Semantically Enhanced, Multifaceted, Collaborative Access to Cultural Heritage

Project IST-034984

D5.1 MOSAICA System & Application Specification

Deliverable

Editor:	Daniel Vladusic
Contributors:	XLAB; SHU, STRAIGHT, INESC, CYBERCULTUS, IDEA, Technion

Dissemination Level:	Public
Version:	1.0
Date:	5 May 2007

Executive Summary

Document Objective

This deliverable *D5.1 MOSAICA System & Application Specification* documents the outcomes of the tasks T5.1 and T5.2. Therefore it consists of user requirements and high level architectural design.

We first developed requirements for the individual technological tasks, negotiated collisions between them, and defined common system requirements for the overall project. The MOSAICA user requirements were based on the formulation and study of use cases, leading to the formulation of application scenarios. The user requirements and application scenarios were then used to define the functional system specifications in a verifiable and measurable form.

Prerequisite

IST-034984 MOSAICA DoW document

Summary

The leader of the D5.1 is XLAB therefore we had to summarize all the available information regarding the requirements of MOSAICA application. This task has initially been started by the WP5 integrator IDEA (Gadi Meyer), but has been transformed into this document. It was published on the project's forum and each one of the following partners: IDEA, INESC, XLAB, CyberCultus, Straight, and Technion had to correct/complete his part. SHU verified the requirements against collaborative requirements engineering processes and compliance of system's high level architecture with the service oriented architecture. The Technion verified the requirements against the user requirements and specification.

Eventually this document has laid the basis for the individual functional specification that together will form MOSAICA.

© 2007 MOSAICA

This document contains proprietary information of MOSAICA and shall not be reproduced or transferred to other documents, disclosed to others, or used for any purpose other than that for which it is furnished, without the prior written consent of MOSAICA.

The trademark and service marks of MOSAICA, including the MOSAICA mark and logo, are the exclusive property of MOSAICA, and may not be used without permission. All other marks mentioned in this material are the property of their respective owners

Contents

1 INTRODUCTION 6

2 USE CASES AND USER REQUIREMENTS 11

2.1 USE CASES 11
2.1.1 USER ENTERS MOSAICA WEBSITE 11
2.1.2 USER CHOOSES A VIRTUAL EXPEDITION 12
2.1.3 USER PERFORMS BASIC SEMANTIC SEARCH 13
2.1.4 USER PERFORMS ADVANCED SEMANTIC SEARCH 14
2.1.5 USER PERFORMS SEMANTIC BROWSING 14
2.1.6 SEMANTIC ANNOTATION 15
2.1.7 USER STARTS USING GIS/TIS 16
2.2 USER REQUIREMENTS 22
2.2.1 SEARCH AND RETRIEVAL 22
2.2.2 BROWSING 23
2.2.3 VIRTUAL EXPEDITIONS 23
2.2.4 EDITING 24
2.2.5 GENERAL 24

3 ARCHITECTURE 26

3.1 HIGH LEVEL ARCHITECTURAL DESIGN 26
3.1.1 APPLICATION FRONT-END 27
3.1.2 APPLICATION BACKEND 27
3.1.3 SEMANTIC LAYER 28
3.1.4 WWW 28
3.2 APPLICATION STACK INTERACTION 28
3.2.1 SEMANTIC LAYER COMPONENTS 30
3.2.2 GIS/TIS 41
3.2.3 VIRTUAL EXPEDITIONS 50
3.2.4 P2PCMS 54
3.3 INTERFACES (SYSTEM REQUIREMENTS) 61
3.3.1 APPLICATION FRONT-END 61
3.3.2 APPLICATION BACKEND 63
3.3.3 SEMANTIC LAYER 64
3.3.4 GIS/TIS 65
3.3.5 DISTRIBUTED CONTENT (P2P LAYER) 65

4 CONCLUSIONS 67

5 REFERENCES 68

1 Introduction

1.1 Purpose

This deliverable *D5.1 MOSAICA System & Application Specification* documents the outcomes of the tasks T5.1 and T5.2. Therefore it consists of user requirements and high level architectural design. As the user requirements reflect what the system should do, we structured them according to their functionalities which are also reflected in the high level architectural design.

This document specifies the user requirements, as seen by the technological partners. These requirements present the technical capabilities of each partner and their view on the MOSAICA application to the content management and pedagogical partners. They will review these capabilities and add or remove the requirements.

This document also portrays the high level overall system architecture and its components, provided by the technological partners of MOSAICA. Eventually, all requirements associated with the same component will be grouped together.

The MOSAICA user requirements will be based on the following: The technological partners and the formulation and study of use cases, leading to the formulation of application scenarios. The user requirements and application scenarios will be then used to define the functional system specifications in a verifiable and measurable form. Collaborative requirement engineering process has been applied in the formulation of the user requirements; the detail of which has been mentioned in the following sections.

1.2 Principles

In the system requirements' perspective, MOSAICA is a complex multifunctional online system constructed from three layers. Enabling Basic and advanced semantic search, as well as, semantic browsing within the cultural heritage domain are only the first tier. The presentation of virtual expeditions and fascinating stories is the second tier. The third tier is MOSAICA's unique capability for allowing users to upload and share their own items, hence enrich the collection of cultural heritage and its preservation.

MOSAICA structure and functionality should be based on a set of principles, determining its construction:

Flexibility - The system should allow the installation of different types of heritage items, that is, visual representations (photo, video, scanned documents, pictures etc.) with their adjunct text.

Authorization - The open nature of the system obliges a strong protection from users who aim to misuse the system or the protection of the privacy of users who want to share their information only within a certain community. The system should have a screening mechanism and allow authorized users to evaluate the uploaded items and configure the sharing level of each item. Users will be able to keep the heritage items totally private; share it with certain groups, or share it with all MOSAICA users.

Ease of use - The system should be designed to have easy to use interfaces for the following activities:

- Basic and advanced semantic search,
- Conceptual (semantic) browsing,
- Virtual expeditions (VE) on GIS/TIS maps (including games, and inquiry assignments) and,
- VE maker (uploading, assessing and validating Heritage items)

Multi-community - except for occasional users, MOSAICA users will be associated with a certain institution (school, museum, archive). The system should allow an assigned administrator to manage his group independently by generating users' templates, adding new groups and managing their contents.

MOSAICA's authorizations:

There should be **two types of users**, holding different authorization levels:

- End users (Explorative, Guided and/r Collaborative usage)- anyone that wishes to use MOSAICA.
- Administrator – appointed editors who will monitor the quality of the heritage items and the self-made VE.
- Editorial board – will be able to appoint the editors and when an item is in doubt (reject or accept) they will have the final decision.

End users - will be able to conduct basic and advanced semantic search, participate in virtual expeditions, write an expedition diary, play games, answer inquiry-based riddles, and respond to an online survey.

End users will also be able to act as collaborative users and create heritage items and/or their own VEs. They will be able to share heritage items, and receive and give commentary and remarks.

Administrators - will be appointed by the Editorial board and authorized to manage users and groups. Most important, they will be able to manage content, that is, accept, reject, or edit heritage items uploaded by the End users. The administrators will be able to open and manage forums or chat-rooms and have access to the expedition diaries and responses to online surveys.

Editorial board – a group of experts in Jewish cultural heritage that will have the power to appoint editors for different content themes (bible, Menorah, synagogues etc.) and/or different groups of users.

MOSAICA's subsystems for collaborative usage:

Each subsystem is responsible for different components and manages different areas of interaction.

The subsystems include:

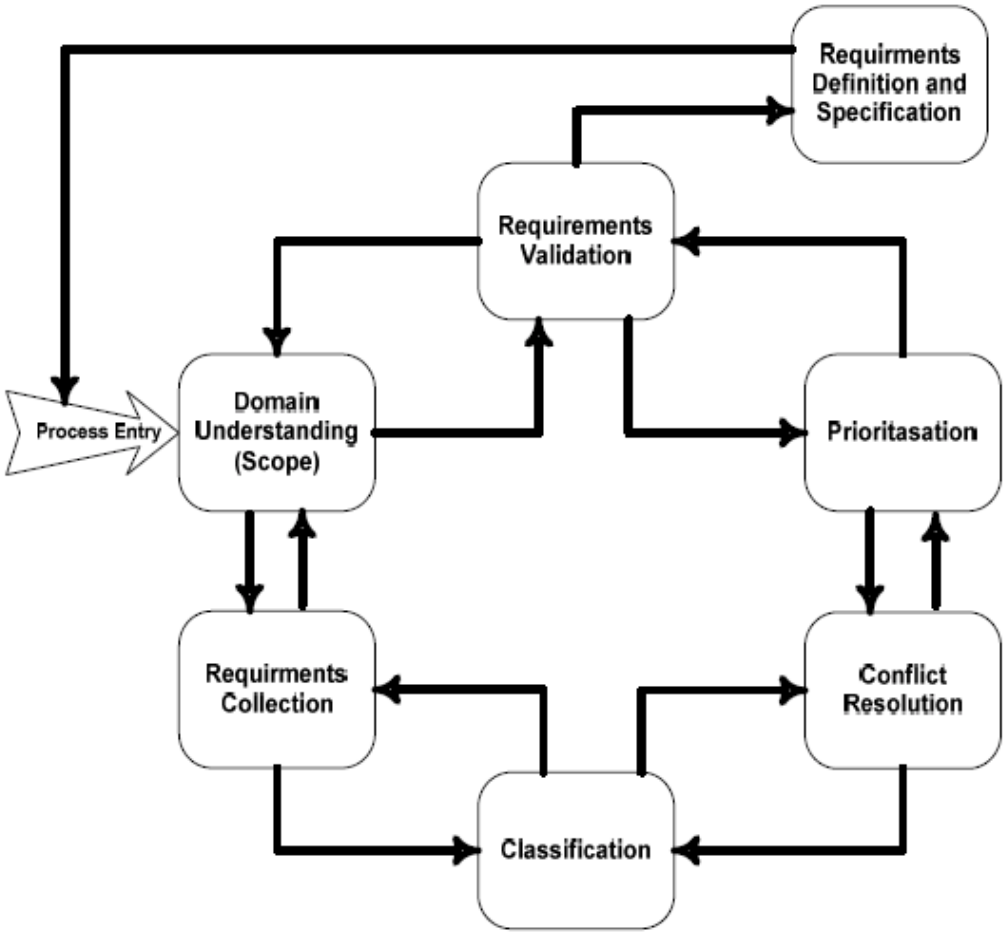
- Content Management System,
- Collaborative System for adding ones own Heritage items,
- Assessment and Editing System (for acceptance and rejection of uploaded heritage items),
- Administration System (for allowing different authorizations and monitoring the uploaded items).

1.3 Requirements Analysis Activities

The requirements analysis consisted of the following activities

- Elicitation
- Classification
- Validation
- Conflict resolution
- Prioritisation
- Definition and Specification

The requirements analysis processes identified above and the participation between these is graphically illustrated in the figure below.



MOSAICA User Requirements Analysis Process

1.4 Requirements discovery

In MOSAICA, the pedagogical partners are providing us with knowledge about the expectations of the targeted user group. Thus, we started describing how the system should work and through understanding with the pedagogical partners formalized these descriptions into storyboards and UML use case diagrams. These use cases were then analyzed and used to write the user requirements in formalized fashion.

Abstraction: Involves making useful generalisation to understand and articulate the rules that apply to a specific domain of knowledge to discover rules that are shared between domains.

Introspection: Method for understanding what properties the system should have through the knowledge worker envisioning the tasks the proposed system should perform.

Structured / Open Ended Interviews: These interviews begin by the interviewer asking questions based on some pro forma and then proceeding to probe for more detail in an open ended manner.

Facilitative Workshops: The primary means for requirement elicitation. The purpose of these is to use the group to generate ideas, identify requirements and provide a focused means of reviewing and questioning a related group of requirements. The starting point for these workshops will be a bounded set of functionalities which can be identified as a scenario. The reader is invited to note that, the scenarios and use cases within the document are utilised to provide a situation oriented perspective of the requirements to further the understanding of the requirements from differing worldviews. Specifically the user requirements for the MOSAICA were collected in three phases:

The initial phase constituted of contributions of the pedagogical partners and expectations from targeted user groups who were interviewed individually concerning their requirements for the MOSAICA functionality.

The second phase comprised of workshops and meetings, and its aim was to get together the user and developer communities to share and attempt to resolve differing perspectives.

The third phase was on MOSAICA forum where this document had some final revisions so that requirements were validated and agreed. The final activity of this phase is the prioritization of the requirements which is carried out concurrently with the peer review process.

1.5 Requirements Prioritization

Requirements are classified in to “mandatory”, “highly desirable” and “desirable”.

- Mandatory requirements specify requirements that MOSAICA is guaranteed to have on completion of the MOSAICA project. .
- Highly desirable requirements refer to those that will only be completed after the mandatory requirements have been fulfilled.

- Desirable requirements refer to those that will only be completed after the above have been fulfilled or are documented for future iterations of MOSAICA.

1.6 Document Organization

The document is organized as follows: First (subsection 2.1), we present the uses cases with storyboards and UML diagrams. These present the idea of how the system should work and what we think the user should be able to do. We then formalized the so-obtained use cases into actual user requirements which are given in the 0section 2. Next, we present the proposed architectural design (section 3) and finally, we give conclusions (section 4).

2 Use Cases and User Requirements

In this section we describe the Use Cases and User Requirements for the MOSAICA application.

2.1 Use Cases

2.1.1 User enters MOSAICA website

The user enters the MOSAICA web site. His profile is set to guest – all the menus, links are set for this profile. If he logs in with a different profile the menus, links (i.e., possibilities) change accordingly. For example, if he logs in as an administrator, he can add/remove accounts, open forums, edit ontologies, objects, (see section 1.2) these web pages (possibilities) are not available for other profiles.

When the user's profile is set, he can proceed to one of the following:

1) Virtual Expedition (including games, inquiry-based assignments, forums, and a survey)
2) Basic/Advances semantic search
3) Semantic browsing
4) GIS/TIS navigation
5) Semantic annotation (cataloging)
6) Creation of VE
7) Participating in a forum

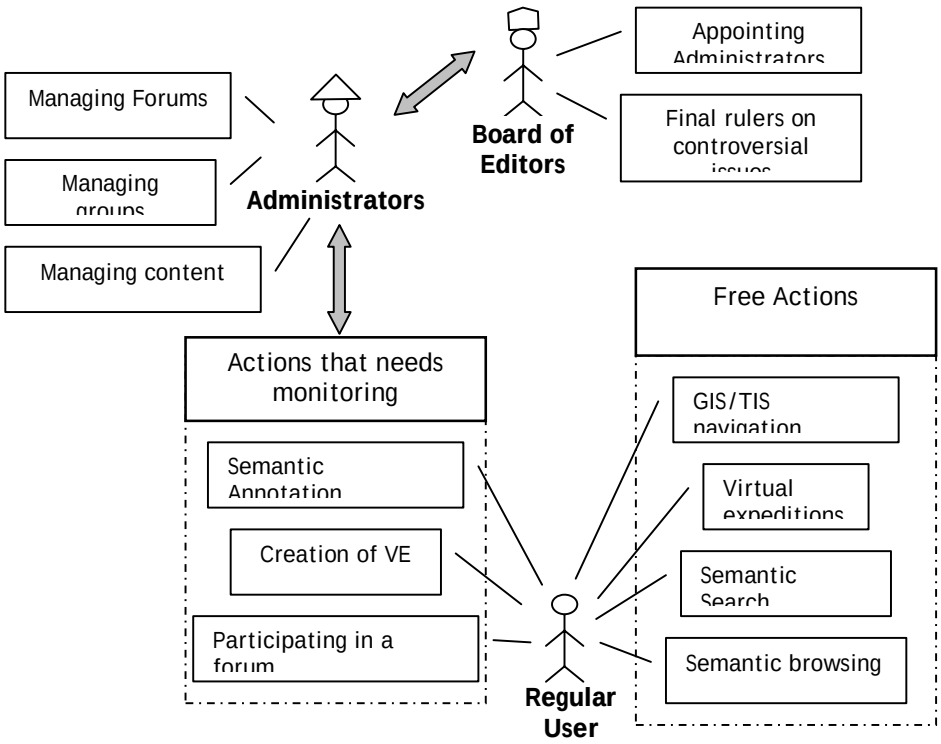


Figure 1: MOSAICAs' three authorization types

It is important to note that during searching for objects and playing virtual expeditions, the user has at his disposal GIS/TIS maps and results of the intelligent presentation. The first module enables him to position the object in space-time frame and, additionally, to search for co-located semantically tagged data, while the second module always shows immediate neighbors (according to semantic) of the current object.

2.1.2 User chooses a Virtual Expedition

User chooses a Virtual Expedition (VE) from a variety of Virtual Expeditions divided by topics and objectives. The VE is based on Semantic Search results. The VE path takes the user to a journey between "data" junctions in that are marked as active buttons and/or hypertext. In that case, sliding the Mouse arrow on top of a button, it will present a short text on the hidden item. Clicking on an active button will result in the appearance of a 'heritage item (text, picture, video or audio files) or a link to a related website that will be opened as a separate window.

Different Graphical User Interface will be used for the VEs:

1. A Simple VE includes a regular HTML page that contains text, pictures and hyperlinks.

2. A Timeline VE consists of a series of active buttons arranged on a chronological timeline ruler. Clicking on an active button opens a "heritage item" (that is, text + pictures, and/or video, audio).

3. A Map VE consists of hot spots that by clicking on them, a heritage item appears.

4. A Genealogy VE consists of a family tree each branch contains the name of a person. Clicking on one of the names opens a heritage item with the persons' relevant data (text, picture, video, audio, etc.)

5. A Drawing VE consists of a picture with details that some are actually hot spots. Clicking one of the hot spots opens a heritage item.

6. A Gaming VE consists of games such as Jigsaw puzzles, matching assignments and questions. For this we can use Thinking World platform.

7. A Complex VE integrates all of the above templates into one. It can be in the format of different doors that opens (like in our demo) or tabs that leads from one template to another.

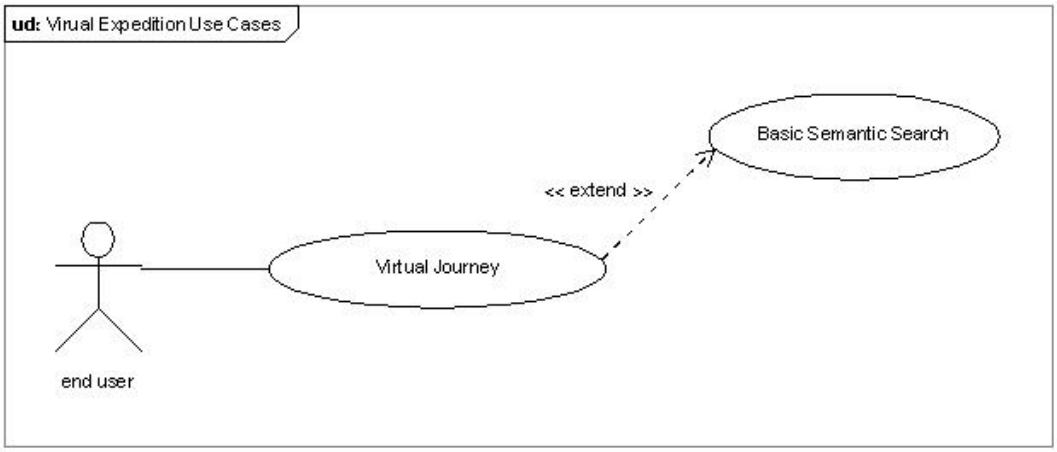


Figure 2: User chooses to go on a Virtual Expedition

2.1.3 User performs basic semantic search

Keyword Querying (Basic Semantic Search) - User submits a query in natural language (either one or more keywords, or a short, possibly even incomplete natural language sentence). The query is transferred to the Natural Language Processing (NLP) engine (embedded within the Semantic Layer), where it is tokenized and morpho-syntactically normalized. The processed query is further transferred to the Peer-To-Peer (P2P) infrastructure, where it is transformed into hashkeys, which are routed to the peer nodes storing matching hashed indices.

The distribution follows parametrized instructions provided with the query, and it could target: (1) indexed ontology concept labels, (2) indexed ontology concept comments, (3) full-text document index (only in case of textual resources), (4) indexed metadata (primarily the content description fields, such “keyword” field in the Dublin Core Metadata Initiative (DCMI)), (5) indexed free-text annotations, etc. These instructions regarding which indices should be targeted could be set either generically by the preset system configuration (i.e. identically for all queries), or from the user interface specifically for each submitted query.

Depending on the targeted indices, the following sets of search results are retrieved from the P2P infrastructure: (1) list of ontology concepts with matching labels or comments, (2) list of documents containing matching keywords, (3) set of metadata containing matching items (depending on the selected metadata schema, and the selected metadata fields), (4) set of matching free-text annotations, etc.

If the submitted query contains multiple query keywords, the set-intersection of search results for each individual query keyword is performed within the P2P infrastructure, in order to obtain only search results relevant for all query keywords (the set-intersection is still a research issue that will be further elaborated within the project's lifetime). In case of retrieved ontology concepts, rather than merely presenting the matching concepts, logical inference is performed, in order to create a logically consistent and complete presentation of the retrieved concepts. This action could include: (1) generalization grouping concepts under one or more generic concepts, (2) specialization including additional concepts providing more specific meanings, (3) set intersection to elucidate additional concepts that were not directly matched by the labels or comments, but are of relevance to the query, etc. (The complexity of the logical inference for semantic search is still a research issue that will be further elaborated within the project's lifetime.) In the final step, the retrieved search results are rendered for display by the front-end application.

2.1.4 User performs Advanced Semantic Search

Advanced Keyword Querying (Advanced Semantic Search) - User submits a query using the advanced semantic search interface (which extends the basic keyword querying specified in the previous use case) that includes: (1) text box for entering query keywords (i.e. basic keyword querying interface), and/or a dropdown list containing suggested query keywords, (2) form fields matching applicable metadata schema fields, (3) form fields reflecting applicable ontological relations, etc. User is allowed to formulate a complex query by filling in the displayed fields, or by selecting data input from preset dropdown lists. The submitted query is parsed in the backend infrastructure, and transferred to the P2P infrastructure. P2P infrastructure transforms query request into hashkeys, routes each query element to the suitable index, and retrieves the search results. Search results are set-intersected within the P2P infrastructure, in order to obtain the final set of results (the set-intersection is still a research issue that will be further elaborated within the project's lifetime) that is then rendered for display by the front-end application.

2.1.5 User performs Semantic Browsing

User is presented with a hierarchical structure displaying all applicable concepts in ontology. User browses through the hierarchical structure selecting concepts of interest. With each selection a list of associated resources is displayed for selection, as well as a structure of additional ontological relations and concepts associated with selection (i.e. ontological assertions in which the selected concept is the subject, a relation is the predicate, and another concept or a value is the object).

2.1.6 Semantic Annotation

User selects a resource (either by submitting it for upload, or by specifying the URI of the existing resource in the system), and an interface displaying annotation options is displayed. The annotation options are created dynamically from the relations in ontology in a step-by-step approach. In each step a set of potential relations (properties) is displayed followed by either a dropdown list allowing selection of ontology concepts, or by a text box allowing typing data. Depending on the selections in the previous step, the options for the next step are dynamically created. In the final step, the composed annotations are submitted to the backend application. The composed annotations are parsed, and submitted to the P2P infrastructure for hashing and storing.

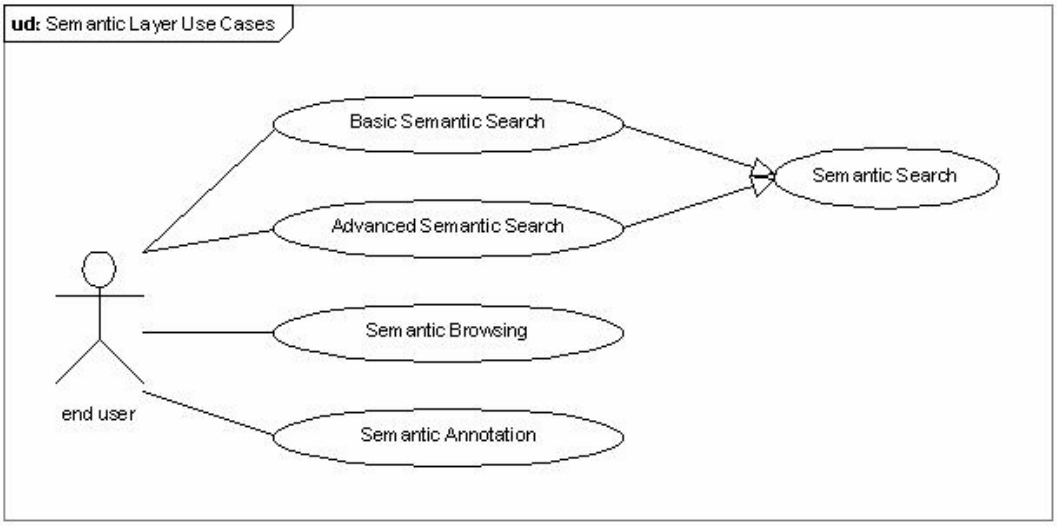


Figure 3: User is making a Semantic Search, Semantic Browsing or Semantic Annotation

Publishing ones own content – we need a system that will allow users to create their own "**heritage items**": 1. clicking on a button "create your knowledge item" will present a text box will be opened, allowing the user to write a short paragraph and attach a related file (photo, picture, scanned letter, video) (see QSIA's screen shot) 2. clicking on a button "add a new item" will add the item to a pool of items waiting for the approval of an editor (a school teacher, a museum employee, or peer users). Once read and approved, the editor clicks on an "approved item" button and the uploaded "knowledge items" – the visual and textual files will be displayed.

To summarize**,** users (in the collaborative mode) will be able to:

1. Upload visual features with accompanied text
2. Choose relevant key words from a scroll down bar
3. Add relevant key words
4. Add the date/time era
5. Mark a relevant location on connected to GIS map
6. Indicate the source of the item (private collection, museum, etc.)

7. Indicate identifying data (name and the group he belongs to, i.e. organization, school and class, community etc.) and contact information (email).

8. Indicate the general theme and sub-theme that the uploaded heritage item belongs to (providing a scroll down bar is recommended).

[Actions 6-8 are important for the appointed editors. It will allow them to track their users and monitor the uploaded content].

Peer-assessment

We need a system that will also allow users to respond to the heritage items presented by their peers. A system that will allow users to open a heritage item read it and comment on it. Commenting on each others' work would ease the editors' workload and bring further involvement among users.

Administrators' monitoring of uploaded heritage items

The administrators must be able to select heritage items that are under their responsibility from a pool of items. Their selection will be by username, group, or theme.

The administrators must be able to see the uploaded heritage item, edit it, accept or reject the item. One accepted, the item must be presented in MOSAICA's content inventory.

2.1.7 User starts using GIS/TIS

The user has selected an object and wants to see its location (current or original). To this end he selects to use GIS and TIS facilities of the MOSAICA application. These components take the currently selected object and use its semantic description to show it on the map.

Another way of using GIS-based maps is having a collection of maps with active buttons on them. When the user clicks on an active button a heritage item (picture+text) pops up in a separate window.

It is important to note that sliding the mouse on top of the buttons should result in the transformation of the arrow into a hand figure (or any other icon for this matter) presenting a short description of the item.

The user can select geographical area whether by clicking on a location and defining the distance or by dragging a diagonal line (to make a square). The relevant results can then be sorted by the semantic search tools.

VE will use GIS/TIS as a part of the expedition, and as assignments such as locating a place on a blank map.

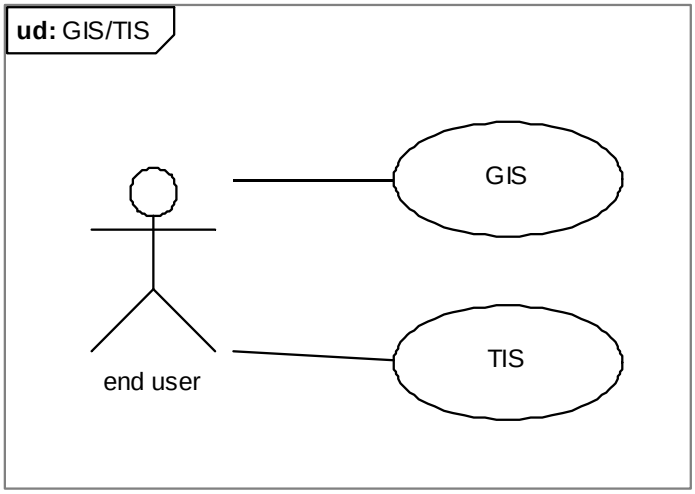
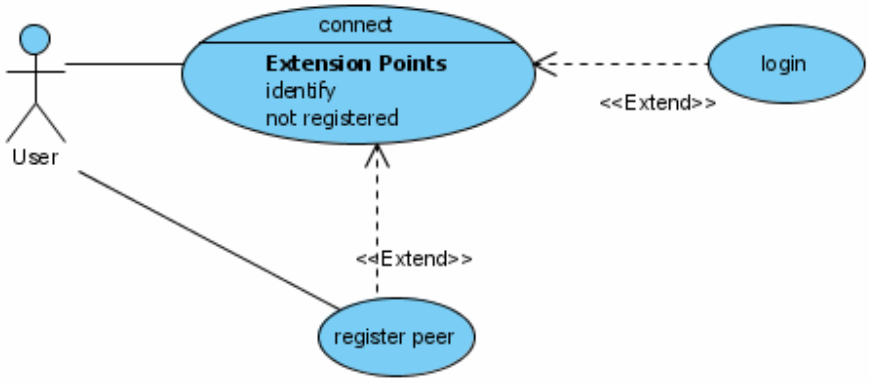


Figure 4: User selects GIS/TIS component for the current object

2.1.7.1 Bootstrap



- User wants to connect to Mosaica's peer-to-peer network
- User clicks the connect button
- If peer not registered, user fulfills registration form and submit it to the system
- If user pretends identify himself he can submit login information using the login interface else it will be attributed to him the default identification - anyone
- If user chooses to connect as an administrator, he must provide login information through the login interface

2.1.7.2 General Use Cases

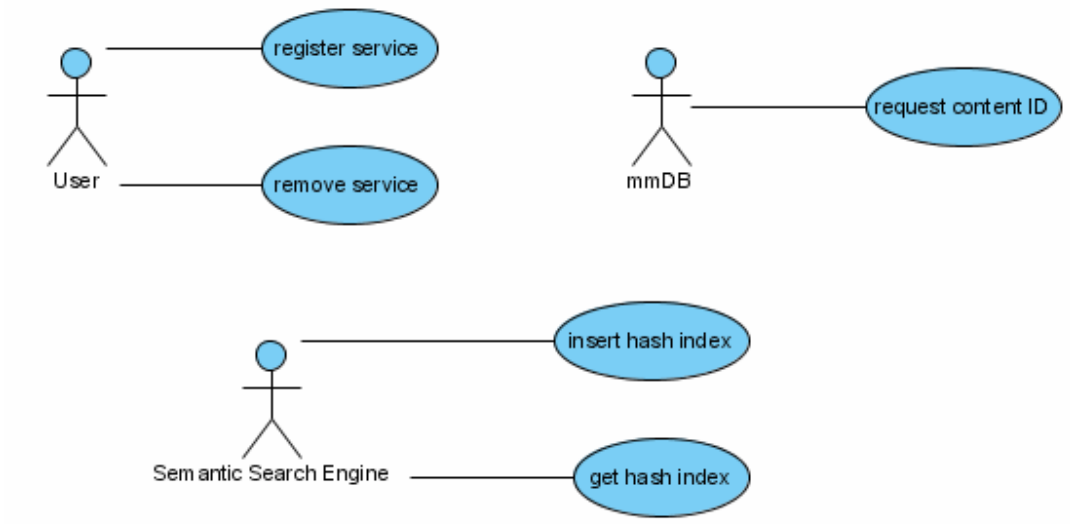
2.1.7.2.1 Precondition:

- Peer is connected to the system
- User is identified
- Administrator is properly logged in the system

Use case name	Overview	Initiating actor
Register content	The semantic annotator sends a SOAP message to the p2pCMS requesting the registration of new content. Semantics and syntax of the message to be defined.	Semantic annotator
Delete content	The semantic annotator sends a SOAP message to the p2pCMS requesting the deletion of existing registered content. Semantics and syntax of the message to be defined. The message must have a valid credential for content deletion.	Semantic annotator
Request content	The semantic browser or the semantic annotator sends a SOAP message to the p2pCMS requesting content. Semantics and syntax of the message to be defined.	Semantic browser Semantic annotator

Search content	The semantic browser sends a SOAP message to the system with a simple keyword query. Semantics and syntax of the message to be defined.	Semantic browser
Manage group	The system administrator logs in the p2pCMS using the p2pGUI and selects the "manage group" option. A window appears for him to insert valid credentials. Specific functionalities to be defined.	Administrator
Manage content	The system administrator logs in the system and selects the "manage content" option. A window appears with a registered content list not monitored. Administrator can visualize content and remove content from the system.	Administrator
Manage rooms	The system administrator logs in the system and select the "manage chat" or "manage forum" options and a window appears for him to delete room. He also select the "create chat room" or "create forum" options to create a new room.	Administrator
Join group	User selects the "join group" option. User is provided with login interface for authentication. User gets access to all resources posed in that group.	User
Join room	User selects the "join chat" or "join forum" options. User is led to choose the group where he wants to join. If user isn't connected to that group, a login interface appears for him to identify himself.	User

There is some use cases that were considered which are postponed for further development and discussion.



"Register service" and "Remove service" use cases provide a mean for users to have their own services. One such service could consist in provide internal database information into a Mosaica group.

The "request content ID" use case must be discussed with other partners about its utility.

A possible form of deploying the search functionality would be via the implementation of a DHT service. DHT has been used in structured p2p systems since 2001, to provide a lookup service that allows reduced search times in distributed environments. DHT provides a method to store in a distributed fashion <key, value> pairs, enabling to locate documents that provide an exact match to the searched key. Studies have shown that given a key, DHT systems can locate the corresponding document with only O(log N) hops (N is the number of nodes in system) [1].

Whereas DHT is very efficient in locating specific documents, it is not naturally fitted to support more complex queries such as those based on a combination of keywords or semantic-based searches where queries are expressed in natural language, as MOSAICA is aiming at. Some work is being developed worldwide to implement on top of DHT this kind of functionality. Some promising results [2] are based on the use of a special class of hashing functions called *Locality-Sensitive Hashing*. With this type of functions, If two documents are similar, then they will generate with a great probability the same hashing result. However, this research area is still quite recent and still presents many challenges.

As such, the actual implementation approach that will be followed for the MOSAICA distribution and search service is still to be decided. Different alternatives to be used on top of DHT based p2p systems will be investigated in WP3 under task 3.1. The outcomes of this study will then provide the grounds for taking a decision. This will be reported in deliverable D3.1 and consolidated in the final report of this task, deliverable D3.2.

It should however be emphasized that the use of DHT for the MOSAICA search service is only a matter of implementation alternative. Regardless of the implementation

approach adopted, the functionality that will be delivered by the search service will be the same, providing the necessary support to all relevant requirements identified in this document.

2.2 User requirements

2.2.1 Search and retrieval

2.2.1.1 Search

2.2.1.1.1 Basic search

1) The search interface should allow the end user to perform Semantic Search.
2) Semantic search **must** be allowed with keyword querying interface
3) Keyword searching **must** support (1) simple (multiple) keywords, (2) Boolean operators ("AND", "OR", and "NOT"), and (3) precise keyword sequences (included in the quotation marks).
4) Targeted indices for keyword querying **must** be specified either as a part of the system configuration, or the user **must** be enabled to specify them through the interface for each query separately, or both.
5) It is *recommended* that the keyword querying supports the Soundex or similar algorithm for phonetic matching.
6) It is *recommended* that the keyword querying allows selection of keywords from a set of applicable predefined keywords (for example, applicable ontology concepts).

2.2.1.1.2 Advanced (semantic) search

1) Semantic search **must** provide advanced keyword querying enabling filtering and ordering of search results by metadata, or by ontology relations.
2) Metadata fields **must** be selected dynamically, depending on the selection of the resource type (i.e. differently for textual documents, multimedia files, etc.), and/or of topics or domain.
3) Dublin Core Metadata Initiative (DCMI) schema **must** be included for all resource types.
4) Applicable ontology relations *should* be included into the advanced keyword querying interface. Selection of the applicable ontology relations *should* be performed dynamically, depending on the previous choices (possibly, by selection of topics or domains).
5) It is *recommended* that the advanced keyword querying interface allows formulation of complex logical statements for querying.
6) Advanced keyword querying **must** support multilingual interface.
7) Advanced keyword querying **must** enable filtering and/or ordering of search results by advanced keyword querying criteria (i.e. metadata and ontology relations).

2.2.2 Browsing

2.2.2.1 Conceptual (semantic) browsing

1) End user *should* be able to perform Semantic Browsing.
2) Semantic browsing **must** provide a hierarchic presentation of entire ontology, or its subsections.
3) It is *recommended* that the hierarchic structure is displayed either as a tree, or a spider diagram, or any similar diagram.
4) Clicking on a node in the hierarchical structure **must** display all associated resources.
5) Clicking on a node in the hierarchical structure *should* display additional ontology relations and ontology concepts applicable to the selected node (i.e. ontological assertions where selected concept is the subject, a relation is the predicate, and another concept or a value is the predicate). It is *recommended* that the display of the additional ontological elements is either textual, or as a spider diagram, or any similar diagram.
6) Nodes in the hierarchical structure **must** be collapsible.
7) Display of the hierarchical structure *should* support zoom-in and zoom-out allowing display of either more specific, or more generic sections of ontology.
8) User **must** be able to navigate from the results displayed by the Intelligent Presentation

2.2.2.2 GIS/TIS

1) GIS **must** provide a User Interface – View mechanism including means for selecting a specific era
2) End user *should* be able to perform GIS/TIS navigation.
3) GIS *should* display additional info. per selected coordinate
4) GIS UI **must** give the option of drill into the additional info.

2.2.3 Virtual Expeditions

1) End user *should* be able to go on a Virtual Expedition.
2) Virtual Expeditions (VE) **must** provide the functionality of selecting a VE Guide based on user preference of predefined by VE
3) VE **must** provide the functionality of choosing a VE from a list of available VEs
4) VE interface *should* support Soundex or similar algorithm for phonetic matching
5) The user **must** be able to record a Virtual Expedition.
6) The user **must** be able to define the "data junctions" of the virtual expedition.

2.2.4 Editing

2.2.4.1 Content contribution

1) End user *should* be able to catalogue items (URIs).
2) It is *recommended* that the data input is validated before submission for the correct data format (numbers, dates, etc.), as well as spell checked.
3) It is *recommended* that the annotation fields provide encyclopedic definitions wherever applicable.

2.2.4.2 Annotation

2.2.4.2.1 Semantic annotation

1) Semantic annotation **must** provide predefined fields for data input.
2) The selection of displayed fields for data input **must** reflect the resource type, the selected topic or domain, etc.
3) Interface for semantic annotation *should* take step-by-step approach, where data input in each step provides the required information for a dynamic selection of the annotation fields in the next step.
4) Semantic annotation *should* apply logical inference during the submission, in order to generate additional annotation associations (for example, by inferring a historical period from date and location).

2.2.4.2.2 Free text annotation

1) It is *recommended* that the semantic annotation provide predefined choices, rather than free-text input, wherever applicable (including, for example, date selection from a calendar).

2.2.5 General

2.2.5.1 Board of Editors

Fight propaganda and rumors –the board of editors will appoint people that are trusted to have pluralistic views and a strong sense of ethics. They will have administration authority. They will be able to manage (add, change and delete) groups (museums, communities, schools) users (the individuals within each group), and knowledge items (the contributions of each user).

Guidelines

The Board of editor will agree upon a guideline document that will be followed by the trusties.

2.2.5.2 Multilingualism

GUI:

1) The MOSAICA front-end application **must** support following languages for the end user:
   - English
   - French
   - German
   - Polish
   - Hebrew

Semantic search:

2) Keyword querying **must** support multilingual query keywords submission.
3) Language recognition of a query keyword *should* be performed automatically, and for each keyword separately.

Semantic annotation:

4) Semantic annotation **must** provide full multilingual support. Resources are annotated through their association with ontological concepts, and not textual strings.

3 Architecture

3.1 High level architectural design

The following two figures are presenting the high level architectural design of the MOSAICA application. Figure 5 shows the MOSAICA application stack, i.e. all layers from the presentation layer (application front-end) to the web services. Figure 6 shows the high level component overview of the MOSAICA application, as presented in DoW. In order to orchestrate these components we define high level interfaces in Section **Erreur ! Source du renvoi introuvable.**.

Sections from 3.1.1 to 3.1.4.2 will briefly present the MOSAICA application high level architectural design. This description will be divided according to the Figure 5.

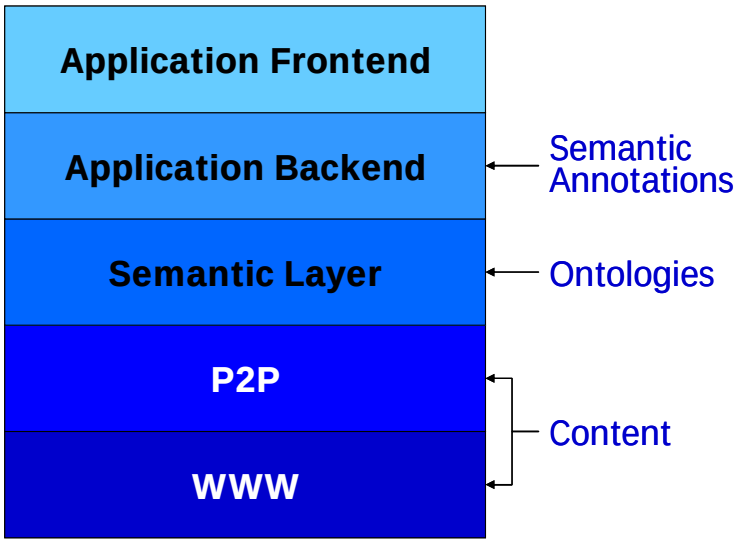


Figure 5: MOSAICA application stack

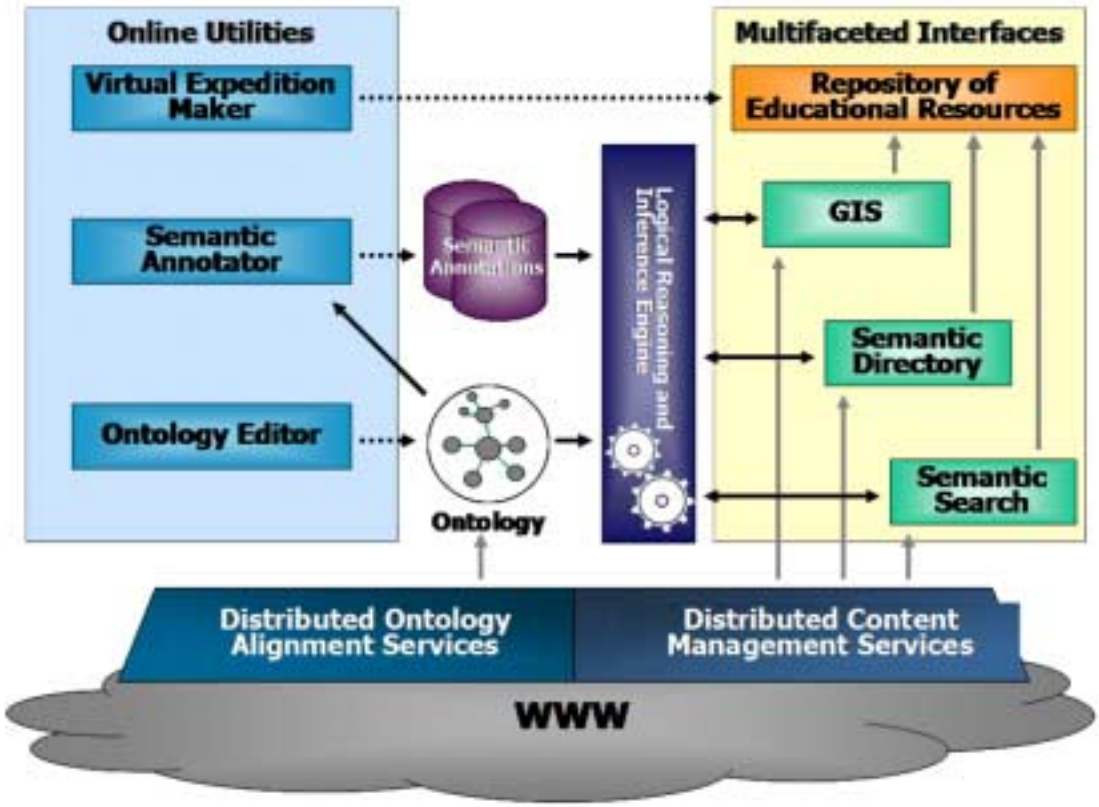


Figure 6: MOSAICA high level component overview

3.1.1 Application Front-end

The application front-end is responsible for handling of the user input and application output: responding to mouse clicks, entering text and rendering the results. It consists of predefined screen masks that are used according to current task the user is executing.

3.1.2 Application Backend

Application backend integrates all the MOSAICA services together. As all the services have already roughly defined interfaces, the backend has an important task of connecting their inputs and outputs together.

It is also responsible for transforming the input from the application front-end into form that MOSAICA services consume, e.g. XML files, SOAP messages. Thus, we can say that application backend orchestrates the MOSAICA services.

Another important task for the application backend is to guide the application according to the predefined workflows.

Currently we envisage the following components that will be included in the application backend:

1) Data Getters (responsible for retrieval of data)
2) Data Setters (responsible for storing data)

3) Workflow manager (responsible for guiding the application)
4) Virtual Expedition Player (responsible for replays of stored VEs)

3.1.3 Semantic Layer

One of the most important and most innovative layers of the MOSAICA application is the Semantic Layer. Use of this layer distinguishes the MOSAICA application from other web-based applications that may have similar content.

This layer is responsible for:

1) semantic search in the MOSAICA ontology
2) semantic annotation (updating the MOSAICA ontology)
3) indexing the end user's uploaded data

It essentially acts as a filter between user's queries and the database.

3.1.4 WWW

3.1.4.1 P2P

The P2P (or P2P Content Management System) is MOSAICA's storage system. It stores data, uploaded into the MOSAICA system in distributed fashion – it also provides data replication that will be used to ensure high availability of the data.

3.1.4.2 GIS/TIS

The GIS/TIS system provides maps of the selected coordinates. Moreover, it provides temporal maps, i.e. maps from different time eras.

3.2 Application stack interaction

The layers in the application stack interact by passing data between components found in each layer. The *Figure 7. Application stack interaction* depicts this interaction.

In the user space, the Mosaica Browser represents the main connection with the application. Mosaica browser is the actual HTML/JavaScript/… representation of the output, received from the GUI block. When the Mosaica browser is playing Virtual Expedition, it makes use of the VE Connector which runs in a separate process of the user space, but exchanges the user's input with visual output with Mosaica Browser. VE Connector retrieves its data directly from Semantic Layer, and acts as a local cache (i.e., storage). Furthermore, the VE connector and Mosaica browser exchange priority lists, where VE, running in the Mosaica browser, forms a priority list of items to download and VE connector returns the items that are downloaded already. Other items are added to the download queue using this priority as the download order. The user can also create virtual expeditions using VE Authoring tool, feeding the meta-data on the VE to the Semantic Layer. Creating of the VE with the authoring tool should make use of the Mosaica browser, which retrieves URIs of the items in the Mosaica. These items are then copied or dragged & dropped onto the VE authoring tool and inserted into VE as hot-links.

The application front-end consists of graphical user interface (GUI), which represents the data from backend in a consistent and user-friendly way. It also passes the user's

input to the application backend. The GUI block represents the transformation between front-end and the back-end. Here we can apply different styles, which are used when rendering the Mosaica application in the browser.

The application backend consists of the business logic layer (BLL), which binds various web services with the application front-end, ensures the data consistency and queries data for the front-end. Another important part of the BLL is controlling the application with workflows (i.e., finite-state automata). If one looks closely at the arrows, connecting various web services and BLL or GUI, it can be seen that web services sometimes do not communicate with the BLL but directly with GUI. The reason for such behavior is in the fact that each web service component already has its own GUI, which will be rendered in the Mosaica browser.

The fact that every web service has its own GUI also creates additional connections on the Web services part of the Mosaica application. Namely, if someone is working in the GIS/TIS GUI component, he or she will also get the data from the semantic layer, hence we need the connection between these two components.

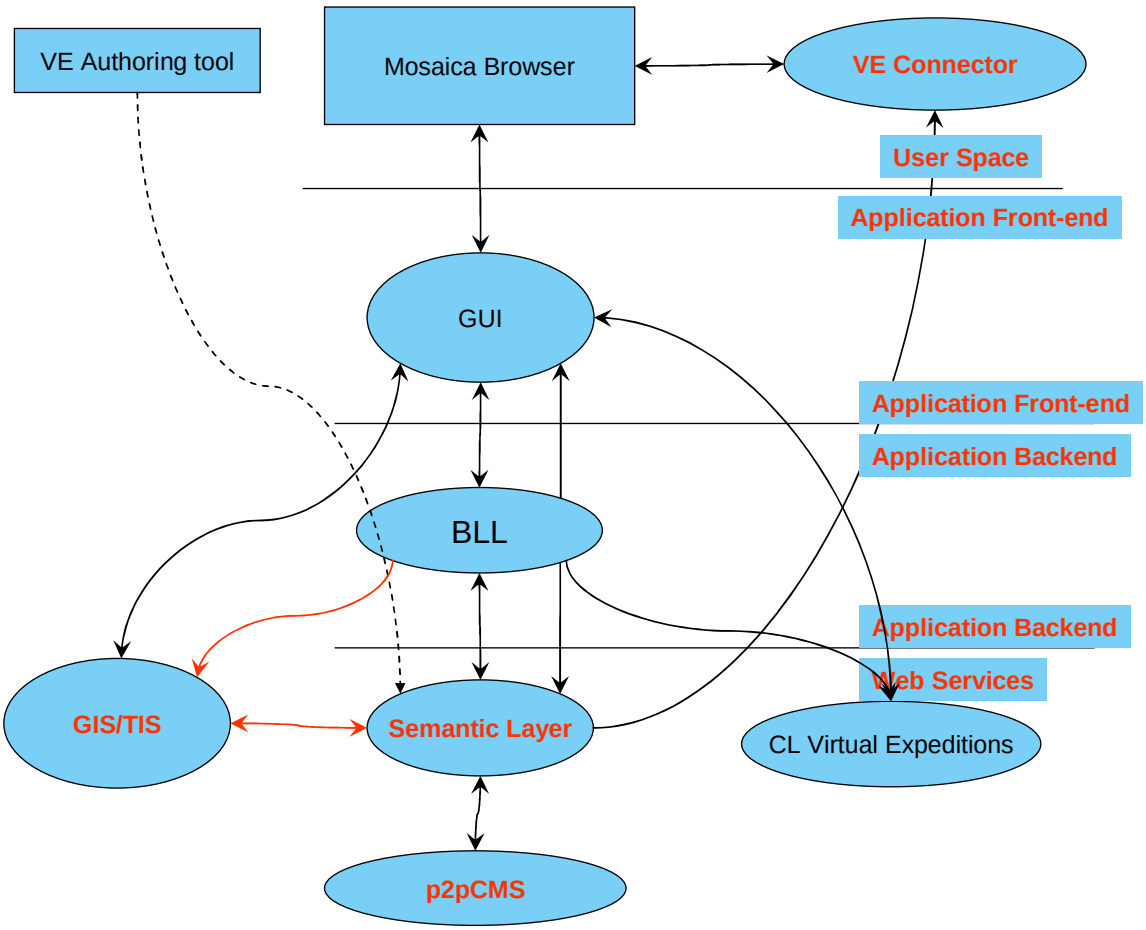
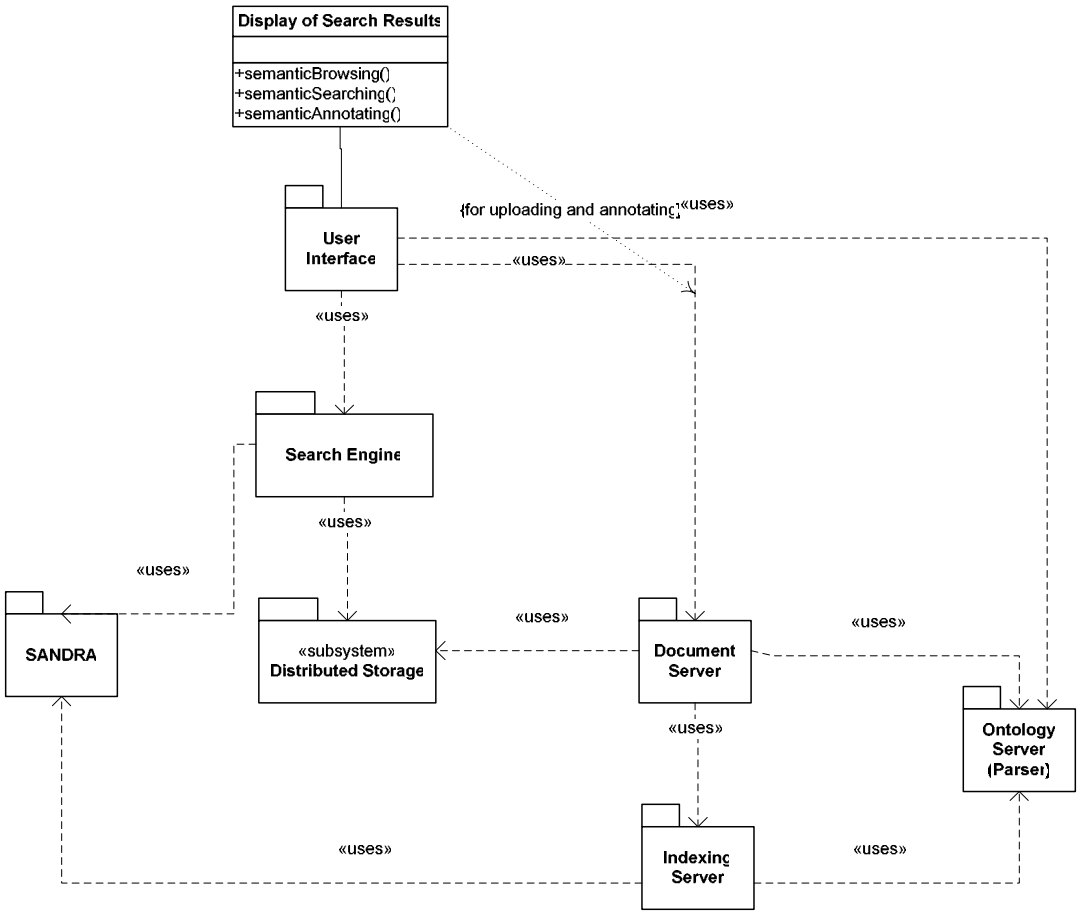


Figure 7. Application stack interaction

3.2.1 Semantic Layer Components

3.2.1.1 Package Structure and interaction

3.2.1.1.1 Package Structure



Above diagram presents package structure and interaction between basic components of MOSAICA Semantic Layer.

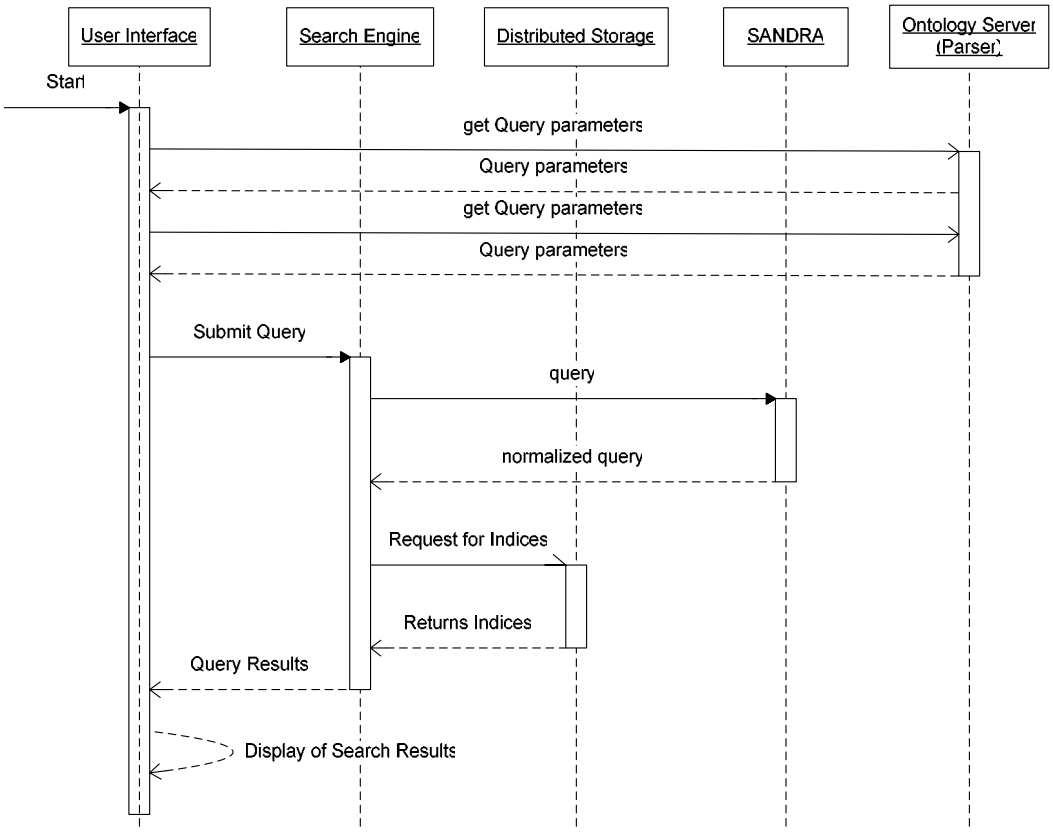
User Interface module, delivering semantic browsing, searching and annotating functionalities, uses Search Engine in order to submit queries and retrieve its results. For semantic annotating and uploading resources Document Server is used. User Interface application utilizes Ontology Server (OWL Parser) for retrieval of MOSAICA Ontology structure.

Search Engine calls SANDRA service for NLP of query strings. Also takes advantage of Distributed Storage subsystem to retrieve search results from stored indices.

Document Server uses OWL Parser to prepare metadata information for Indexing Server. It also stores indices returned by Indexing Server into Distributed Storage subsystem.

Indexing Server reads metadata using OWL Parser services and process it with NLP provided by SANDRA.

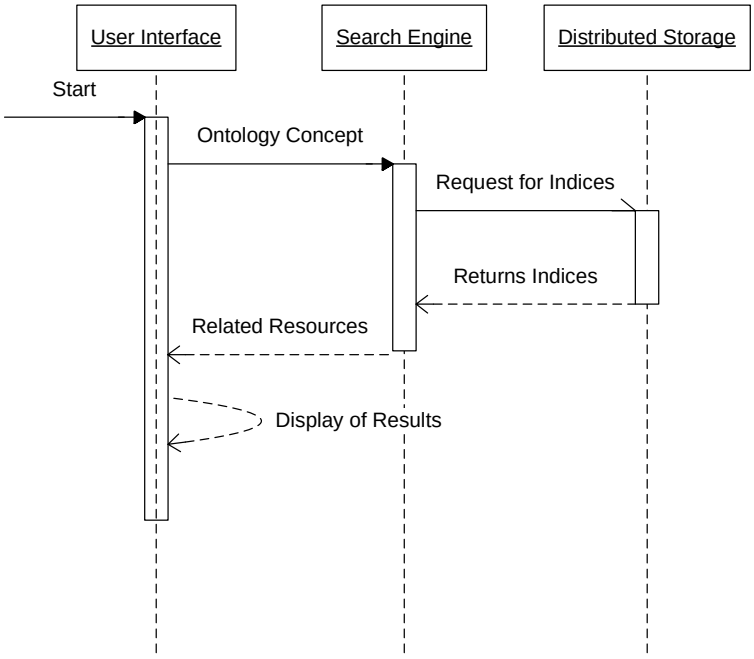
3.2.1.1.2 Semantic Search



Above diagram presents interaction between modules during process of submitting query and retrieval of its results which takes place in Semantic Searching.

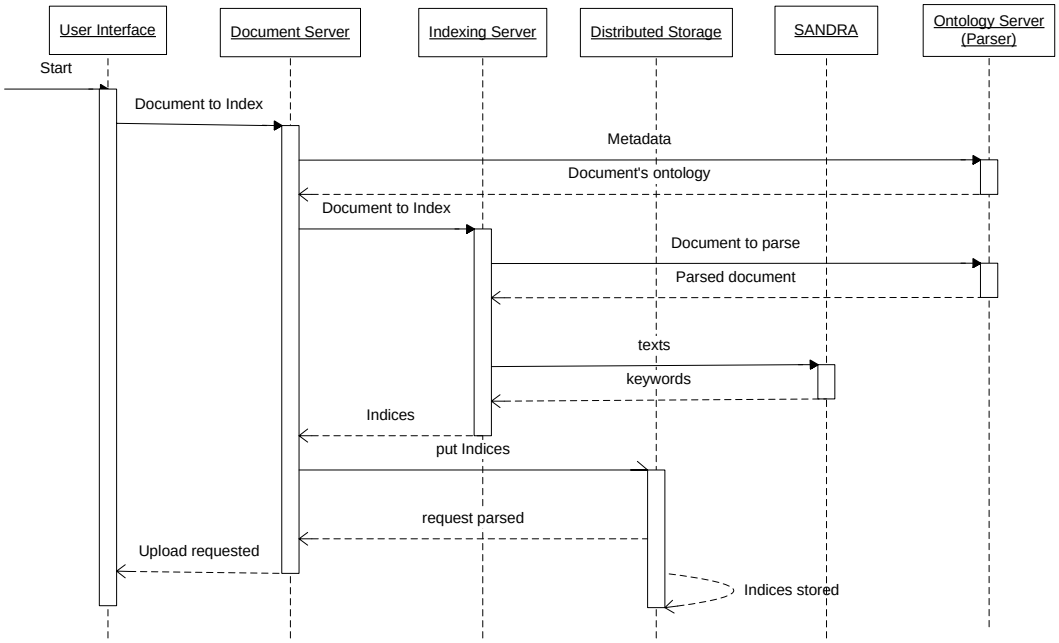
Query parameters are prepared with help of Ontology Server providing user with prompts. Then query is being submitted to Search Engine which normalizes it with SANDRA and retrieve data from Distributed Storage indices structures. Finally display of Search Results is performed.

3.2.1.1.3 Semantic Browsing



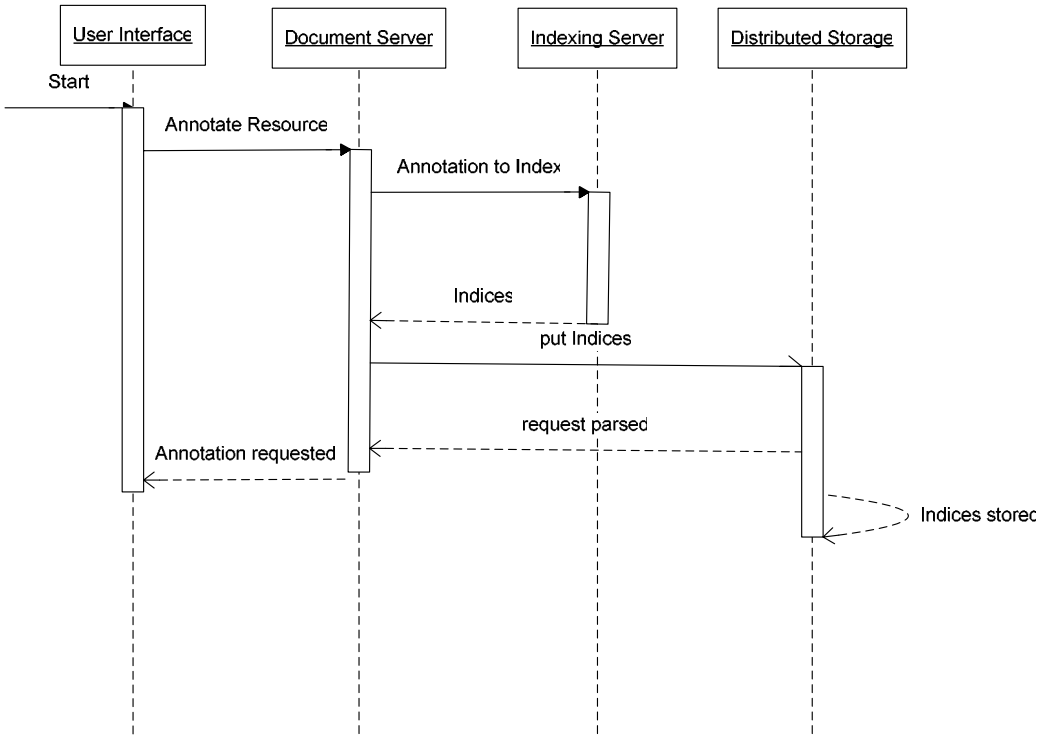
During Semantic Browsing, Search Engine retrieves Distributed Storage data concerning Ontology Concept requested by user. Results are displayed by User Interface.

3.2.1.1.4 Upload Document



Uploading document starts when user requests an document to index. Document Server assigns metadata for this file and sends it to Indexing Server which parses its input using OWL Parser and process it using SANDRA service. Resulting indices are propagated back to Document Server which requests putting them into Distributed Storage. Results of parsing are returned and indices are being queued for storing.

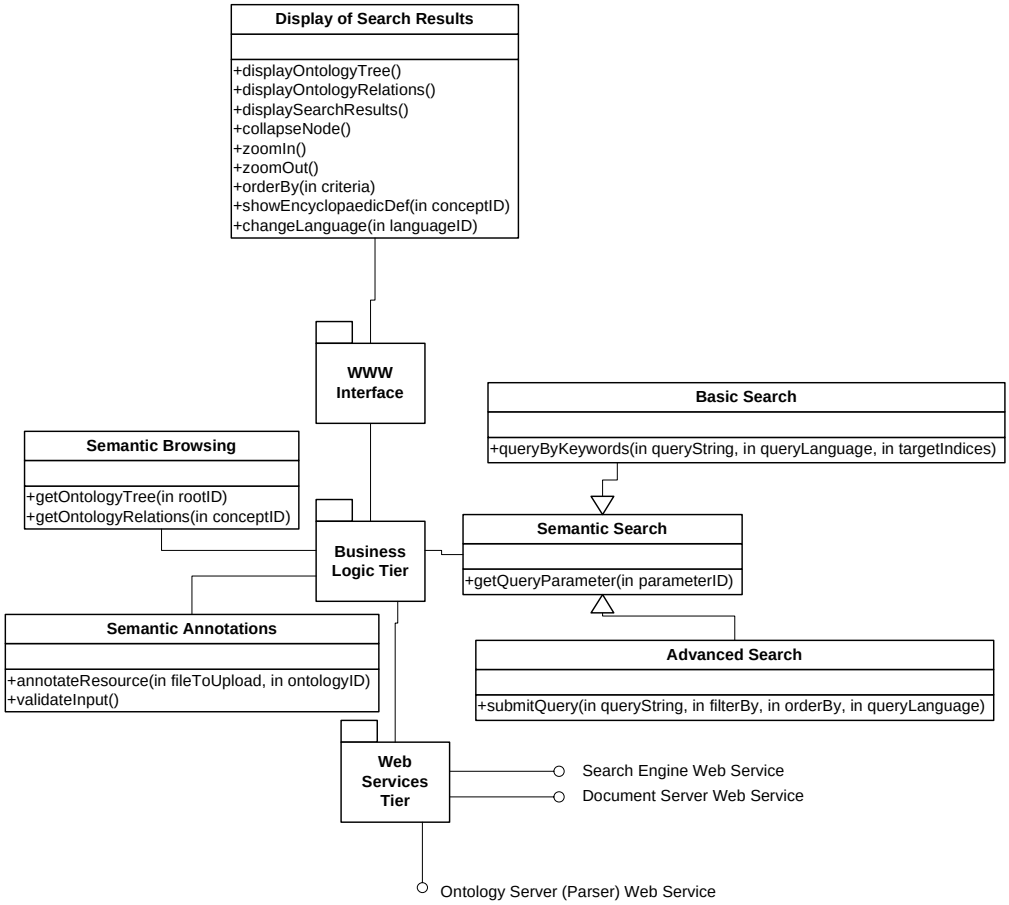
3.2.1.1.5 Semantic Annotation



When Semantic Annotation is requested, Document Server propagate user's request to Indexing Server in order to acquire indices which finally are being stored in Distributed Storage subsystem.

3.2.1.2 Application Structure

3.2.1.2.1 User Interface Web Application



Main MOSAICA Web Application will have 3-tier architecture:

- WWW Interface
- Business Logic Tier
- Web Service Tier

WWW Interface will be responsible for interaction with end user and it will call appropriate Business Logic modules:

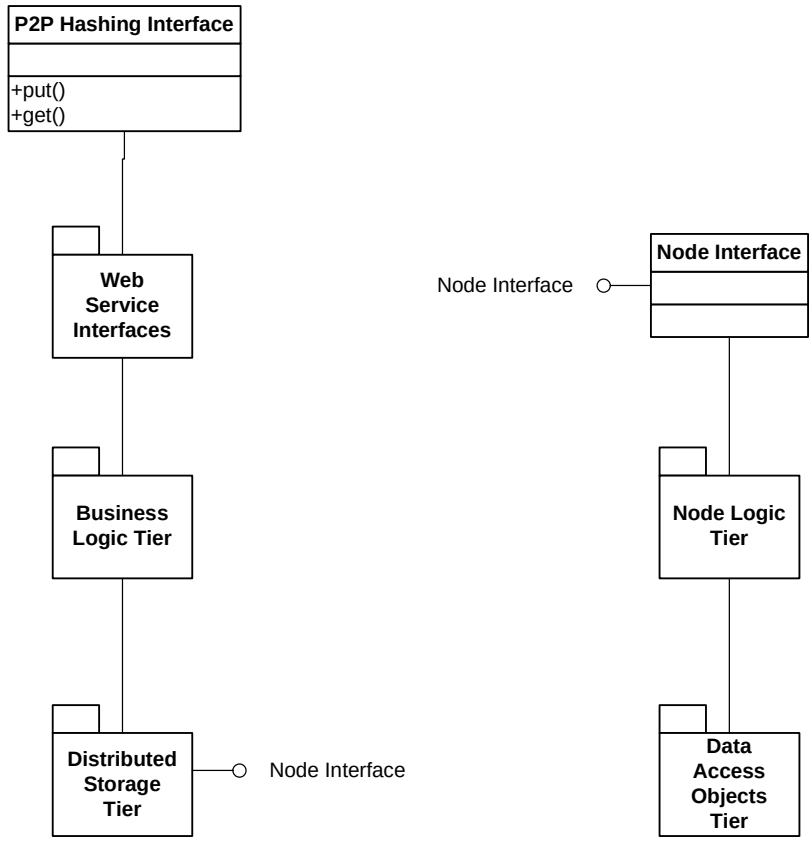
- Semantic Search (Basic and Advanced)
- Semantic Browsing
- Semantic Annotations

Those modules will use following Web Service interfaces (present in Web Services Tier):

- Search Engine WS
- Document Server WS
- Ontology Server (Parser) WS

It is important to avoid "cross-reference" dependencies between Business Logic modules.

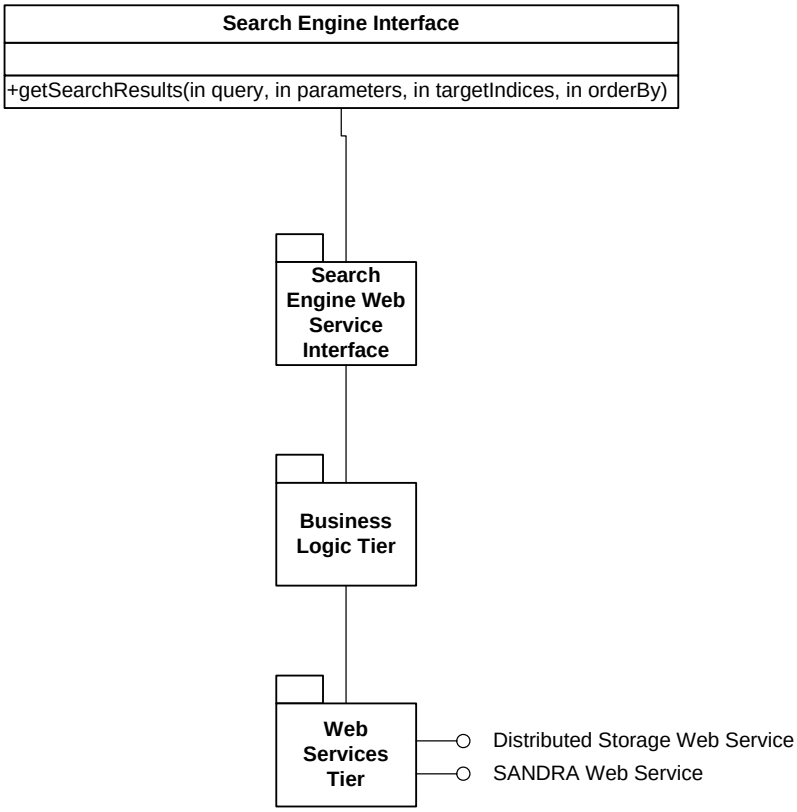
3.2.1.2.2 P2P Infrastructure



Peer-to-Peer Infrastructure will provide basic put/get interface. Structure of this Distributed Storage subsystem will be independent on other MOSAICA modules and will consist of subparts called Nodes. Interface between Super-Node and Sub-Nodes is not defined here as it is irrelevant to the whole system architecture.

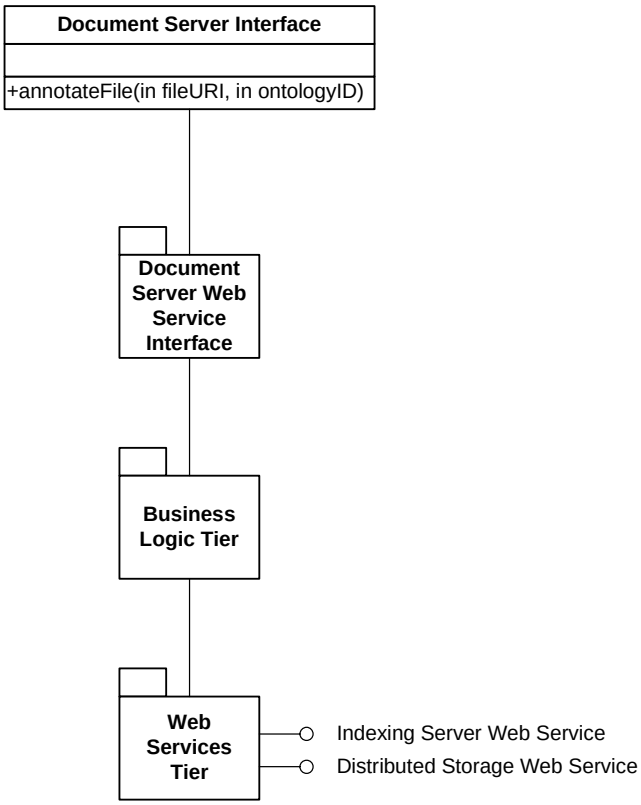
3.2.1.3 Semantic Layer

3.2.1.3.1 Search Engine



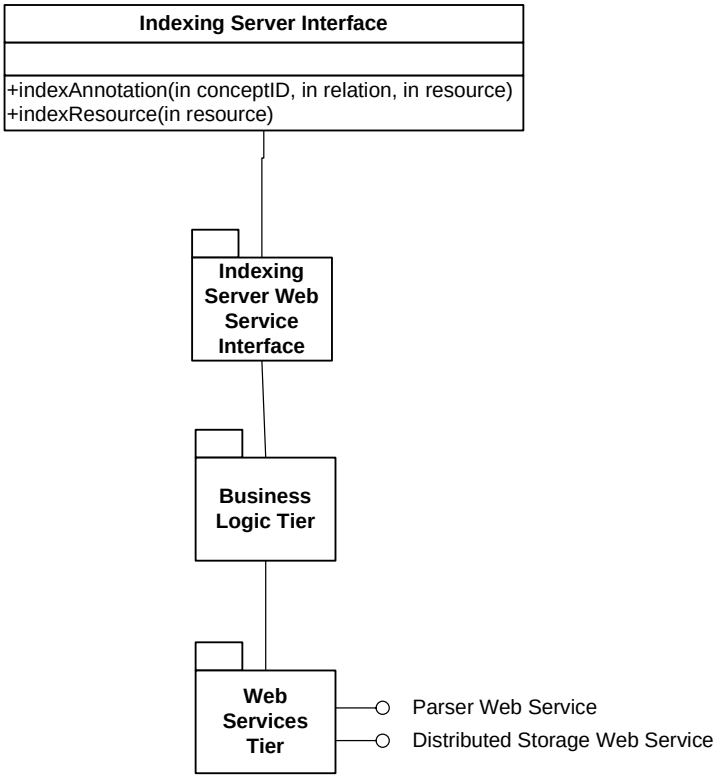
Search Engine module will provide interface for querying and search results retrieval. It will be possible to address target indices and specify ordering of results.

3.2.1.3.2 Document Server



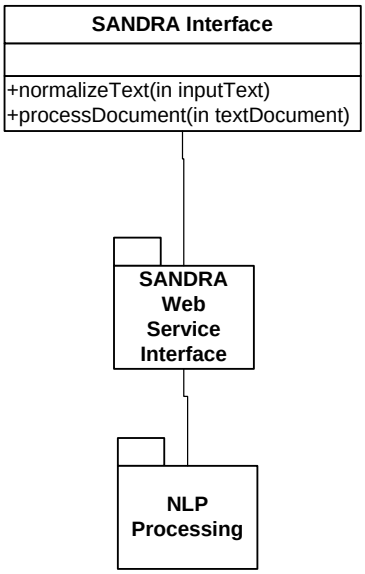
Document Server is responsible for annotating files. It's possible to annotate existing resource. Otherwise, file given by its URI will be uploaded first.

3.2.1.3.3 Indexing Server



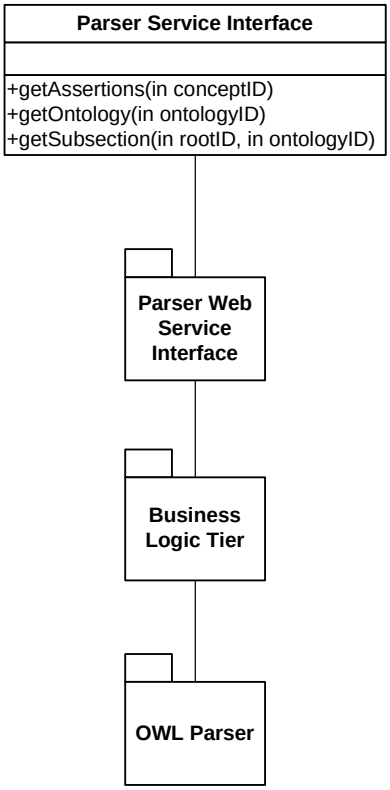
Indexing Server handles indexing and/or annotating of any existing resource.

3.2.1.3.4 SANDRA



SANDRA serves as NLP engine and provides functionality for normalizing texts or documents.

3.2.1.3.5 OWL Parser

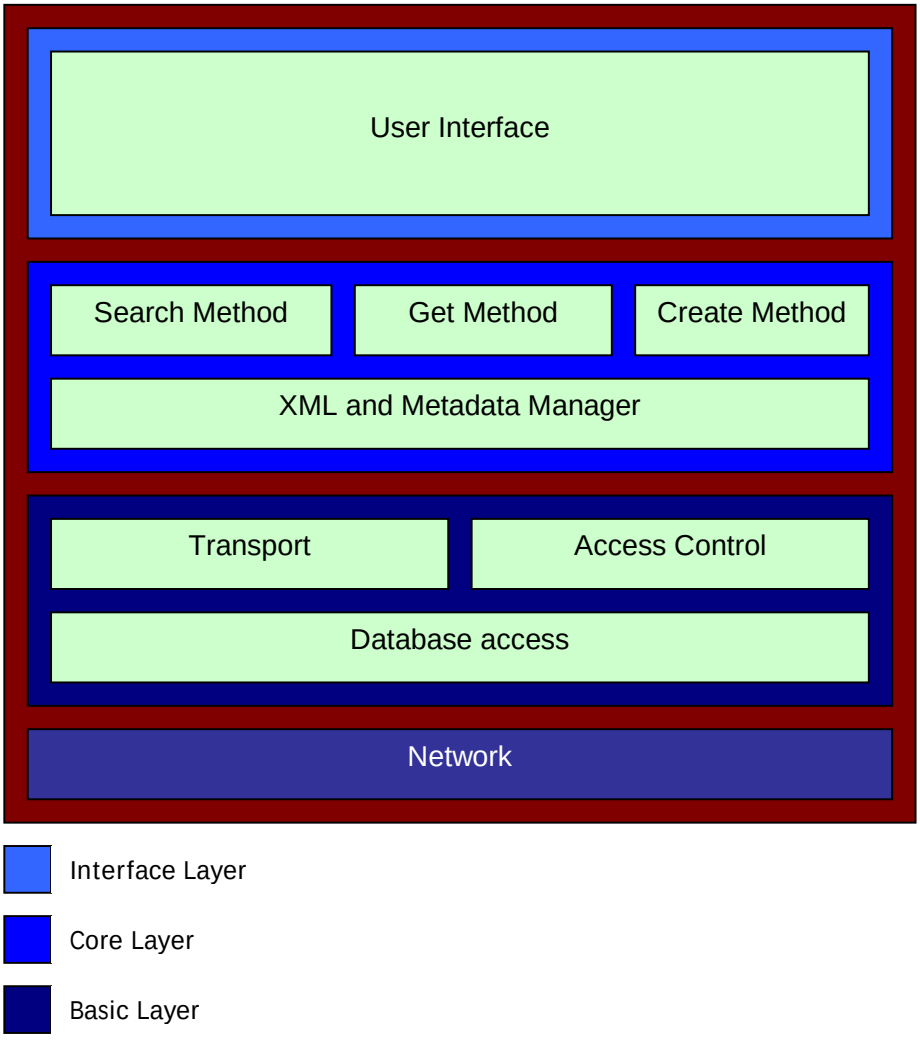


OWL Parser is a separate module for operating on OWL files. It will deliver assertions related to particular concept, subsection or full view of selected ontology.

3.2.2 GIS/TIS

3.2.2.1 High level architecture

3.2.2.1.1 Functional architecture



The GIS/TIS application is using easy simple client/server architecture. All the actions made by the user (simple user, administrator…) on the Graphical layer are transmitted to the application layer that contact the transport layer to know which method must be called on the server side of the application.

The transport layer contacts the server side that executes the needed command and return a simple XML in SOAP standard after the execution of the request on the server.

3.2.2.1.2 The Basic layer

- **Transport**: it implements the message transport between peers. The principal network protocol that will be used is the TCP/IP but it is possible to use other network transport protocols such as HTTP or UDP.
- **Database Access**: it implements the method used by the application to retrieve and save data in the database.
- **Access Control**: permits identify, authenticate users. It also provides system with user permissions to use some services. This service relies on the security service.

3.2.2.1.3 The Core layer

- **Search Method**: the search provides the system with search capabilities. It will take advantage of the Indexing Service to quickly find the content.
- **Get Method**: the get method provides specific method to retrieve data from the database and to display the objects located in the selected place.
- **Create Method**: the create method provides specific method to save the changes made by the user on the objects of a specific map (add, remove, move…)
- **XML and Metadata Manager:** this service provides functionality to manage the content of metadata return by the previous methods.

3.2.2.1.4 The Interface layer

The interface layer will provide a user interface to create (editor mode) or just to display objects (visualization), thus allowing them to use the value delivered by this MOSAICA subsystem.

Communication with external subsystems will be made via a Web Services interfaces using SOAP to exchange messages, as agreed within WP5.

3.2.2.2 Package Structure and Interaction

The diagram in figure 1 represents the services' structure of the GIS/TIS, illustrating the interaction that will be established between the services.

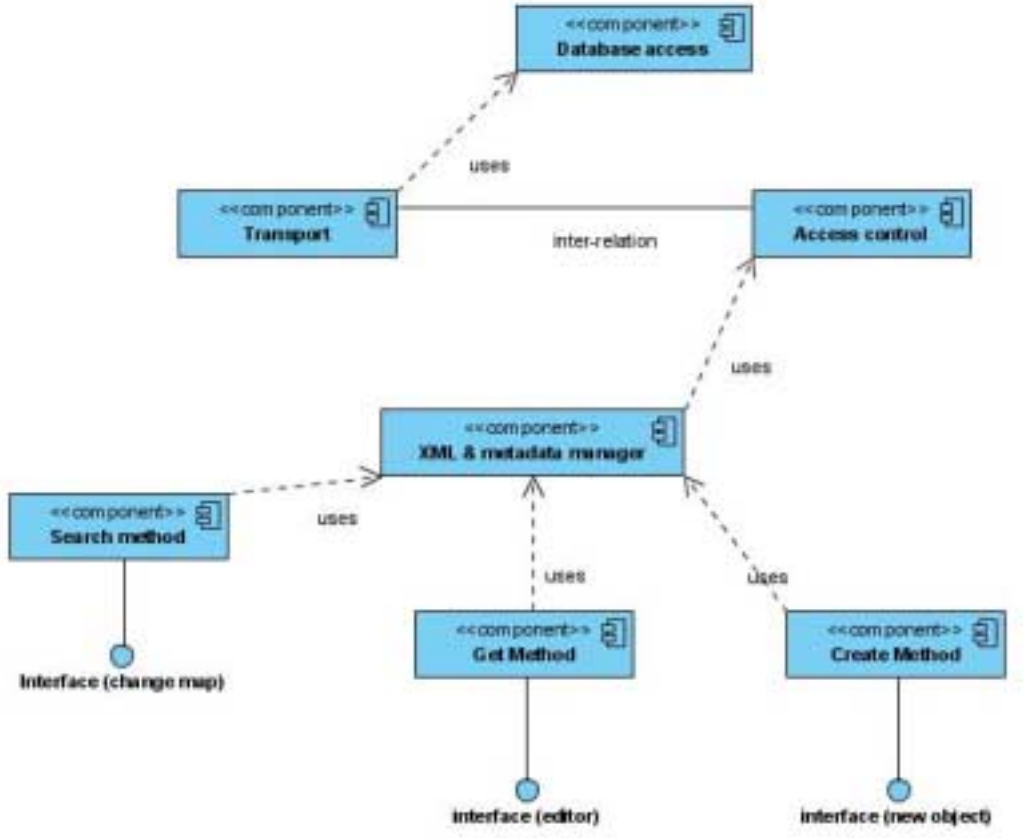
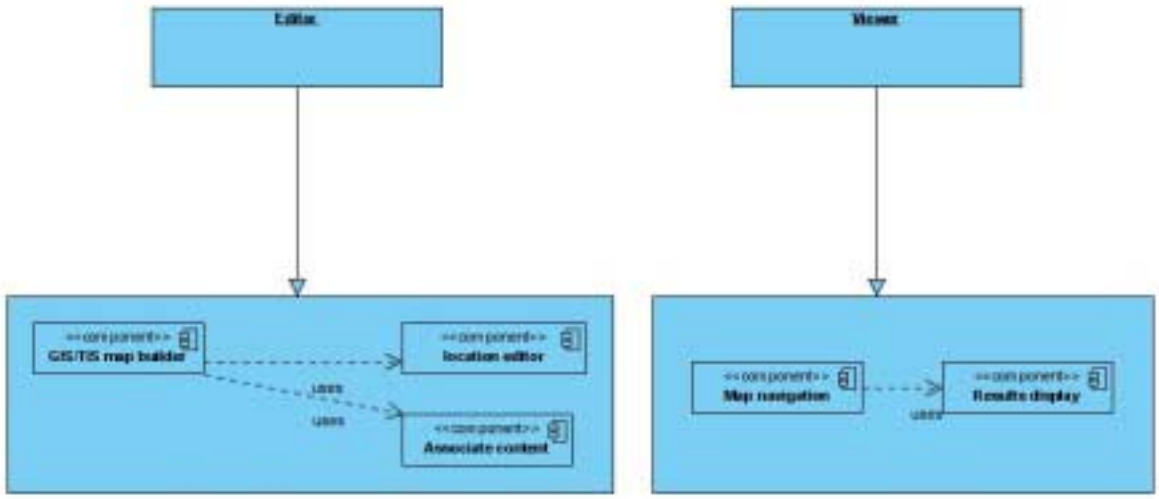


Figure 1

Actually, we use a specific database but for Mosaica, the data (load, save…) will be accessed through the semantic layer by implementing a common interface on each side.

3.2.2.3 The GIS/TIS



There are two different main modules in the GIS/TIS application, the Editor which contains all the method needed to create, locate and assign objects to place, and the Viewer which allows the user to navigate through the maps and objects create by the previous one.

3.2.2.3.1 The Editor

The viewer supports:

- **GIS/TIS Map builder:** This component manage the maps and time and allow to create new map or to modify existing maps.
- **Location editor:** The location editor is for the creation of different locations on map. There are different types of location: Capital, city, town, site, zone, trajectory… and other.
- **Associate content:** This component provides the needed methods to associate objects to the previous elements create with the location editor. Different objects can be associate or even other maps.

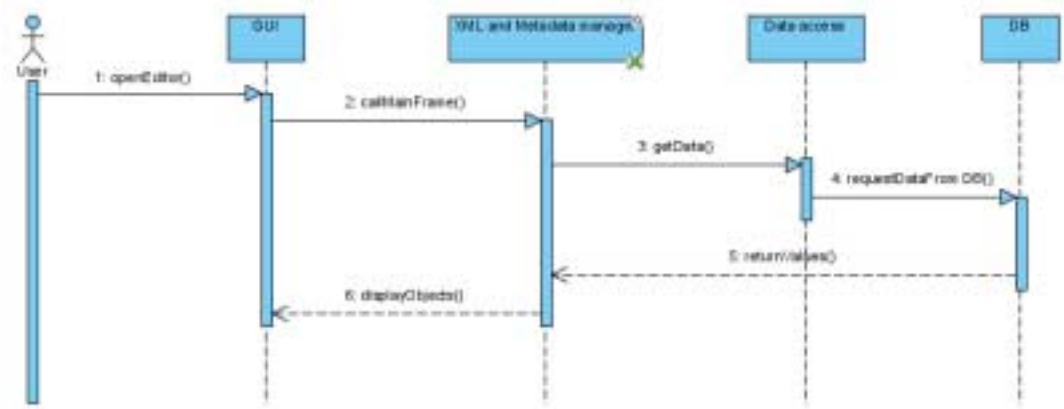
3.2.2.3.2 The Viewer

The viewer supports:

- **Map navigation:** This component allows the user to navigate through maps and to find the content he looking for directly on the map.
- **Results display:** This component allows displaying the results of a specific search made by the user in the web site. For example, for a search on Jerusalem in 1320, the results will be display on a map of the city.

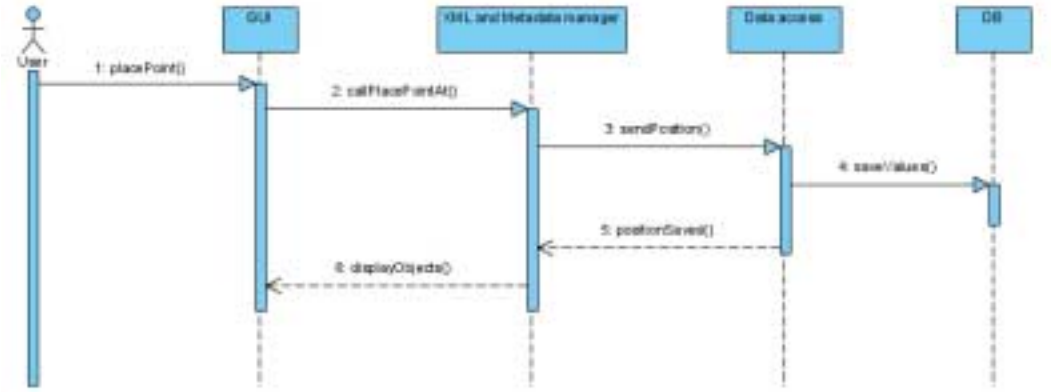
We present hereafter what can be done with the editor through a series of collaboration/sequence diagrams.

Opening editor



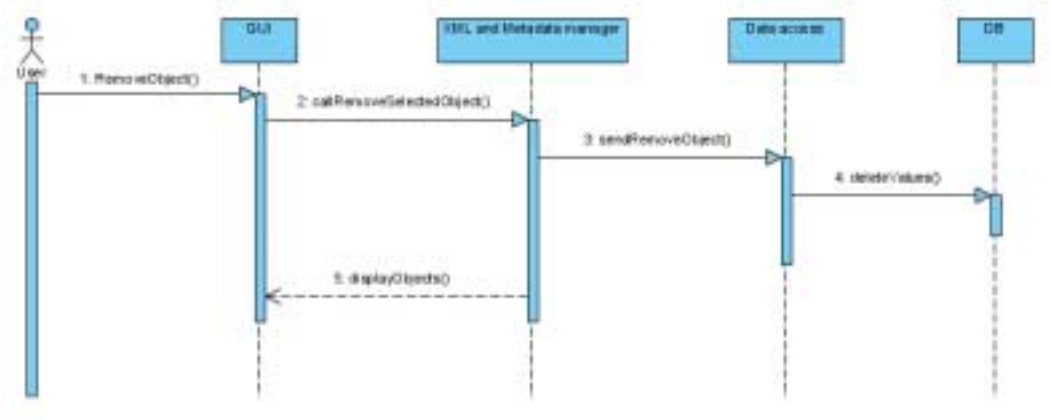
This diagram represents the opening of the editor and how the data from the DB are retrieved by the application

Place an object on a map (e.g.: a city)



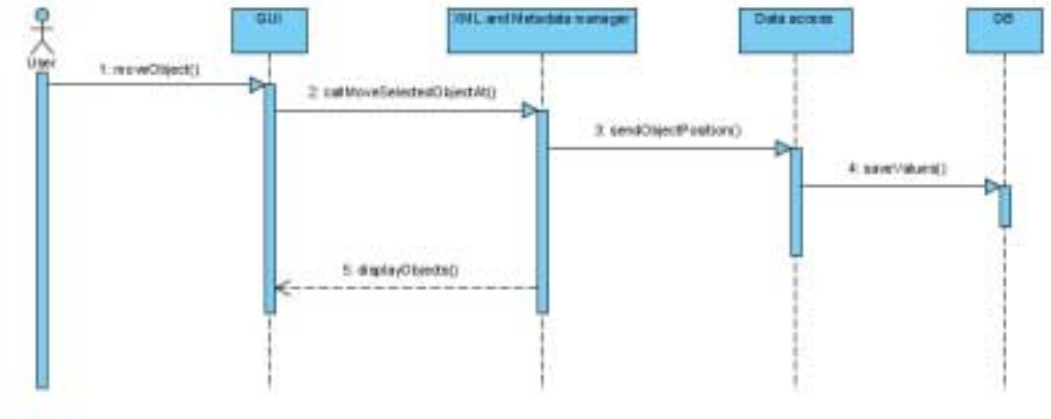
This diagram represents the placement of an object in the editor, how the data are saved in DB and how the application gets the list of objects to display from the DB.

Remove an object



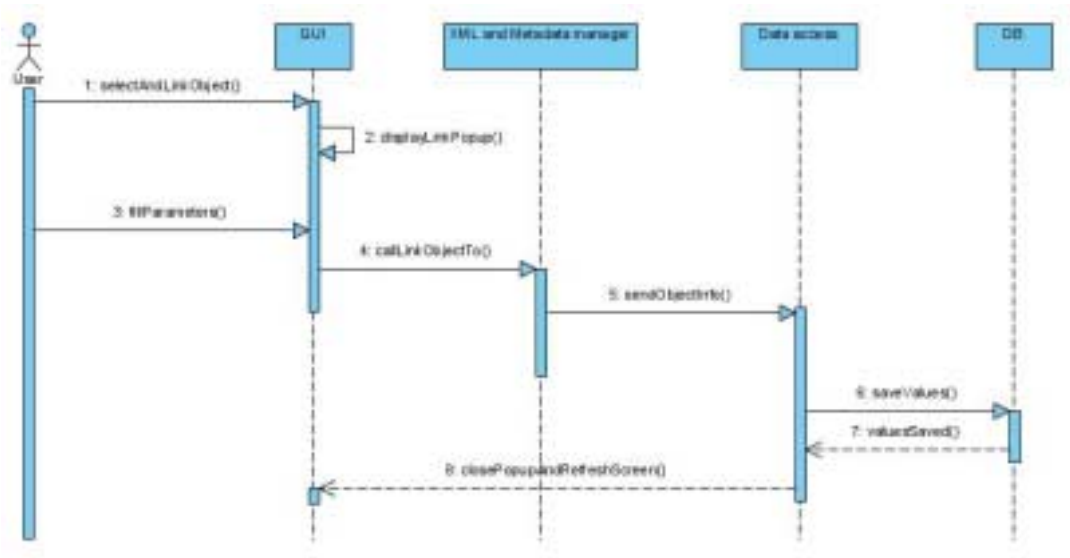
This diagram represents how to remove an object in the editor.

Move an object



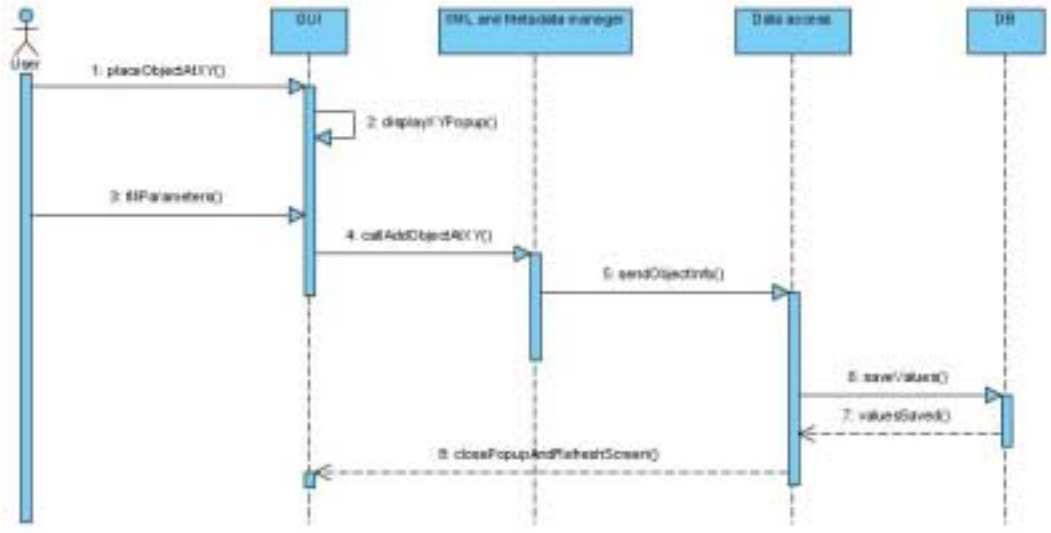
This diagram represents the repositioning of an object in the editor and how the new position is saved in DB.

Assign a sub map (or another object) to an object



This diagram represents the attachment of a sub map or object to an existing place (city, capital, site…) on the map. A sub map can be attached for a specific period or not.

Place an object at a specified coordinates (x, y)



This diagram represents the placement of an object on a specific coordinates in the editor and how the data are saved in the DB.

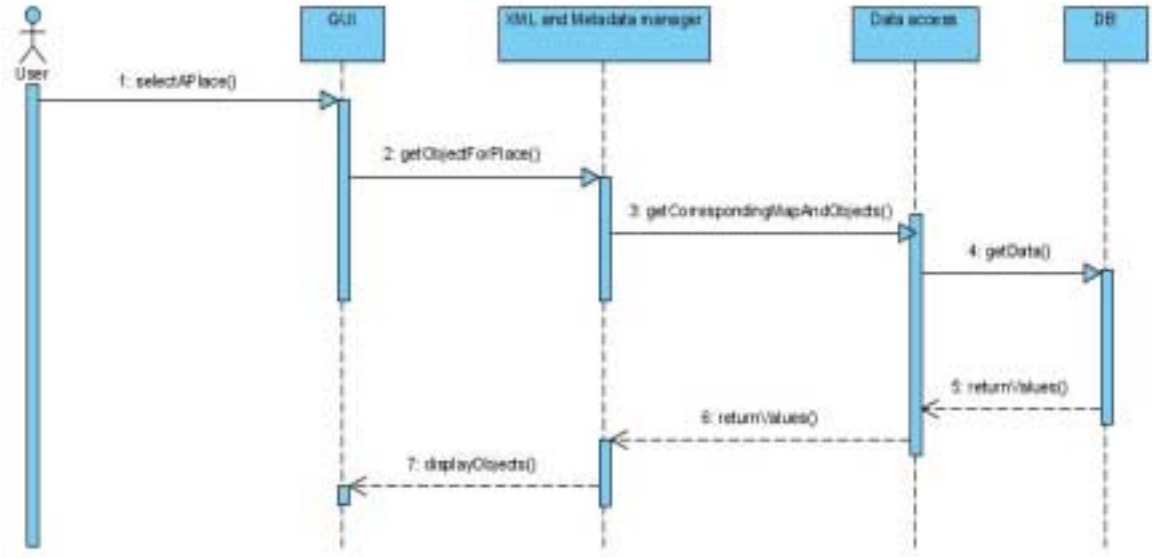
3.2.2.3.3 The Navigation

There are two ways of using the navigation:

- A user can start to navigate directly by map and after selected a place, he'll get the object assigned to this place (the navigation will request this data to the semantic layer)
- Or he can use a search engine to find the content he looking for and the results will be displayed on a map. For example, the user chooses Paris in 1830 and a map of Paris with the corresponding objects will appeared.

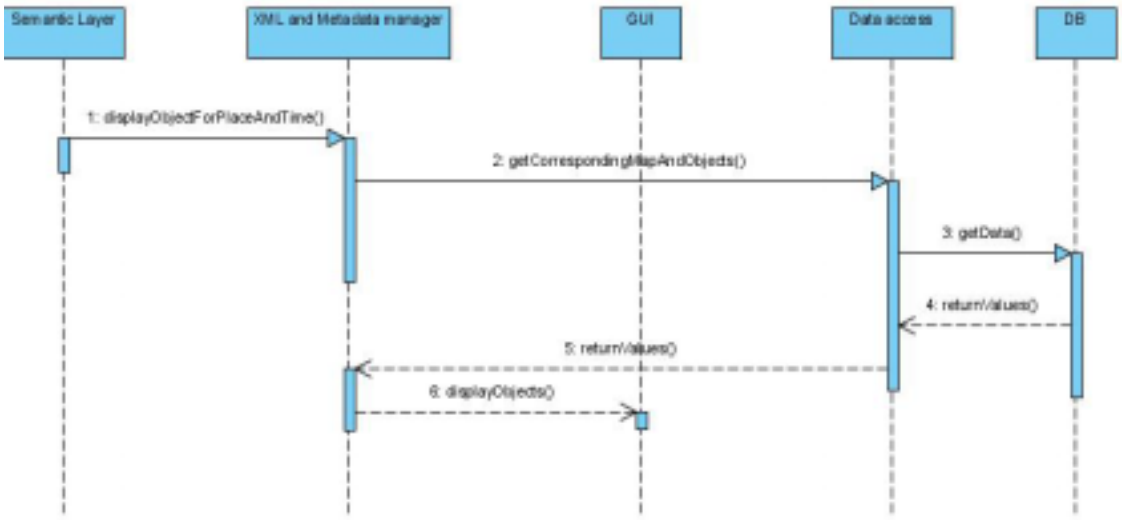
We present hereafter what can be done with the viewer through a series of collaboration/sequence diagrams.

Navigation by map / time



The user select a place and the application will return the objects and map linked to this place. For the moment we use our own database but the data will be requested through the semantic layer in Mosaica.

Display results of a search



The semantic layer sends a request to the GIS/TIS to display the map and objects corresponding to the user search. The GUI will display a map with objects.

Presenting content

The GIS/TIS will also allow the display of different types of content like images (jpg, gif…), animation (flash…), website or others to be defined in the future.

3.2.3 Virtual Expeditions

Due to the variety of different opinions regarding implementation of the Virtual expeditions, we briefly present the alternatives, with one of the most advanced fully explained.

The alternatives for implementation of the Virtual Expeditions are:

- HTML based VEs
  - Frame based
  - AJAX/AHAH based
- Flash based VEs
  - Wizard based
  - Standalone Flash tools based
- 3D environment based (in-depth explanation given below).

Brief description of the original HTML-frame-based VEs was already given in the DOW. We think that the HTML-frame solution is a bit outdated, but can be used if other alternatives fail. Solution with the same HTML building blocks as the MOSAICA, but with more advanced methods of gluing them together is the AJAX/AHAH based one. Namely, using such solution, we can exploit new server architectures, giving the VEs better look and feel, mainly through using asynchronous methods and callbacks. Such a solution would also move the web application into the desktop direction. The Flash based solutions incorporate, as the end result, Flash VE. The difference between them lies in the composition of the Flash interactive movie. Namely, we could incorporate a freeware Flash compiler into the MOSAICA as a service and basically use a wizard for composition of the Flash script. This script would then be automatically compiled and the end-user would receive only the results in Flash. Supporting the diversity of user actions and scenarios, such a solution would be hard, but not impossible to implement. On the other hand, there already exists a plethora of different Flash tools, which could be used by the end users in order to compose a Flash VE. Finally, an advanced solution by using a 3D web-based interactive engine from Caspian Learning was also discussed – it is thoroughly presented below.

3.2.3.1 High level architecture:

The high level architecture is shown below. The Caspian Learning Engine component stands for the whole Caspian Learning software – the graphical engine, the task editor, etc. Mosaica VE Connector is the basic component for connecting the Mosaica web services (mostly search engine). This component is responsible for a) capturing the output of the Caspian Learning Engine, b) getting the required data from the Mosaica Search Engine, c) providing input to the Caspian Learning Engine and d) cashing the VE (or its state).

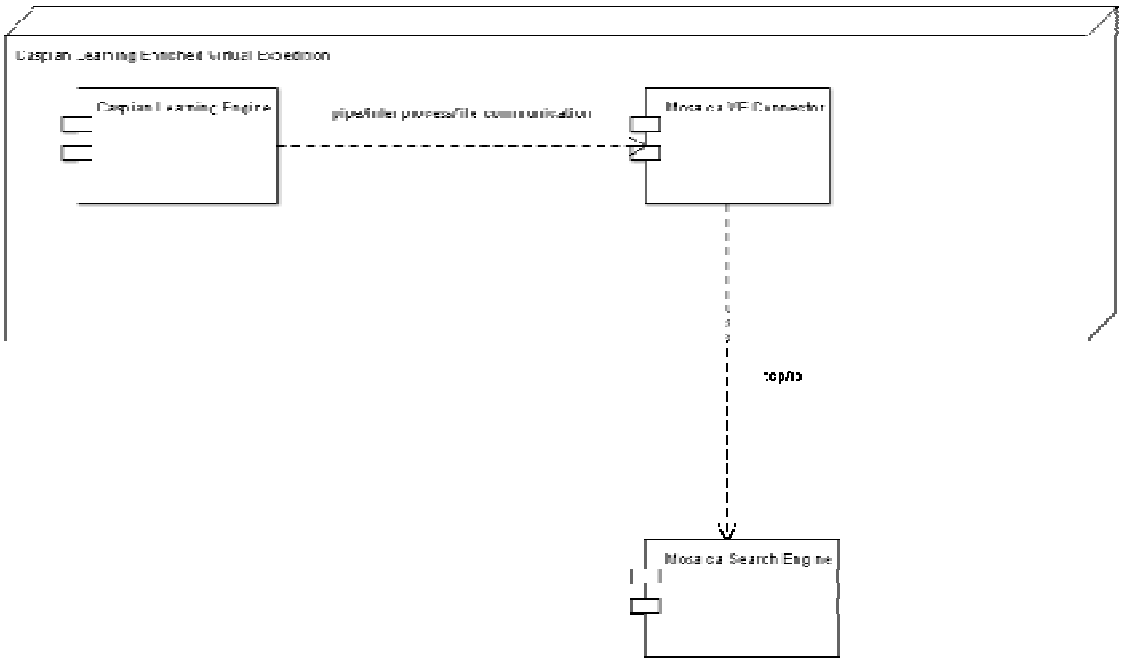


Figure 8: High level architecture of the Mosaica VE aided by Caspian Learning Engine

3.2.3.2 Mosaica VE Connector

The breakdown of the Mosaica VE Connector into classes is shown. Basic tasks of the Mosaica VE Connector are:

- to open and close connection towards the Mosaica servers,
- to store and retrieve the VE game data (questions, answers)
- to retrieve the VE data
- to download the VE data on demand and cache it.

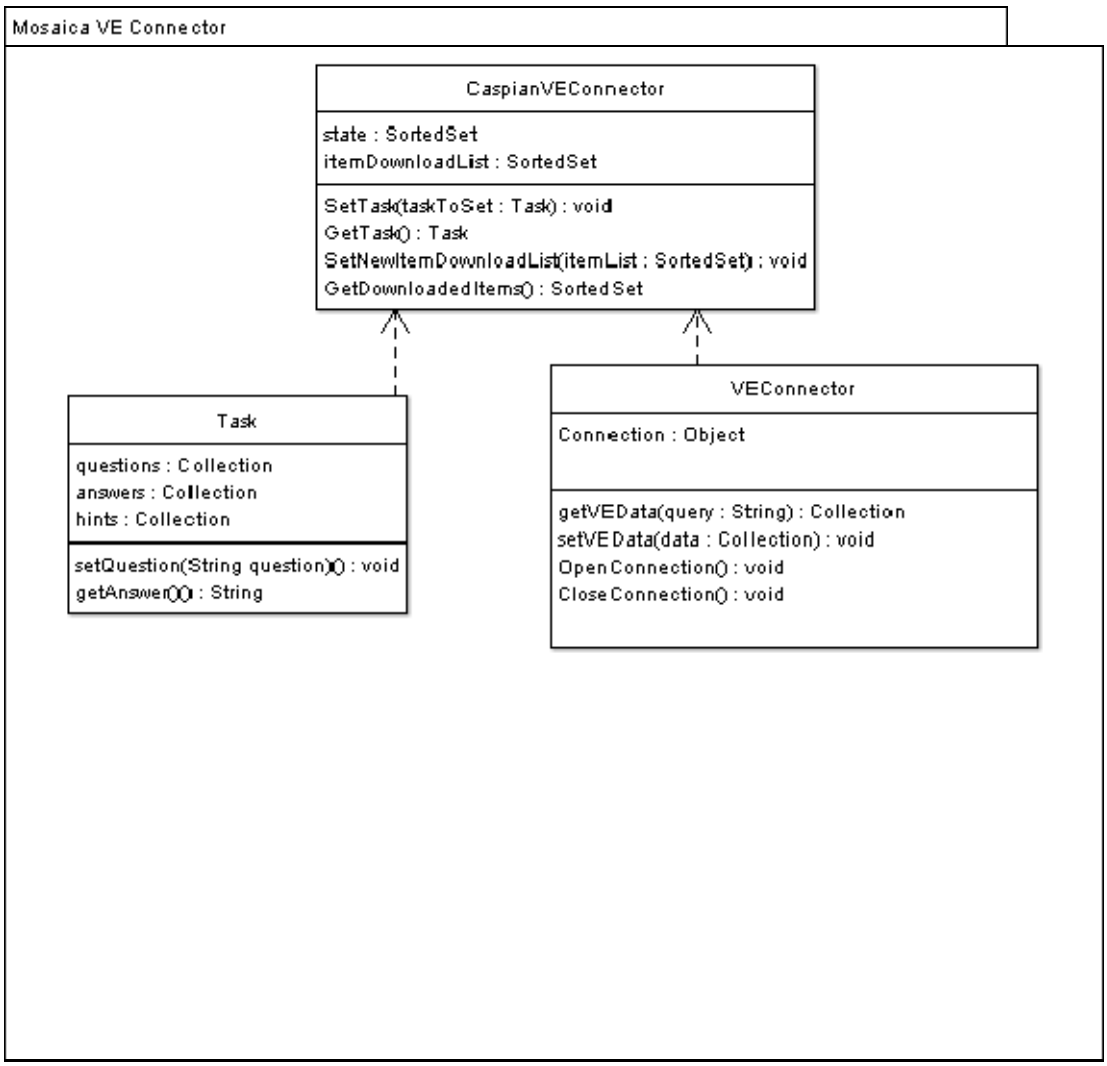


Figure 9: Class diagram of the Mosaica VE Connector

3.2.3.3 VE Activity diagram

The Virtual Expedition will start by clicking on the link for the Virtual Expedition. This will start a component in the browser that will locally run the VE connector and then redirect to the ActiveX component, provided by the Caspian learning. The skeleton VE will start in the browser, while in the background, the data for the VE will be downloaded (according to the prioritized download list) and cached locally.

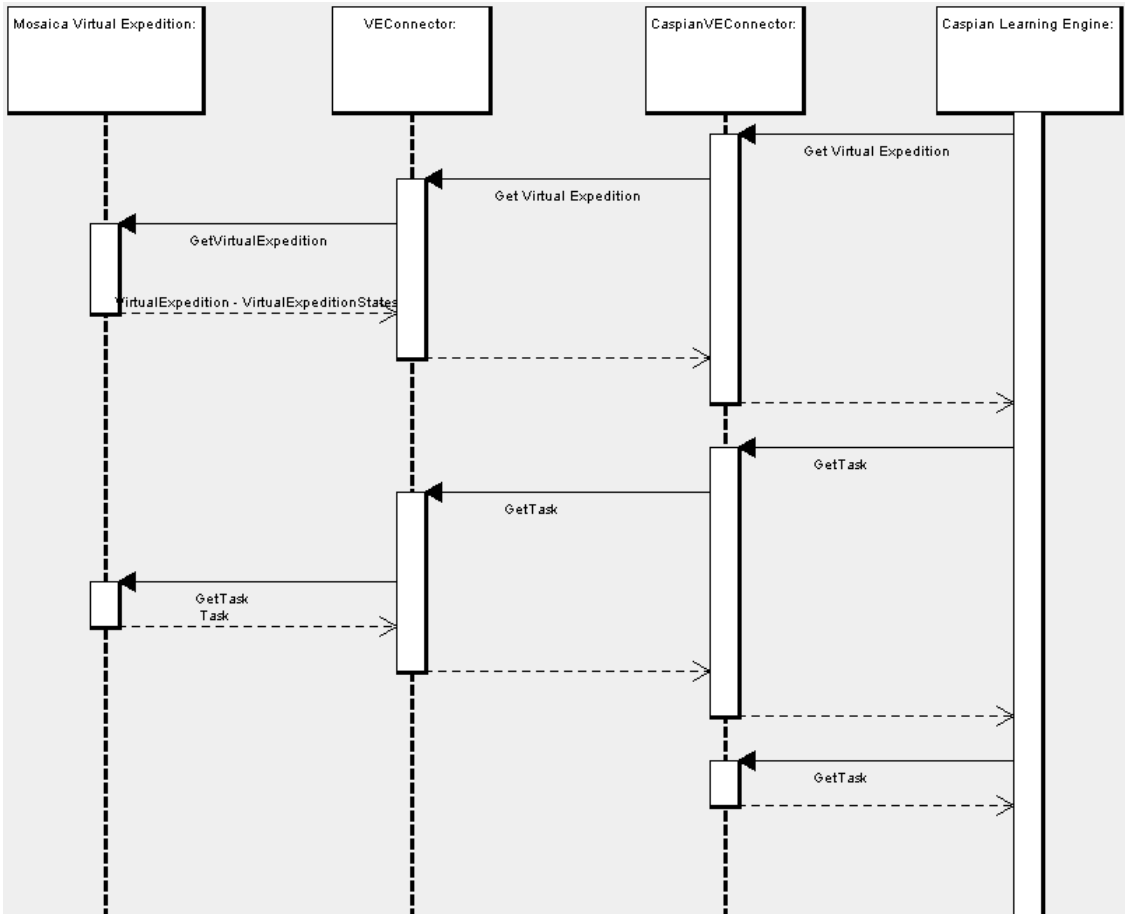


Figure 10: Mosaica typical sequence diagram of the VE as seen from the VE Connector

3.2.4 p2pCMS

3.2.4.1 High level architecture

3.2.4.1.1 Functional Architecture

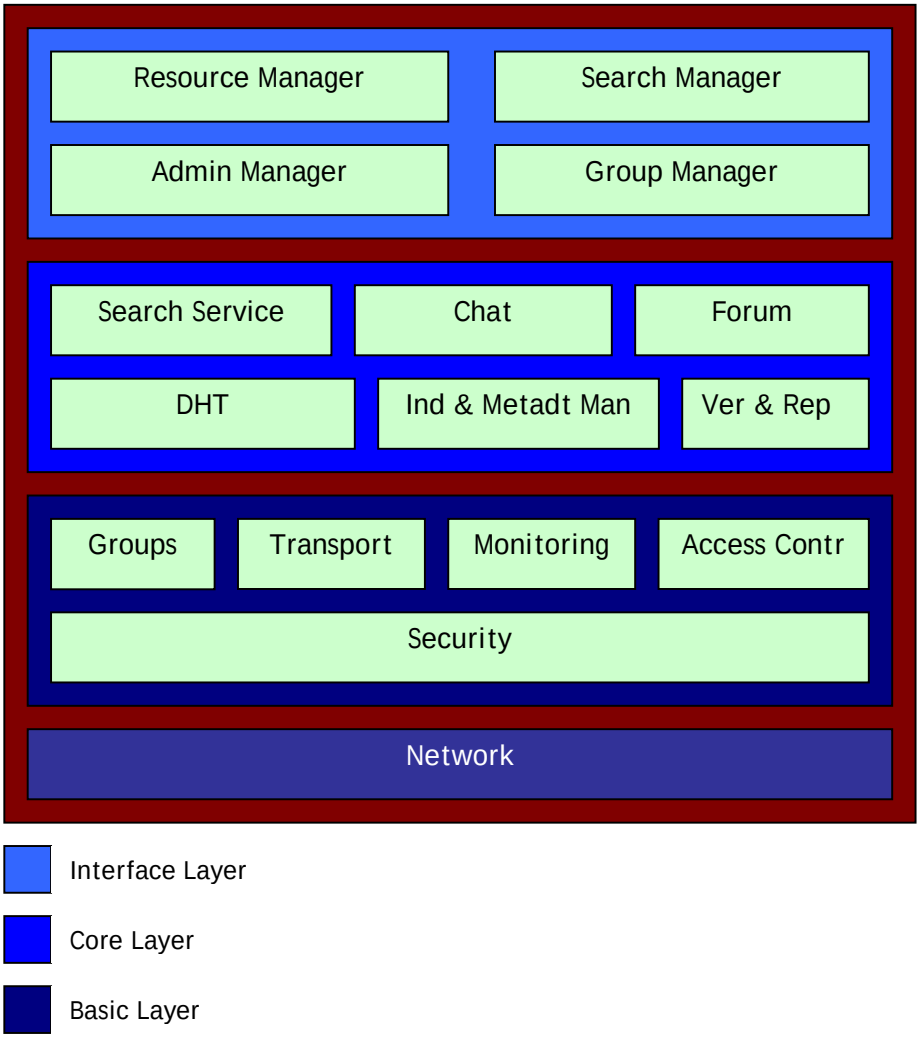


Figure 11 – high-level layered architecture of the p2p-cms

The p2p-cms presents a layered and service-oriented architecture. This approach is very attractive as it allows isolation of the application layer functionality from implementation details. It therefore enables to replace at a later stage, components that implement specific functionality without having to alter the interface layer and therefore without interfering in the implementation of other MOSAICA subsystems. It also allows easy extension of the functionality provided by the p2p-cms by incorporating new components that implement new services, thus offering extended functionality to the upper layer. In addition, by further dividing the back-end functionality of the p2p-cms in two layers has the advantage of also isolating the

MOSAICA-specific functionality from the generic functionalities needed to run a distributed content management system. As such it is possible to re-use existing open-source software for these more generic functions, while allowing to invest more efforts in the MOSAICA-specific functions.

The basic layer provides elements that are absolutely necessary within DCMS implementation and that generally can e found in any p2p architecture for he controlled access to multimedia content. It thus implements functionality that is not MOSAICA-specific. In this layer the main P2P principal protocols are defined and implemented..

The core layer contains services that use the functionality provided by the basic layer and that implement specific functionality to satisfy MOSAICA requirements..

The third and upper layer is provided for interaction with external actors.

3.2.4.1.1.1 Basic Layer

In this layer, generic p2p functionality is implemented. Part of this functionality is even essential for running a p2p system, as is the case of the 'Transport' services. In many cases it will be possible to re-use existing modules available in the open source repository of the project JXTA.

- **Security**: the security service permits secure communication and message authentication. It is crucial in services that need access control.
- **Groups**: the group service provides with an abstraction in the set of all the peers. Peers use the group service in order to organize them in groups of interest.
- **Transport**: it implements the message transport between peers. The principal network protocol that will be used is the TCP/IP but it is possible to use other network transport protocols such as HTTP or UDP.
- **Monitoring**: Monitoring service is used to monitor valid content.
- **Access Control**: permits identify, authenticate users. It also provides system with user permissions to use some services. This service relies on the security service.

3.2.4.1.1.2 Core Layer

This layer provides the functionality specific to the MOSAICA requirements that is not directly accessed by the user but that enables the p2p-cms to deliver the required value to the external actors (users or other MOSAICA subsystems).

- **DHT**[1]: this service allows the MOSAICA system to maintain a distributed list of indices to the content available throughout the Web, implemented as a distributed hash table. This approach provides the mean to quickly send query messages to appropriate peers and accordingly locate an access the desired content..

[1] As referred in section 2.1, the use of a DHT based service to implement the search functionality is still to be decided. Nevertheless it is foreseen the use of DHT to distribute the MOSAICA resources and retrieve already searched and selected digital objects This will nevertheless be described in deliverables D3.1 and D3.2 of WP3.

- **Indexing & Metadata Management**: this service provides functionality to manage indices and content metadata. In an initial version, this service will provide an interface that ignores ontology. Other types of implementation can be developed thereafter.
- **Versioning & Replication**: it enables distribution, versioning and replication of content. Replication is a basic mean to provide fault-tolerance in presence of high churn. Both Indexing and Replication services will run over a structure service that will implement a DHT[2]. That service can also be implemented through a central server.
- **Search Service**: the search provides the system with search capabilities. It will take advantage of the Indexing Service to quickly find the content. When the former method fails, the search service should have the capability of broadcast queries to logical neighbours of presumably content owners.
- **Forum and Chat Services**: those services provide the user with static and dynamic communication channels inside a group.

3.2.4.1.1.3 Interface Layer

The interface layer will provide the external actors with the means to interact with the p2p-cms, thus allowing them to use the vale delivered by this MOSAICA subsystem.. External actors will interact with DCMS system using the classes of the interface layer. Communication with external subsystems will be made via a Web Services interfaces using SOAP to exchange messages, as agreed within WP5.

3.2.4.1.2 Package Structure and Interaction

The diagram in figure 2 represents the services' structure of the p2p-cms, illustrating the interaction that will be established between the services to implement the identified functionality. All the represented services will be implemented in the core and basic layers of the p2p-cms.

[2] In this case, the use of DHT refers to the implementation of the distribution of content. Although there are still some doubts concerning the use of DHT to implement the search functionality (see section 2.1.7.2), it is foreseen that the MOSAICA p2p-cms will indeed make use of DHT to distribute and store the MOSAICA digital objects across MOSAICA peers and then to retrieve already identified content. This will nevertheless be described in deliverables D3.1 and D3.2 of WP3.

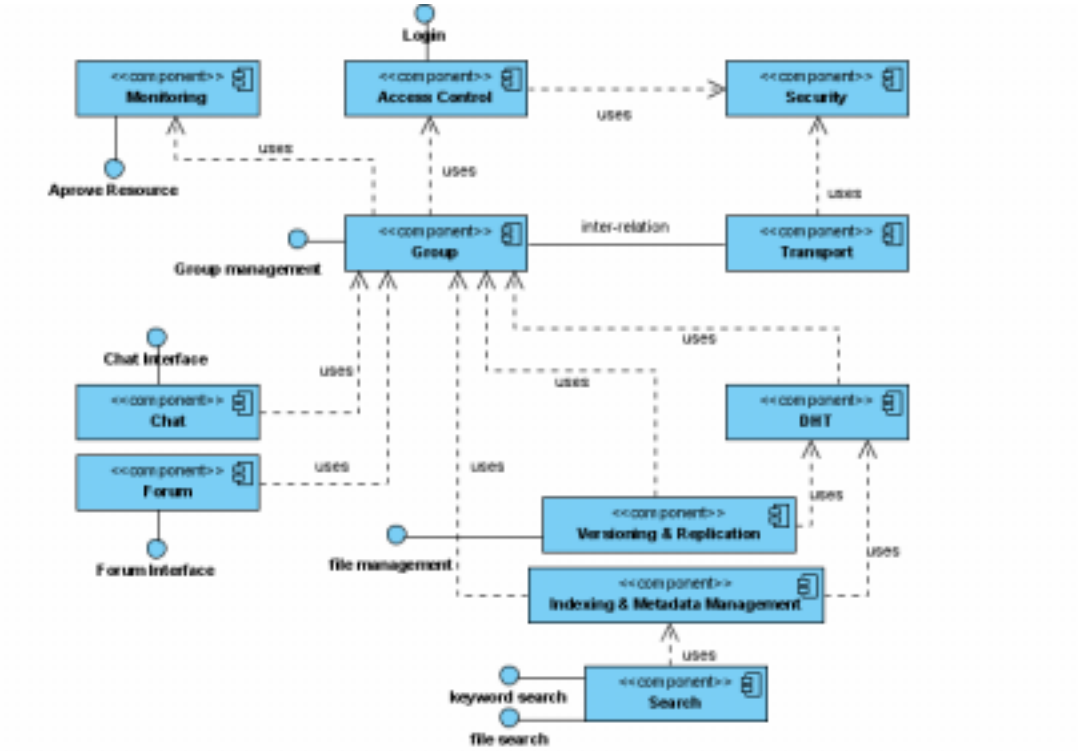


Figure 12 – the p2p-cms services' structure

Some basic services are provided by specific groups although every group implements some crucial services like *Monitoring* or *Access Control*. There will be a default group where every user has access to it. Groups are created by administrators.

The *Group Service* can run over several machines and needs some sort of transport service. On the other side, transport messages belong to some group.

Forum and *Chat rooms* are created by group administrators inside a specific group.

The *Access Control Service* is used to allow a user to log in the system. When some user requires some administrator specific operation, *Access Control* must be consulted in order to allow or deny user request.

Versioning and Replication Service will be used to store content. This service will use some sort of DHT[3] functionality. The P2P system could provide a simple keyword search mechanism based on file name and author.

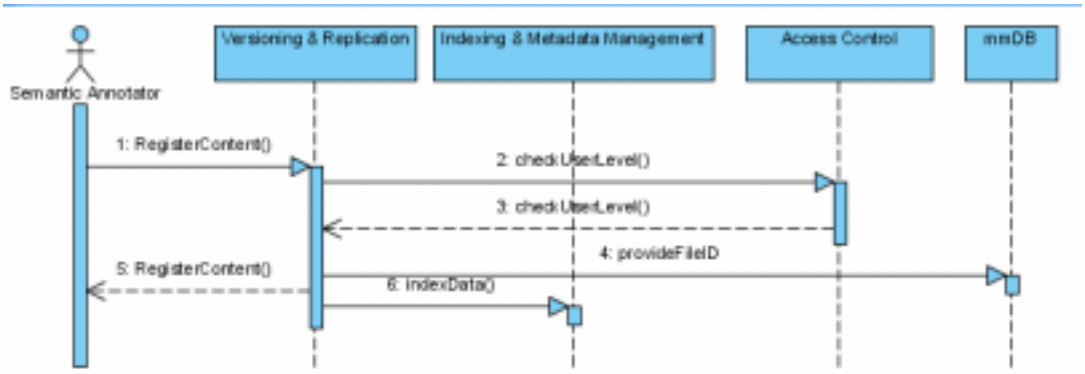
Administrators use the *Monitoring Service* to allow or deny network resources.

The *Security Service* is used to encrypt or sign messages between peers. Probably it will run locally as a module in each peer where MOSAICA's application is installed.

[3] In this case, the use of DHT refers to the implementation of the distribution of content. Although there are still some doubts concerning the use of DHT to implement the search functionality (see section 2.1.7.2), it is foreseen that the MOSAICA p2p-cms will indeed make use of DHT to distribute and store the MOSAICA digital objects across MOSAICA peers and then to retrieve already searched and identified content. This will nevertheless be described in deliverables D3.1 and D3.2 of WP3.

3.2.4.1.3 Collaboration/Sequence diagrams

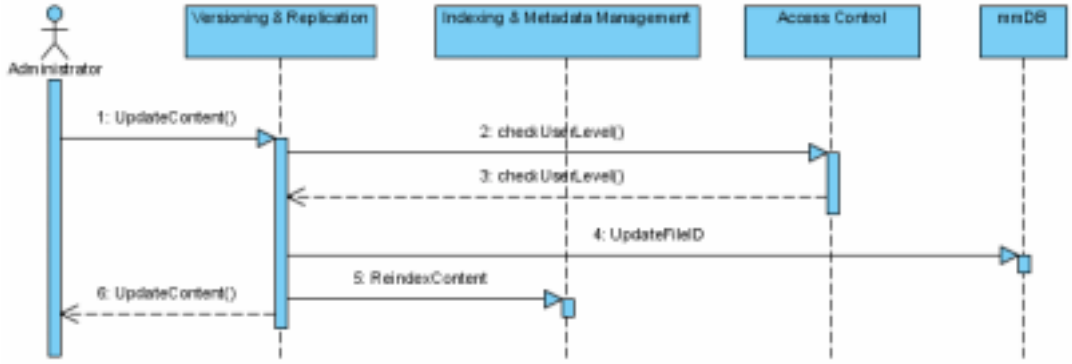
Content Registration



This diagram represents content registration. Every user has permissions to register his content in the groups that he has access. There is no need to consult Access Control Service to determine if user has permissions to register content in the default group.

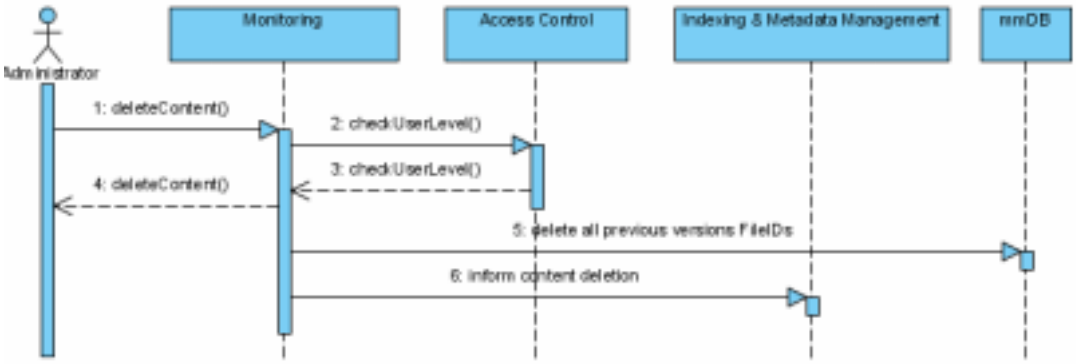
Content is constituted by the file and the correspondingly metadata information.

Content Update



Content update requires administrator level access. When content is updated by the administrator, is created a recent version of the content. The implementation process hasn't been studied yet.

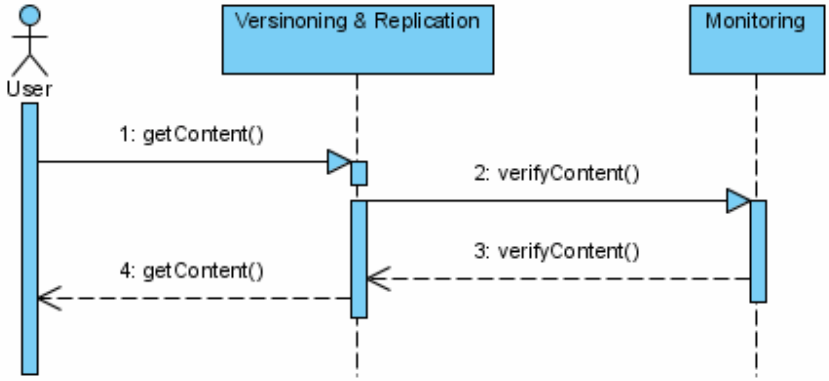
Content Deletion



The administrator can access a content pool where he can allow or deny some resources registered in the system. This feature is provided by the Monitoring Service. Once the content isn't own by the system, content isn't really deleted but removed instead.

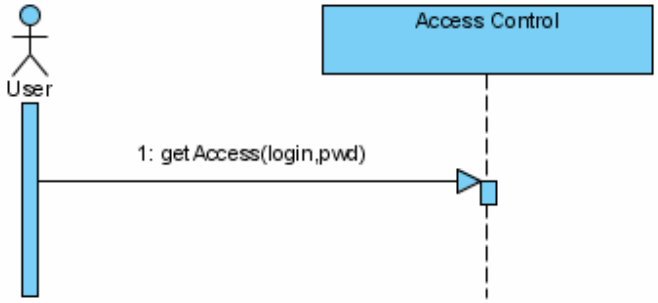
Other deletion method is provided by the Versioning and Replication Service which is used to content removal by the owner. This method is also called by the Monitoring Service when content is denied by the administrator.

Content Request



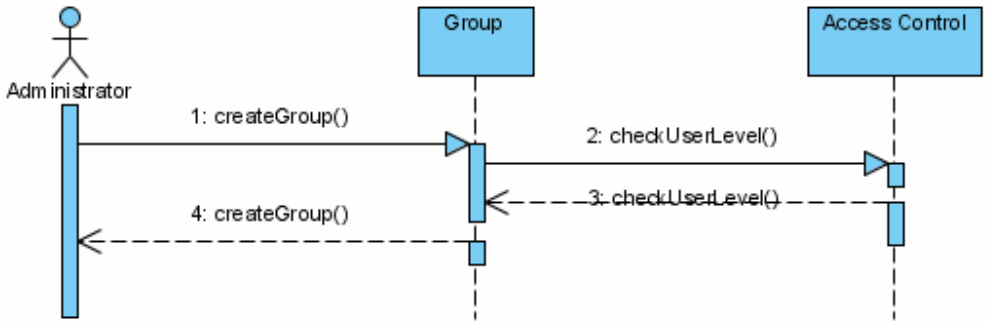
When content is requested, it is Monitoring Service's responsibility to determine if the content is valid in the system.

Login



In order to use the system, the user must boot in the system by the Access Control Service. The Access Control Service implements Identification, Authentication and Authorization processes. If a user doesn't login into the system, he will be considered as an anonymous user.

Manage Groups



Groups are created by administrators. Access Control Service is consulted to determine if the user has the permissions to create a group. Group deletion is a hard issue because several peers may be connected to a group during group deletion. Administrators can create groups with a time to live providing a mean to connected peers agree with group deletion.

The same schema is valid for users to join a group. Before user can join a specific group, the Access Control Service must be consulted to determine if user has group joining permission.

Administrators also have permissions to create chat and forum rooms inside a group. Users have the ability to use chat and forum rooms to communicate with each other.

3.2.4.2 Services Structure

This section provides an overview of the preliminary interface architecture. It should be noted that some methods (like those used to manage content) must be implemented as threads due to their asynchronous nature.

The login process will be attained by consulting a central authority server (or some small set of servers). This can be implemented as a Web Service. In fact, the bootstrap process can be accomplished using Web Services. On the other side, the rest of services will be message oriented and will use JXTA like protocols (this issue is open for discussion).

In the former schematic diagram, Advertisements are XML structured documents that contain all information about some network resource.

3.3 Interfaces (system requirements)

3.3.1 Application Front-end

3.3.1.1 Display User Interfaces

1) All components should use the same UI standards in order to give the end user a consistent/friendly environment on one hand and to ease software maintenance on the other.
2) Front-end application must render ontological assertions (i.e. axioms consisting of a concept as its subject, a relation as its predicate, and another concept or a value as its object) for display.
3) Front-end application must display the result of the intelligent presentation.
4) Front-end application must enable use of different languages.
5) End user must be able to upload additional content.
6) End user must be able to develop Virtual Expeditions.

3.3.1.1.1 Semantic Search:

7) Front-end application must render search results for display.
8) Front-end application must render ontology concepts organized in a hierarchical structure for display.

3.3.1.1.2 Semantic Browsing:

9) Semantic layer must provide on demand entire ontology, or selected subsections of it.
10) Semantic layer must provide on demand ontological assertions associated with a particular ontological concept.

3.3.1.1.3 GIS/TIS:

11) Front-end application must display an era-depended map
12) Front-end application must display an object on a map (link created by GIS editor)
13) Front-end application must represent objects as icons
14) Front-end application must manage navigation between maps
15) Front-end application must display all objects assigned to a specific map add an object to a map (at a specific coordinate)

3.3.1.1.4 Virtual Expedition player

16) VE must provide a VE-Selection user interface
17) VE must provide a Guide selection user interface
18) Virtual Expedition player must play the stored virtual expedition.
19) Virtual Expedition player must enable the user to stop the virtual expedition and continue on a different search.

3.3.1.2 Editing User Interfaces

3.3.1.2.1 Semantic Annotator

20) Front-end application must enable upload of new resources.

21) Front-end application must enable selection of the resources existing in the system for semantic annotation.

22) It is recommended that the front-end application provides facilities for validation of the user data input, including (1) data type validation (e.g., numbers, dates, etc.), and (2) spell checking.

3.3.1.2.2 GIS/TIS Editor

23) A GIS editing tool **must** enable the user to:

a. assign places, points on a map (geo-referenced or not)

b. give a period to a map (from date to date)

c. create a zone on a map

d. link an object with a representation on a map

3.3.1.2.3 Virtual Expedition Authoring Tool

24) The system should be able to edit and store VEs

25) The system must provide the Mosaica API to the VE authoring tool.

26) The VE authoring tool must output the data in the proper Mosaica format.

27) Virtual Expeditions must enable its organization into multiple successive stages.

28) Every stage of a Virtual Expedition must enable display of multiple content items (Cultural Heritage Items).

29) End users must be allowed either:

- To select any desired stage in the Virtual Expedition (semi-guided use), or
- To move to the next stage only after completing the previous (guided use).

30) Transition from one stage to another should include option to condition it with the completion of either:

a. A task (for example, successfully complete a riddle or a game), or

b. Viewing of all available content in the previous stage.

31) Transition from one stage to another must support the following features:

a. Forking enabling to define succession to multiple stages from a single stage, and

b. Convergence enabling to define succession to a single stage from multiple stages.

32) At each step the user must be presented with the option to terminate Virtual Expedition.

33) In case that Virtual expedition was terminated before its completion, the end user should have the option to resume the same Virtual Expedition in the future from the last visited stage.

34) Virtual Expedition must support the following content types:

    a. Text,

    b. Sound,

    c. Graphic images,

    d. Video streams, and

    e. Animation.

35) The content displayed at each stage must be one of the following:

    a. Internet resources (for example, sound file or graphic file located on a Web server),

    b. System resources (for example, files uploaded by users and stored within the system), or

    c. Preset text (entered through an authoring tool during editing).

36) Virtual Expeditions must enable end users to provide text input (by typing) while undertaking Virtual Expedition.

37) Virtual Expedition must be interactively linked to the Semantic Search, Semantic Browsing and GIS/TIS components, in order to provide the following features:

    a. Querying semantic index and display of search results,

    b. Querying of ontology and display of ontology elements,

    c. Display of GIS/TIS maps for interactive navigation.

38) Virtual Expeditions must provide instructions at every stage, guiding end user how to proceed.

39) System must provide an easy-to-use editor (authoring utility) for creation and modification of Virtual Expeditions.

3.3.2 Application Backend

40) System components should be able to interface via web-services using XML.

41) The application backend must interface with the Semantic Layer via XML based web-services

Semantic Layer:

42) Backend application must transfer keyword query in XML format.

43) Backend application must provide parameterized instructions regarding which indices are targeted by a keyword query.

44) Backend application must parse data input for advanced keyword querying, and transfer it in XML format.

45) Backend application must transfer requests regarding which ontology concepts will be displayed in a hierarchical structure.

46) Backend application must transfer requests to display associated information with each selected ontology concept.

47) Backend application must parse retrieved search results.

48) Backend application must transfer requests to retrieve a specific selected resource.

49) It is recommended that the backend application be able to transfer requests to retrieve cached version of a specific selected resource.

50) Backend application must parse semantic annotations, and transfer them in XML format.

P2P Content Management System (p2pCMS):

51) p2pCMS must be able to receive SOAP messages from the semantic annotator, requesting the registration of new content.

52) p2pCMS must be able to receive SOAP messages from the semantic annotator, requesting the deletion of existing registered content.

53) p2pCMS must be able to receive SOAP messages from the semantic browser, requesting content.

54) p2pCMS must be able to receive SOAP messages from the mmDB, requesting unique content identifier.

55) p2pCMS must provide with "system administration" utilities in order to manage the system.

GIS/TIS:

56) Backend application must provide an interface to other components based on web-services through XML

57) Backend application must be able to communicate with a database in order to store and retrieve GIS information

3.3.3 Semantic Layer

3.3.3.1 Semantic Search

58) Semantic layer must provide search results in response to a query.

59) Semantic layer must provide indexing facilities.

60) Semantic layer must provide querying facilities. Querying can target various indices, including (1) ontology concepts, (2) metadata, (3) full-text documents (either for textual resources or for free-text annotations), etc.

61) Semantic layer must provide facilities for implementing Boolean operators in querying.

62) Semantic layer must provide facilities for querying precise keyword sequences.

63) Semantic layer must provide neighboring objects of it's current search result in a form of the RDF graph.

64) Semantic Search Layer must provide an interface to the VE component.

65) Indexing facilities should be capable of indexing any requested item (e.g., tokens from full-text, or ontology element labels and comments, metadata fields, etc.).

66) Semantic layer must provide facilities for relevance ranking of search results.

67) Semantic layer must provide facilities for sorting and ordering of search results by additional criteria (e.g., metadata).

3.3.3.2 Semantic Browsing

68) Semantic layer must provide semantic (concept) indexing, which matches ontological concepts to textual resources.

69) Semantic layer must provide metadata indexing using applicable metadata schemas (which may be different for different resource types).

70) Semantic layer must provide facilities for tokenization and morpho-syntactic normalization of submitted queries.

71) In case of textual items (e.g., ontology concept labels and comments, free-text annotations, textual documents, etc.) semantic layer must provide tokenization and morpho-syntactic normalization facilities, in order to index each token separately.

3.3.3.3 Semantic Annotation

72) Semantic layer must be language independent, and provide all functionalities in any available language.

73) Semantic layer must provide facilities for indexing semantic annotations (i.e. annotations consisting of an ontological concept, an ontological relation specifying the association of that concept to a resource, and the resource location (URI)).

74) It is recommended that the semantic layer provides facilities for keyword matching by Soundex, or similar phonetic matching algorithm.

3.3.4 GIS/TIS

1) GIS must provide the correct map based on the input coordinates.

2) TIS must provide the correct map based on the input coordinates and time.

3.3.5 Distributed Content (P2P layer)

Semantic Layer:

1) P2P infrastructure must provide hashing facilities for various inputs.

2) P2P infrastructure must provide distribution of hashed indices by their hashkey.

3) P2P infrastructure must provide retrieval facilities based on hashkeys.

4) P2P infrastructure must provide set-intersection facilities for filtering search results.
5) P2P infrastructure must accept structured index input (i.e. “put” request for distribution of indices).
6) P2P infrastructure must accept structured query inputs (i.e. “get” request for retrieval of requested indices).

4 Conclusions

The document presents the user and system requirements of the MOSAICA application, along with the high level architectural design.

The components are described using the UML notation, along with their placement in the overall system architecture.

5 References

[1] Yingwu Zhu and Yiming Hu., "Semantic Search in Peer-to-Peer Systems". [Book Chapter]. In Handbook of Theoretical and Algorithmic Aspects of Ad Hoc, Sensor, and Peer-to-Peer Networks . J. Wu (ed.), Auerbach Publications, 2006, pp. 643-664.

[2] Yingwu Zhu, Honghao Wang, and Yiming Hu. "Integrating semantics-based access mechanisms with P2P file systems", in Proceedings of the 3rd International Conference on Peer-to-Peer Computing, pages 118–125, Linkping, Sweden, September 2003.