2005b). Semantic web technologies for interpreting DNA microarray analyses: the MEAT system. In *Proc. of the 6th International Conference on Web Information Systems Engineering WISE'05*, New York.

[Khelif et al., 2006] Khelif, K., Dieng-kuntz, R., and Barbry, P. (2006). Web sémantique pour la mémoire d'expériences d'une communauté scientifique : le projet MEAT. In *Actes des 6èmes journées d'Extraction et de Gestion de Connaissances, EGC'06*, pages 175–186, Lille, France.

[Krafzig et al., 2005] Krafzig, D., Banke, K., and Slama, D. (2005). *Enterprise SOA: Service Oriented Architecture Best Practices*. Prentice Hall.

[Li and Lai, 2005] Li, E. and Lai, H. (2005). Collaborative work and knowledge management in electronic business. *Decision Support Systems*, 39(4):545–547.

[Moussavi, 1999] Moussavi, M. (1999). A case-based approach to knowledge management. In Aha, D., editor, *Proc. of the AAAI'99 Workshop on Exploring Synergies of Knowledge Management and Case-Based Reasoning*, volume WS-99-10, Orlando, Florida. AAAI Press.

[Ruzicka et al., 2004] Ruzicka, M., Dieng-Kuntz, R., and Minier, D. (2004). Virtual staff - software tool for cooperative work in a health care network. Technical Report RR-5621, INRIA.

[Salvat, 1998] Salvat, E. (1998). Theorem proving using graph operations in the conceptual graph formalism. In *Proc. of the 13th European Conference on Artificial Intelligence, ECAI98*, pages 356–360, Brighton, UK.

[Schools, 2006] Schools, W. (2006). SOAP Tutorial. http://www.w3schools.com/soap/default.asp.

[Schwarz, 2003] Schwarz, M. (2003). *Strategy Process: Shaping the Contours of the Field*, chapter A Multilevel Analysis of the Strategic Decision Process and the Evolution of Shared Beliefs, pages 110–136. Blackwell Publishing, Oxford.

[Schulte and Natis, 2003] Roy W. Schulte and Yefim V. Natis, Introduction to Service-Oriented Architecture, Research Note, 14 April 2003, Gartner.

[Snell and Tidwell, 2001] James Snell. Doug Tidwell. Pavel Kulchenko. Programming Web Services with SOAP. 2001. O'Reilly.

[Southey and Linders, 1999] Southey, F. and Linders, J. G. (1999). Notio - a Java API for developing CG tools. In *Proc. of the 7th International Conference on Conceptual Structures, ICCS'99*, number 1640 in LNAI, pages 262–271. Springer-Verlag.

[Sowa, 1984] Sowa, J. (1984). *Conceptual structures: Information Processing in Mind and Machine*. Addison-Wesley.

[Weed, 1971] Weed, L. D. (1971). The problem oriented record as a basic tool in medical education. *Patient Care and Clinical Research. Ann Clin Res*, 3(3):131–134.