Service contract B4-3301/2001/329175/MAR/B3
"Coastal erosion – Evaluation of the need for action"
Directorate General Environment
European Commission

Living with coastal erosion in Europe: Sand and Space for Sustainability

[Source Document]

Manual of procedures for setting up Local Information Systems

VOLUME II : TECHNICAL SPECIFICATIONS

Final version – May 10 2004

National Institute for Coastal and Marine Management of the Netherlands (RIKZ)
EUCC – The Coastal Union
IGN France International
Autonomous University of Barcelona (UAB)
French Geological Survey (BRGM)
French Institute of Environment (IFEN)
EADS Systems & Defence Electronics

Volume II

Technical specifications for prototyping an Internet-based application

INTRODUCTION

Objectives of the technical specifications for prototyping the Internet-based application

This volume of the guidelines describes the technical specifications of the future Internet-based prototype application (so-called "LIS prototype"). This prototype will constitute the model application from which real LIS will be adapted and, as such, will contain the basic features expected to be common to all of these LIS. As local adaptations and further extensions of the information system at local sides is needed, this volume provides a step-wise methodology to set up a local information system. Although this volume can be considered as terms of reference that an IT developer will be assigned to implement, it is important to be read and understood by users and local stakeholders as well. These guidelines are considered to be generic. Thus this guidelines focus on the way given functionality can be implemented, and additionally how new functional requirements can be incorporated.

The main objectives of this technical specification are the following:

- To support the decision-making process about Local Information System design and development in the way that it suggests clear questions to answer and provide implications depending on given answers.
- To suggests a method to estimate time costs and financial investments for building, installing and maintaining an instance of Local Information System.
- To be used as a generic guideline rather than a ready to implement system specification, as each LIS will have his specific implementation details.

Methodology

Starting point of the specification of a local information system should be the LIS functionalities paper, produced in the task I.1.3.2.. Out of the functionality defined there a set of features of the system can be derived. The scope of this feature definition is twofold: Firstly to bridge between the end-user viewpoint and the developer perspective, and secondly, to define the means for the LIS manager and decision maker to decide which desired functionalities should finally be implemented. Features allows identifying components to use as building blocks of a Local Information System.

The methodology used throughout this volume is called “separation of concerns”. According to this methodology, any system needs be observed from several points of view, or perspectives (e.g. user, decision maker, developer). These points of view represent dimensions in some imaginational hyperspace. Depending on how one looks at the system, it may be split into functions, features, components, classes, aspects, subjects etc. All “building bricks” mentioned above, usually overlap and therefore cannot fit in one “dimension”. For example, one feature can be scattered across several functions and comprise of several components. Within each considered dimension, concerns of every part of the system should be identified and then separated. In the guidelines we will consider three main points of view and thus, three dimensions, as shown in the following table:

Perspective	Requirements	Modularity unit
User	Functional requirements should be specified with respect to typical use cases for this kind of Information System.	Function
Decision maker	The whole system should be presented as consisting of the essential part and a set of features that are not essential but may be chosen to be implemented. Costs of each part and its impact should be known. Features should be “additive” to facilitate the decision making process.	Feature

Developer	Components, main building blocks of the system, should be designed to ease the implementation and maintenance.	Component

Functions

Functions reflect the functionality which the system must provide to be of value to its user. They are based on typical use cases for the kind of systems considered. These use cases are described in the form of functional requirements. Not all functions listed in the LIS functionalities paper must be implemented obligatory. Some of the might seem very useful for end-users but cost to much developing efforts in practice. However, the core functional needs must be fulfilled in any case. That is why, it is recommended to mark all functions either as core functions or optional ones. For optional functions the implementation costs should be considered and compared to the expected benefit.
The mechanism of features aims at supporting the decision maker in such situations. Functional requirements will be a starting point for the analysis performed in this technical specification paper. Based on a first a rough functional decomposition of the system, interdependencies between functions and sub-functions can be found. These interdependencies provide the means for decision making. Examples of functions in the functional decomposition are:

- Searching for the metadata
- Editing metadata records
- Uploading documents and connecting them to the records

Features

Features can be defined as properties of the Local Information System that may be chosen to be implemented in a certain installation. One feature may include one or more functions, so they might be also seen as groups of functions. But in contrast to the groups of function that we can see in the functional specification, where the groups unite functions that are usually used together, in one use case, features divide all functions into parts that are important for the developer. One feature groups functions that might be (and ideally should be) implemented independently of others.
Features enrich the core system functionality in different directions. Projecting a Local Information System, LIS manager or other decision maker should find a reasonable compromise between functionalities and implementation resources. Features help to do so. Examples of features in the feature-oriented decomposition are:

- Access control. Manages the access to the metadata records stored in the Local Information System.
- Record templates. Enables create new record from existing templates rather than from the scratch.
- Internationalisation. Makes possible to render the user interface in different languages.

Features might be seen from both user and developer perspectives and, therefore, can be effectively used to bridge functions and components, or, in other words to define components implementing desired functions. From the user viewpoint features bring additional functionality to the system in the way that they enables new functions and affect existing ones. For example:

- Access control. Brings new functions that allows defining access control policy. Depending on the complexity of the access control mechanism such functions as role managements, access rights management might be included. This feature also affects all functions that somehow work with the metadata storage, since it takes the control over it and hence manipulates the scope of all operations.
- Record templates. Enables two new functions, namely creating a record from the existing template and saving a record as template. Only these two functions are affected because the template mechanism is transparent to the rest of the functionality.
- Internationalisation. Needs one function to be added: “Select language locale”, which, however, may be optional. User might use the default setting of the client program (like Web browser) or operation system if the system is able to detect it.

From an other point of view, features represent a guideline for the developer. Taking into account the fact that features include functions, not necessarily presenting in the core functionalities list, their insertion into the system must be as seamless as possible, otherwise evolving user requirements may make the development process extremely ineffective.

Feature-based decomposition provides a tool for making precise decision about the inventory of the system. In order to simplify the decision making process, features are made additive, i.e. the recourses needed for implementing a set of features will be approximately the sum of the resources needed for implementing each feature individually. From the other point of view, a feature is a set of modules that is normally transparent to other parts of the system and hence can be seamless added or removed. Features appear in the analysis when the functions are detailed enough to determine functional parts that lie across in several branches of the functional hierarchy. A feature normally can be implemented by a set of interacting components
Relation between functions and features and their orthogonality is depicted on the following figure.

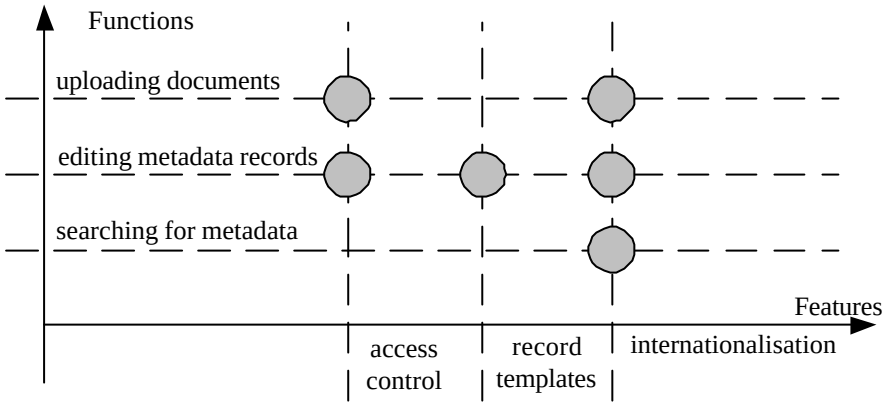


Figure 22

Components

Components are the main building blocks of a Local Information System. As in the case of functions and features, the whole system may be broken down into relative simple and independent components. The main focus of components, however, is to ease implementation and maintenance. Components may be treated as parts of features. Deciding to include a feature to a Local Information System implementation, one actually decides to implement components representing this feature. Components are described here in the terms of the Abstract component architecture (see chapter 1.2.5.). In this architecture a component is thought as a system part, providing a single service. Identification of such services is a very important task of the system architect. When services are well separated across the components, the system is easier to reuse and maintain. The Abstract component architecture assumes each component to have three properties: role (identifier of the service), interface (type of service), implementation and configuration. These three properties are independent. It means, for example, that one implementation might play different roles, provided with corresponding configurations.
Examples of components in the component-oriented decomposition are:

- Record editor. Components, encapsulating operations changing the metadata record such as adding or removing element, filling out field etc.
- Metadata storage manager. Provides a service for accessing the metadata storage and performing such operations as storing and retrieving records, search etc.
- Transformer proxy. This component allows using the universal record format in the metadata storage. Although records might be originated with different structure, they should be all transformed to some universal format to leverage the pervasive search.

Specifications: Overview

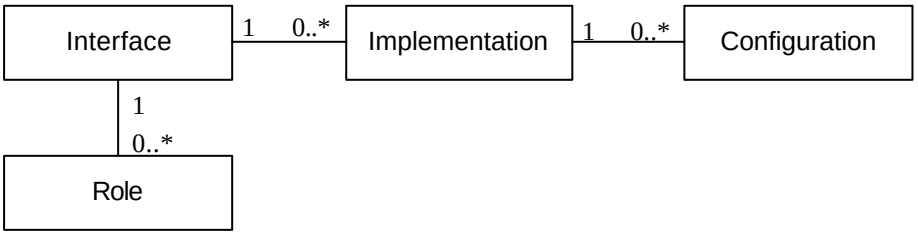
In the next chapters, the decomposition of a Local Information System from three different perspectives is conducted. It starts with the functional specification that lists functional requirements that are put upon such a system. It proceeds with the feature specification including list of features that might be reasonable for a system. As this specification is addressed to the decision maker, so the implementation, temporal and financial impact is included. However, neither list of functions nor feature list must be seen as exhaustive: both may be extended according the methodology presented here. Every feature specification contains the "Decomposition" section that is indented for developers and system architects. It shows the design steps that lead from the feature specification to the component-oriented decomposition of the system. Components are defined in the terms of the abstract component architecture that can be implemented on the base of popular component architectures, like the Apache Avalon or Enterprise Java Beans. These sections might be especially useful when extending function or feature specification with new functions and features. The decomposition description concludes with the component specification that is actually the result of the design process described in the "Decomposition" sections. In this specification the components are described, in contrast to the "Decomposition" sections, in the terms of their interfaces. This chapter is addressed mainly to the developers.

Abstract component architecture

Components used for modelling a LIS implementation are defined here in the terms of the abstract component architecture. Some main ideas for these architecture has been adopted from the Apache Avalon project [9]. Base facts about the components are:

- Component acts as a black box, providing a service to other components.
- Component provides only a single service.
- Component has a well-defined lifecycle.
- Components are placed into the container that ensures performing of the lifecycle for every its component.

The lifecycle of a component includes following steps: initialisation, configuration, start, service, stop. Container possess a set of components and is in a charge of performing their lifecycle. The steps of the lifecycle of the particular interest are configuration and service. On the configuration step any component may be provided with a tree-wise configuration. All component configurations are stored and managed centrally that makes the architecture very flexible. On the service step a component may obtain a reference to others. To do this, the component must know the role of the component it needs. Role is associated with some interface and can be also seen as identifier for the service that the component provides. Role-based inter-component binding gives the possibility to replace components in the container without affecting others, since they rely only on the roles of other components and their interfaces and not on a certain implementation. The following figure shows the cardinal relations between roles, interfaces, implementations and configurations of components.



- **Role** is actually a name under which a certain service is know within the component architecture. In a distributed system their roles usually managed by so-called Naming Services.
- **Interface** is a set of functional agreements that a component should satisfy to be able to provide the service. In terms of a programming language, interface is a set signatures of

component's public functions that may be called. Every role implies an interface, otherwise a found service could not be used. However, one interface can be used in more than one services. An example of such a case is the metadata storage interface that may be used as the main record storage as well as the storage for record templates. Saying in other words, roles allow the components find each other while interfaces allow them to communicate to each other.

- **Implementation.** Every interface may be implemented in different ways depending on specific functional requirements or other conditions. One of the main gains from such architecture is the possibility to change the implementation of individual components. Obviously, any implementation implies the interface.
- **Configuration.** To make the components even more reusable, we allow them to have a configuration. Configured differently, one implementation might be used for several roles. Recalling the previous example, one can say that the main storage and the template storage may be implemented the same way with all difference in database URL, where the records are stored. It is wise to reuse one implementation instead of making two different.

Below grammatical structure of a container's configuration can be found. Comma is used here as a delimiter in a sequence and asterisk sign for denoting zero or more entities of the same form.

Container configuration :- (Component configuration)*

Component configuration :- (Role, Implementation, Configuration)

Container configuration is a list of the configurations of its components. Each entry in this list specifies Role as a service identifier, implementation of component (for example name of Java class) and actual component's configuration.

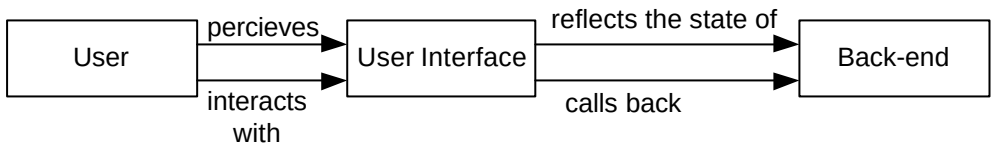
On the service step of the lifecycle, container serves the components, giving them references to others against their roles. As it can be seen further on, there are cases when several components with the same role are desired. An example of such a case is inserting a proxy between two components to affect their intercommunication in a transparent fashion. If several proxy components are present, they must be ordered in a certain sequence. One solution for this problem could be extending the component description part with a numeric or string value that denotes the position of the component in the role's sequence. Having these positions, container is capable to organize service the way that all component with the same role are chained properly. Another possibility is to use the "natural" order of elements in the container's configuration that is present, for example, in the case of XML configuration.

Separation of back-end and front-end

Discussion on the system decomposition process is divided into two main branches, one concerns back-end and other - front-end components. Obviously, front-end modules must be separated from the back-end components in order to make latter independent of chosen front-end platform. For example, the server-side metadata editor must be usable with the HTML-based user interface as well as with Swing-based one etc.

First, the main components of a front-end system are named. Usually the user interacts with the system through an area on the screen or other device which contains passive content (in form of text or graphics) and widgets (active content). According to [6], widget is a way of using a physical input device to input a certain type of value. Some of widgets are used only in the inter-widget

communication while other are intended to cause certain actions on the back-end side. The way these actions are invoked is also known as call-backs.



The user interface communicates with the back-end components in two ways: firstly, it is invoked to reflect the current state of the back-end components and present it to the user and, secondly, it makes call-backs when the user interacts with certain widgets. Both ways of communication are strongly dependent on the toolkit used for the presentation and the interaction with the user. For example, in object-oriented toolkits, such as Java Swing, reflecting the state of the back-end may involves a transformation of the state information into the object structure of widgets and call-backs may be performed directly using Java method calls. In the Web-oriented architecture, where widgets are mark-up elements and the user input is available in the form of “name-value” pairs, the situation changes dramatically. Instead of building object structure, a mark-up is created, and direct call-backs are no longer possible. It is needed to somehow map the “name-value” pairs to the back-end method calls. In [7] one solution of the problem for Web applications is shown. The idea is to use names in “name-value” pairs to encode the call-back actions in one of scripting languages, such as OGNL (Object Graph Navigation Language). Having arrived to the Web-server, these names are decoded and corresponding actions are performed in the proper order with the proper parameters. Also according to this solution, back-end components provide the information about their state mostly in the XML format, so it can be transformed afterward into the proper mark-up using XSLT transformation technique.

User Manual Creation

The User Manual Creation is tightly connected to the functional specification of step 1. For every function a generic description which has to be translated to the concrete user manual of a Local Information System. For every function which is implemented by a Local Information System the generic description has then to be listed under the title in the user manual. The generic description has to be modified according to the listed adoptions.

Interoperability of Local Information Systems

Since the guidelines are generic, they do not restrict a Local Information System to be implemented on the basis of certain platforms or technologies. However, according to the main objective of such systems – improved information exchange, a sufficient level of interoperability has to be provided. The basis for the interoperability, proposed here is the common data model. This data model must be flexible and extensible enough to anticipate possible applications and future needs of a Local Information System. That leads normally to the relative complexity of the data model. If then the user interface is build on the top of the data model, the latter appears to be too complicated for comfortable use. Normally a group of user is interested only in one or several distinct subject areas and does not actually need all the diversity of the data model. One of reasonable workarounds for this problem is to define specialized views on top of the data model. These view encompass only limited part of data model, only data entries that are relevant for the subject area. Such views might be defined using the same descriptive technology (language). Returning to the interoperability issue, it is necessary, that the records, produced in a Local Information System conform to the common data model, otherwise system on different sites would not be able to exchange information. The conclusion is that some kind of transformation of records between common data model and other model, based on specific views, is needed. In order to define such transformation, one has to take into account the following:

- If the common data model is based on any standards, the understanding of semantics of the standard is necessary. Normally transformation includes a mapping of some

field in the common data model to the corresponding field(s) in the view as vice versa. This mapping must be chosen with care in order not to violate standard's clauses about the semantics of data entries.

- Data types in the common data model may differ from those in specific views. It can even occur, that one field is mapped to several ones. Consequently, proper type casting is needed.
- Some views may contain implicit information that is not entered by the user, but is implied when the transformation to the common data model is performed. In the reverse transformation, such piece of data are usually discarded.

1. STEP:
FUNCTIONAL SPECIFICATION

FUNCTION:	**II.1.1.**	DEFINE GEOGRAPHIC CRITERIA OF SEARCH BY TYPING A GEOGRAPHICAL LOCATION
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

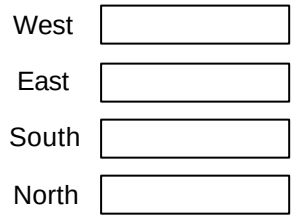
DESCRIPTION

The function allows the user to enter the spatial coverage of a metadata record by typing the coordinates of the bounding rectangle. In the most simple case there are four fields to fill:

- western-most coordinate of the limit of the dataset extent, expressed in longitude in decimal degrees (positive east)
- eastern-most coordinate of the limit of the dataset extent, expressed in longitude in decimal degrees (positive east)
- southern-most coordinate of the limit of the dataset extent, expressed in latitude in decimal degrees (positive north)
- northern-most coordinate of the limit of the dataset extent, expressed in latitude in decimal degrees (positive north)

In this case a default geographic reference system is implied.

VISUAL SCHEMA



FEATURES

- II.2.1. Search

SUBFUNCTIONS

- II.1.2. Specify reference system

HUMAN RESOURCES (FRONT-END)

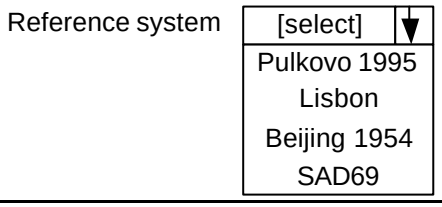
1 person day

SUBFUNCTION: II.1.1.1.	SPECIFY REFERENCE SYSTEM
FUNCTION: II.1.1.	DEFINE GEOGRAPHIC CRITERIA OF SEARCH BY TYPING A GEOGRAPHICAL LOCATION

DESCRIPTION

When it is necessary to enter the special coverage of metadata records, expressed in other units than latitude and longitude in decimal degrees, a reference system must be specified along with coordinates. Since providing a possibility to choose a reference system implies the need of coordinate transformation, only the reference systems, supported by an available coordinate transformation module should be options.

VISUAL SCHEMA



FEATURES

- II.2.2. Coordinate transformation

HUMAN RESOURCES (FRONT-END)

1 person day

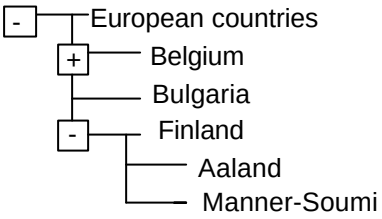
FUNCTION:	**II.1.2.**	DEFINE GEOGRAPHIC CRITERIA OF SEARCH BY SELECTING A LOCATION IN A GEOGRAPHICAL THESAURUS
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to pick up the coordinates for the spatial coverage directly from the geographical thesaurus. Geographical thesaurus is a tree, representing the hierarchy of the administrative units: countries, states, regions, lands etc. Each entity in the geographic thesaurus is associated with the set of coordinates that are provided as a search criteria when the entity is selected. Selecting a node in the tree, the user picks the corresponding term up and adds it to the search criteria.

This function can be seen as an alternative to the function II.1.1.

VISUAL SCHEMA



FEATURES

- II.2.3. Thesaurus

HUMAN RESOURCES (FRONT-END)

10 person days

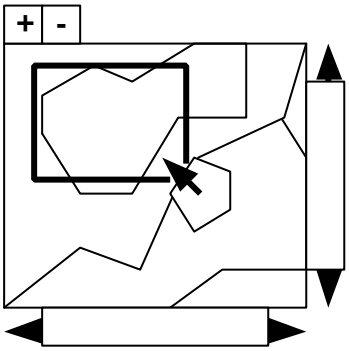
FUNCTION:	**II.1.3.**	DEFINE GEOGRAPHIC CRITERIA OF SEARCH BY DELIMITING AN AREA ON AN INTERACTIVE MAP
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function gives the user possibility to delimit a geographic region on a interactive map. The map shows a part of the globe or a continent and allows such actions as:

- Scroll. Shows the region next to one on a map (in four or eight directions)
- Zoom in. Changes the scale of the map so that selected sub region can be seen entirely
- Zoom out. Changes the scale of the map so that the current viewed region will be smaller and located in the centre of the map
- Select region. Transfers the coordinated of the selected region back to the calling component.

VISUAL SCHEMA



FEATURES

- II.2.4. Map

HUMAN RESOURCES (FRONT-END)

20 person days

FUNCTION:	**II.1.4.**	DEFINE THEMATIC CRITERIA OF SEARCH BY TYPING FREE KEYWORDS
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to add free text keyword to the metadata records. Free text keyword might be needed when the desired keyword cannot be found in the proposed thesaurus. Only one field for such a keyword is needed, since neither unique identifier nor thesaurus reference are relevant.

VISUAL SCHEMA

Keyword to search [ ]

FEATURES

- II.2.1. Search

HUMAN RESOURCES (FRONT-END)

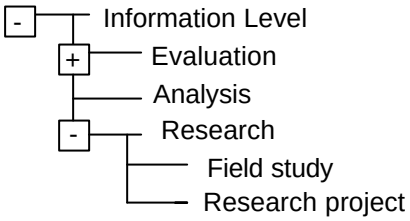
1 person day

FUNCTION:	**II.1.5.**	DEFINE THEMATIC CRITERIA OF SEARCH BY SELECTING KEYWORDS IN A THEMATIC THESAURUS
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to pick up a keyword from a thesaurus. This way of entering keyword is always more preferable than entering free keywords. Terms (keywords) in the thesaurus are assigned to their unique identifiers and therefore language independent. Terms should have a clear hierarchy set by “broader-narrower” relationships. In other words, more general (broader) terms have more specific (narrower) terms as children in this hierarchy.

VISUAL SCHEMA



FEATURES

- II.2.3. Thesaurus

HUMAN RESOURCES (FRONT-END)

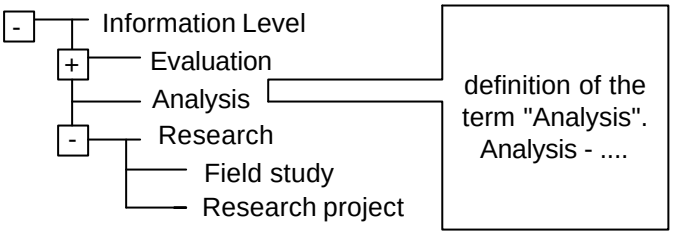
10 person days

FUNCTION:	**II.1.6.**	CONSULT THE DEFINITION OF A SPECIFIC TERM VIA AN INTERACTIVE GLOSSARY
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to view terms' definitions when browsing the thesaurus. The glossary might be organized in the form of hyper text, so that the user can use links to go from a term definition to the related terms.

VISUAL SCHEMA



FEATURES

- II.2.3. Thesaurus

HUMAN RESOURCES (FRONT-END)

3 person days

FUNCTION:	**II.1.7.**	FIND OUT THE LIS DATA REPOSITORY WHICH DATA SOURCES ARE MATCHING THE SELECTED CRITERIA
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

Constructs a query, collecting values of the search criteria from the form and executes it.

VISUAL SCHEMA

Coordinates

Keywords

Other criterias

Search

FEATURES

- II.2.1. Search

HUMAN RESOURCES (FRONT-END)

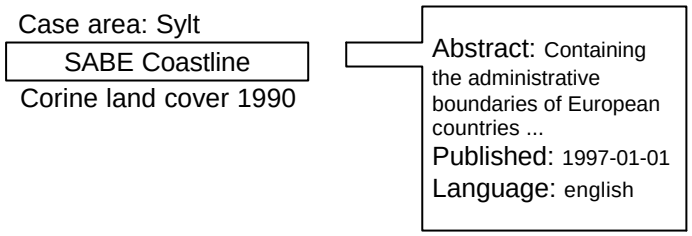
5 person days

FUNCTION:	**II.1.8.**	SELECT A DATA SOURCE IN A LIST OF ITEMS MATCHING THE SELECTED CRITERIA AND VIEW ITS RELATED METADATA
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

Enables browsing the result of a query as a list of found metadata records. Provide the user with the possibility to view metadata records corresponding to the entries of the result list

VISUAL SCHEMA



FEATURES

- II.2.1. Search

HUMAN RESOURCES (FRONT-END)

5 person days

FUNCTION:	**II.1.9.**	IF AVALIABLE, DOWNLOAD THE DATA
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function gives the user a direct access to the data described by a metadata record, if it is available. These data might be previously uploaded to the server or made accessible via an specified URL.

VISUAL SCHEMA

Case area: Sylt	Download
SABE Coastline	Download
Corine land cover 1990	Download

FEATURES

- II.2.5. Linking

HUMAN RESOURSES (FRONT-END)

1 person day

FUNCTION:	**II.1.10.**	IF AVAILABLE, VIEW THE DATA ITSELF WITH AN APPROPRIATE VIEWER (WORD, EXCEL, ETC.)
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

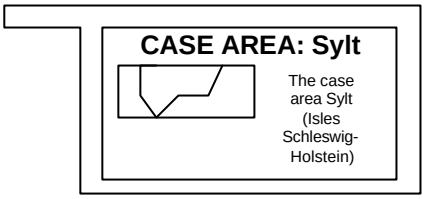
The function allows user to view the corresponding data file with an appropriate viewer after downloading it. The viewer is supposed to be present in the operating system and associated with the right content type.

VISUAL SCHEMA

Case area: Sylt [Download]

SABE Coastline [Download]

Corine land cover 1990 [Download]



FEATURES

- II.2.5. Linking

HUMAN RESOURCES (FRONT-END)

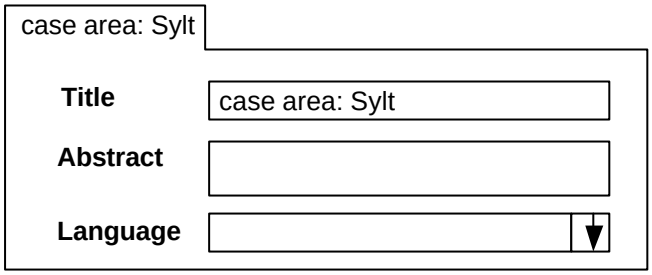
1 person day

FUNCTION:	**II.1.11.**	EDIT AND SAVE NEW METADATA RECORDS
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

Enables creating new metadata records and loading already created for further editing.

VISUAL SCHEMA



FEATURES

- II2.6. Editor

SUBFUNCTIONS

- II.1.11.1. Create metadata record
- II.1.11.2. Choose record type
- II.1.11.3. Create metadata record from Template
- II.1.11.4. Load metadata record in the editor
- II.1.11.5. Edit metadata record
- II.1.11.6. Save metadata record
- II.1.11.7. Save metadata record as Template
- II.1.11.8. Close record in the editor
- II.1.11.9. Choose record

Note: These sub-functions are not described individually.

HUMAN RESOURCES (FRONT-END)

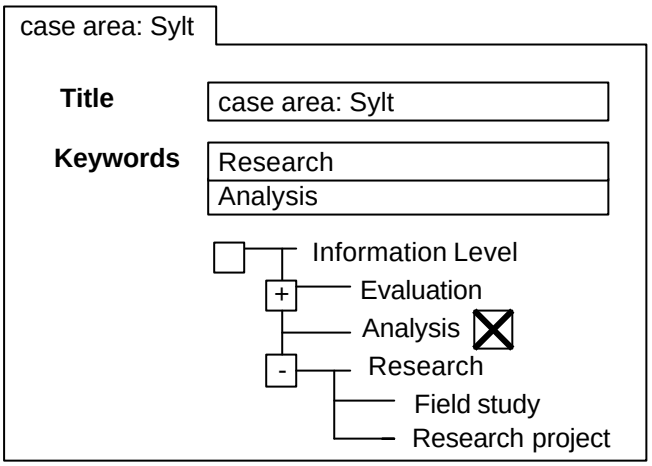
20 person days

FUNCTION:	**II.1.12.**	ATTACH TO THE RECORD A SET OF KEYWORDS FROM A THESAURUS
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function aims at allowing the user to pick up a set of terms from the thesaurus and to attach them to the metadata record that is currently being edited. These keywords are used further to perform the thematic search.

VISUAL SCHEMA



FEATURES

- II.2.3. Thesaurus

HUMAN RESOURSES (FRONT-END)

2 person days

FUNCTION:	**II.1.13.**	DEFINE THE ACCESS RIGHTS TO DOWNLOAD OR VIEW A DATA FILE
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows to introduce restrictions applied when the data file associated with a metadata record is being accessed. Depending on the granularity and policy of the access control, there could be different options to set. Here the table is shown where column depict viewers of data files. Rows represent owners of data files, in other words, users who have uploaded the metadata on the server.

VISUAL SCHEMA

Viewer	test	administrator	LISmanager	John Clark
Owner				
test				
administrator				
LISmanager				
John Clark				

FEATURES

- II.2.10. Access control

HUMAN RESOURSES (FRONT-END)

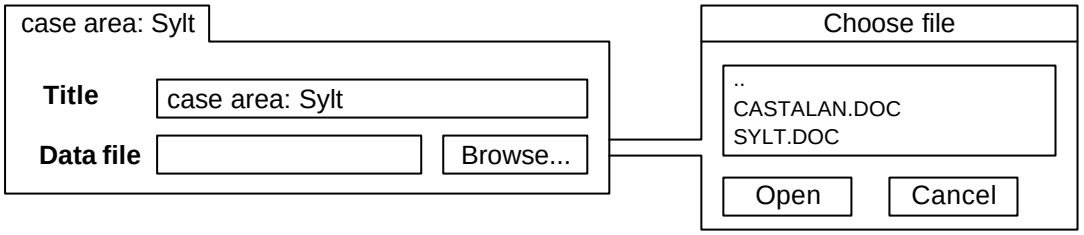
10 person days

FUNCTION:	**II.1.14.**	ATTACH TO THE RECORD A DATA FILE
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to upload a data file from the local file system and attach it to a metadata record. When uploaded, files are effectively moved from the client computer to the server. Having uploaded a file, one gets the link to the uploaded copy on the server, so the file is now accessible for other users. Uploading is performed by clicking on the “Browse” button that opens standard file choose dialog.

VISUAL SCHEMA



FEATURES

- II.2.5. Linking

HUMAN RESOURSES

2 person days

FUNCTION:	**II.1.15.**	ACCESS ONE OR SEVERAL FORA RELATED TO COASTAL ZONE MANAGEMENT
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to see the discussions on the forums designated to the coastal zone management theme. On the topmost level the user can observe the list of available forums. The following information about a forum might be shown in the list:

- Name of the forum. Gives an idea about the content
- Number of topics (threads) in the forum
- Number of messages posted to the forum. Can server as an indicator of forum's popularity
- Last message posted to the forum. Shows the presence of "live" topics in the forum

When one of the forums is selected, the list of topic in this forum is shown. The user can see then the following information about the topics:

- Name of the topic (thread). Usually contains a question
- Number of answers. Shows the activity of the users answering the question of the topic or giving their comments
- Number of view. Shows how many user have read the topic.

VISUAL SCHEMA

List of forums			
Forum	Topics	Messages	Last message
Coasts of Spain	19	132	03.05.2003 John Clark
Administrative forum	9	43	02.05.2003 administrator
Technical questions	11	74	03.05.2003 John Clark
FAQs	14	15	23.04.2003 LISmanager

Forum: FAQs				
Topic	Answers	Author	Views	Last message
What is LIS?	3	test	23	LISmanager
How do I edit metadata records?	4	John Clark	19	administrator
Information import	7	LISmanager	37	test

FEATURES

- II.2.11. Forum

HUMAN RESOURCES (FRONT-END)

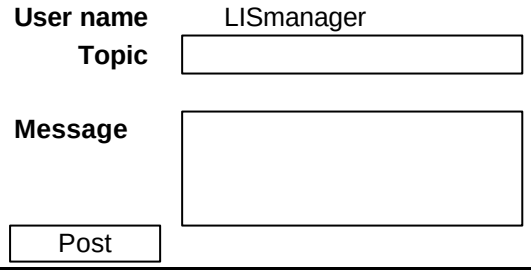
3 person days

FUNCTION:	**II.1.16.**	POST NEW MESSAGE TO A SPECIFIC FORUM
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to participate in forums by posting messages to them. Since the user is already logged in when posting messages, it is necessary only to enter topic name and the message.

VISUAL SCHEMA



FEATURES

- II.2.11. Forum

HUMAN RESOURCES (FRONT-END)

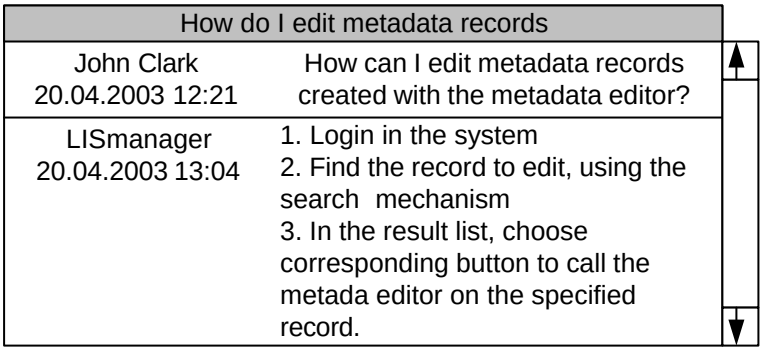
3 person days

FUNCTION:	**II.1.17.**	READ PREVIOUS MESSAGES POSTED TO A SPECIFIC FORUM
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to view messages posted to a specific forum. Information in a forum is organised in the form of topics and topics contain messages.

VISUAL SCHEMA



FEATURES

- II.2.11. Forum

HUMAN RESOURCES (FRONT-END)

3 person days

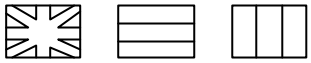
FUNCTION:	**II.1.18.**	SELECT THE LANGUAGE FOR THE CRITERIA OF SEARCH, THE INTERFACE, AND THE GLOSSARY
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to specify the language in which search criteria, user interface, glossary are shown.

VISUAL SCHEMA

Select language



English German French

FEATURES

- II.2.12. Internationalisation

HUMAN RESOURSES (FRONT-END)

2 person days

FUNCTION:	**II.1.19.**	MODIFY OR DELETE EXISTING METADATA RECORDS
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to get an access to already created metadata records and modify or delete them. In order to specify which record should be modified or deleted, the user picks it up from the record list.

VISUAL SCHEMA

Case area: Sylt	Edit	Delete
SABE Coastline	Edit	Delete
Corine land cover 1990	Edit	Delete

FEATURES

- II.2.6. Editor

HUMAN RESOURCES (FRONT-END)

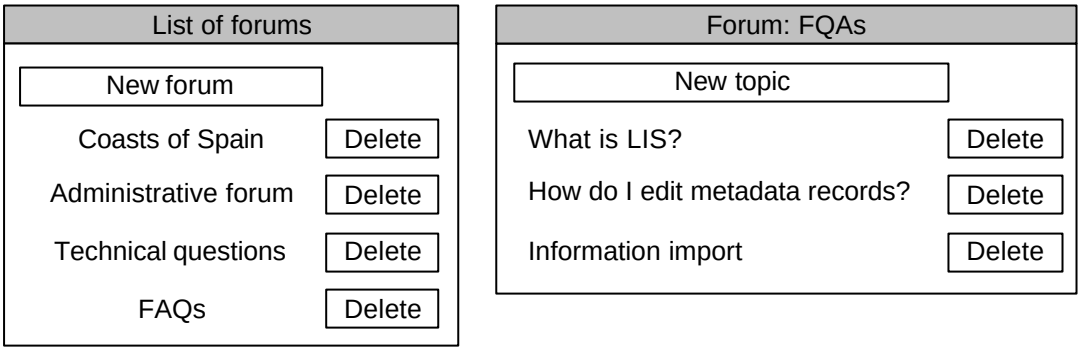
3 person days

FUNCTION:	**II.1.20.**	CREATE OR DELETE A NEW FORUM
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function gives the user a possibility to create or delete a new forum and/or a thread of the discussion in the forum. These actions should be performed by a user with corresponding rights on the forum (administrator, moderator).

VISUAL SCHEMA



FEATURES

- II.2.11. Forum

HUMAN RESOURCES (FRONT-END)

2 person days

FUNCTION:	**II.1.21.**	CREATE, MODIFY, DELELE USER PROFILES WITH SPECIFIC ACCESS RIGHTS
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to manage the access rights of other users, restricting the access to the metadata records. There might be different options to set depending on the access control policy. Using this function, the user, managing the access control can specify the settings, inherited by the metadata records when they have been created by specific user. Here 3 levels of the access are considered:

- Read. User can view metadata record, but cannot change it
- Write. User can edit metadata record or delete it
- Manage. User may change the access control options of metadata records, created by a specific user.

VISUAL SCHEMA

Rights of	test	administrator	LISmanager	John Clark
Owner				
test	Write	Manage	Write	Write
administrator		Write	Read	
LISmanager		Manage	Write	Read
John Clark		Manage	Write	Write

FEATURES

- II.2.10. Access control

HUMAN RESOURCES (FRONT-END)

5 person days

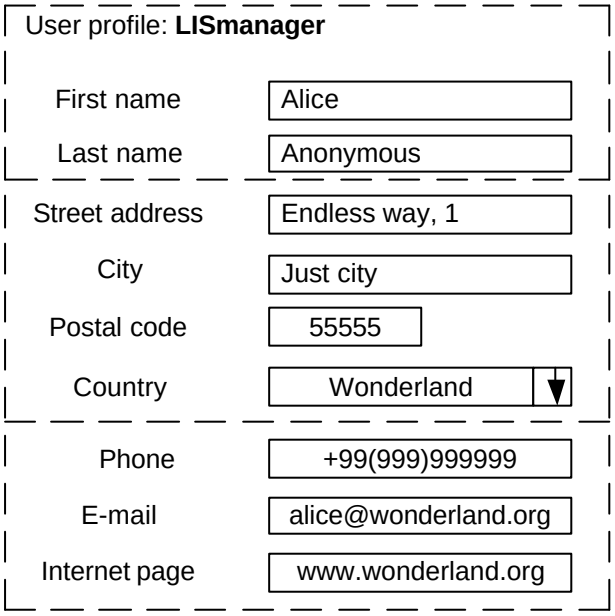
FUNCTION:	**II.1.22.**	CREATE, MODIFY, DELETE NEW USERS WITHIN EACH USER PROFILE
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to view and change information in his/her user profile. The data in this profile are used for contacting purposes and contain following entries:

- Name of the user (first name, second name, etc.)
- Address information (street address, city, county, postal code)
- Telephone and fax numbers
- E-mail addressed and links to the Internet page with more detailed information about the personality

VISUAL SCHEMA



FEATURES

- II.2.13. Users

HUMAN RESOURCES (FRONT-END)

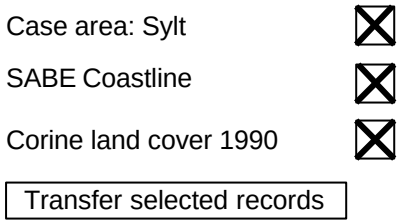
5 person days

FUNCTION:	**II.1.23.**	REGULAR TRANSFER OF METADATA, AND IF AVAILABLE DATA THEMSELVES, TOWARDS EUROSION SERVER
CHAPTER:	**II.1.**	FUNCTIONAL SPECIFICATION

DESCRIPTION

The function allows the user to perform regular transfer of the metadata, entered in the LIS, towards the Eurosion server. This facilitates the information exchange process, since after the transfer the metadata and attached data (if available) become available to the other sites.

VISUAL SCHEMA



FEATURES

- II.2.14. Transfer

HUMAN RESOURCES (FRONT-END)

3 person days

FEATURE SPECIFICATION

FEATURE:	**II.2.1.**	SEARCH
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

The feature groups together functions, aiming at performing metadata search.

FUNCTIONS

- II.1.1. Define geographic criteria of search by typing a geographical location
- II.1.4. Define thematic criteria of search by typing free keywords
- II.1.7. Find out the LIS data repository which data sources are matching the selected criteria
- II.1.8. Select a data source in a list of items matching the selected criteria and view its related metadata

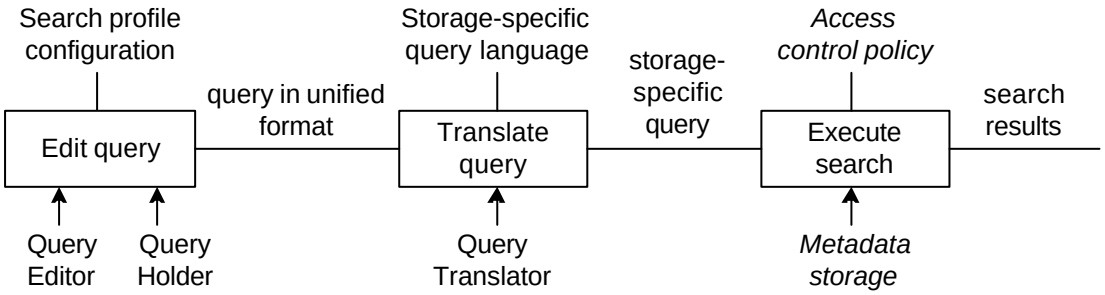
MAIN ASPECTS

- Extensibility of the search form
- Easy integration with the external applications, such as map viewers, thesaurus browsers, etc.
- Independence of search interface on the type of the metadata storage

DECOMPOSITION

Functions II.1.1. “Define geographic criteria of search by typing a geographical location” and II.1.4. “Define thematic criteria of search by typing free keywords” are interdependent since they aim at entering different criteria. The values of the search criteria are entered in some kind of search form. This form contains, for example, free text field, date fields, as well as field with special manner of input, such as a geographical map, where one may stretch a bounding box, thesaurus tree etc. The same special input can be also found in the editor’s forms. **f** they look the same way, the user of the system will not be confused associating corresponding fields of the search form and in the editor. The search form contains a small part of fields that are present in the forms of the editor. It goes from the fact that when searching, one actually matches a part of a record with complete records in the storage in order to find those that satisfy the matching conditions. The list of search criteria may differ from one Local Information System to another. In order to make the search form flexible and easily configurable, the same mechanism as in the record editor may be employed.

DIAGRAM (FUNCTIONAL)



COMPONENTS INVOLVED

- II.3.5. Editor (as QueryEditor)
- II.3.2. Document Holder (as Query Holder)
- II.3.6. Query Translator
- II.3.1. Metadata Storage

FEATURE:	**II.2.2**	COORDINATE TRANSLATION
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

The feature represent a possibility to perform coordinate translations between different geographic reference systems. This feature must be flexible in the way that any new coordinate transformation could be added quickly and easily.

FUNCTIONS

- II.1.1.1. Specify reference system

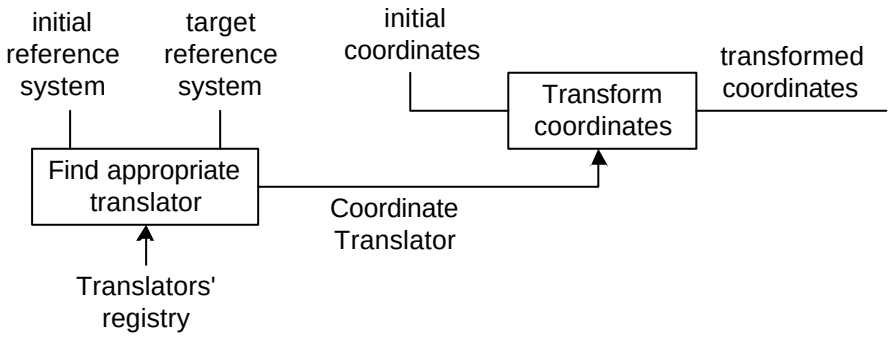
MAIN ASPECTS

- Extensibility in the sense of adding new transformations

DECOMPOSITION

In order to maintain diverse reference systems in the coordinate translation, one needs a unified way to refer them. Some suggestions may be found on the OpenGIS Consortium Web-site (www.opengis.org), for example, in the document named “Recommended Definition Data for Coordinate Reference Systems and Coordinate Transformations” (http://www.opengis.org/techno/specs/01-014r3.rtf). Another challenge is to make the coordinate translation easily tunable. When a new coordinate transformation is implemented, there must be an easy way to incorporate it to the working system without changing any other parts. That is why there should be a translators’ registry where all available translators are registered and then multiplexed when needed.

DIAGRAM (FUNCTIONAL)



COMPONENTS INVOLVED

- II.3.7. Translators Registry
- II.3.8. Coordinate Translator

FEATURE:	**II.2.3.**	THESAURUS
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

The feature groups together function concerning access to an interactive thesaurus, containing in the hierarchical way information about keyword, geographical regions etc. Interactive glossary, associated with the thematic thesaurus is also encompassed with this feature.

FUNCTIONS

- II.1.2. Define geographic criteria of search by selecting a location in a geographical thesaurus
- II.1.5. Define thematic criteria of search by selecting keywords in a thematic thesaurus
- II.1.12. Attach to the record a set of keywords from a thesaurus
- II.1.6. Consult the definition of a specific term via an interactive glossary

MAIN ASPECTS

- Flexible thesaurus that can be edited and extended when needed
- Diverse additional information that may be provided along the thesaurus terms (like glossary content)
- Call-backs towards an editor (query editor or record editor)

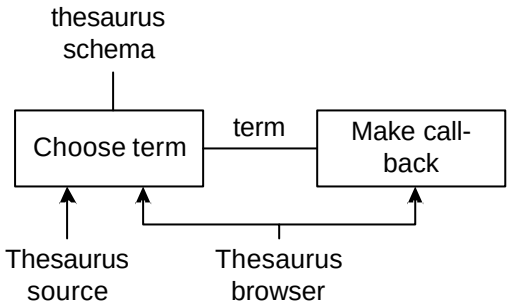
DECOMPOSITION

For maintaining flexible thesaurus that can be extended any time, it is needed to separate the data source, where the content of the thesaurus is stored, from the applications or components that access this source (like thesaurus browser).

Functions II.1.2 “Define geographic criteria of search by selecting a location in a geographical thesaurus” and II.1.5 “Define thematic criteria of search by selecting keywords in a thematic thesaurus” show that thesaurus browser might be employed for semantically different thesauri. The data, picked up from the thesaurus are also different in both cases: id of the selected term and coordinates of the geographical region respectively. In order to make the thesaurus flexible, so-called thesaurus schema might be defined. This schema describes the additional data fields, contained in every thesaurus form along the standard fields, needed for navigation (name of the term, broader term, narrower terms). Interactive glossary, containing terms definitions, can be implemented in the form of additional information provided along the terms.

Another aspect are the call-backs that need to fired towards an editor, since thesaurus browser serves for filling fields in the search form or in the metadata editor.

DIAGRAM (FUNCTIONAL)



COMPONENTS INVOLVED

- II.3.9. Thesaurus source
- II.3.10. Thesaurus browser

FEATURE:	**II.2.4.**	MAP
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

The feature represent a possibility to use an interactive map to specify a geographic bounding box.

FUNCTIONS

- II.1.3. Define geographic criteria of search by delimiting an area on an interactive map

MAIN ASPECTS

- Restoring the geographical region on the map with respect to the data, being edited (in the search form or in the metadata editor)
- Considering the possible difference of coordinate reference systems, used in the map viewer and in the metadata model
- Call-backs towards an editor (query editor or record editor)

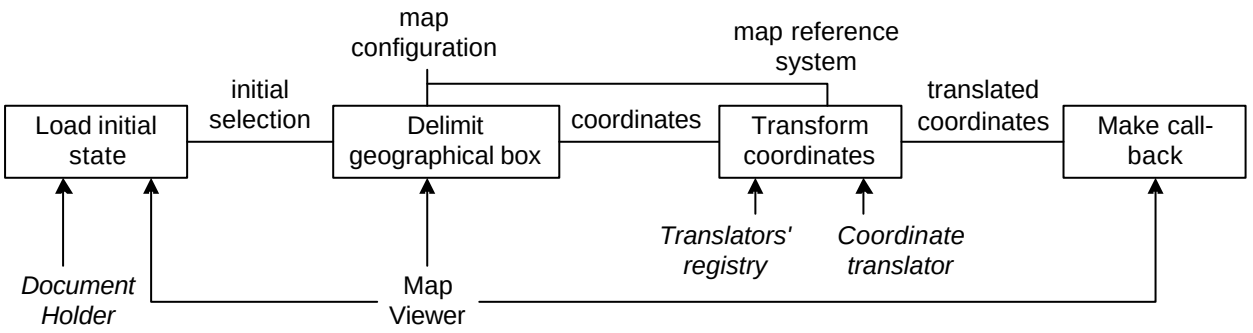
DECOMPOSITION

In the very simple case, the functionality concerning delimiting a bounding box and making the call-back to the query or record editor might be provided by a single component, Map Viewer. All Information about the map's representation and appearance is contained in the map configuration. This configuration can refer to shape files, coordinate reference systems of the map and the metadata model.

If needed, the initial state of the selection can be read from the Document Holder to render the coordinates that are already entered into the record or into the search form.

If the coordinate reference system, used with the interactive map differs from the reference system, used in the metadata model, they need to be transformed before the callback towards an editor is made.

DIAGRAM (FUNCTIONAL)



COMPONENTS INVOLVED

- II.3.11. Map Viewer
- II.3.2. Document Holder
- II.3.7. Translators Registry
- II.3.8. Coordinate Translator

FEATURE:	**II.2.5.**	LINKING
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

The feature groups together functions that provide uploading, downloading and view of the data, attached to metadata records.

FUNCTIONS

- II.1.9. If available, download the data
- II.1.10. If available, view the data itself in an appropriate viewer
- II.1.14. Attach to the record a data file

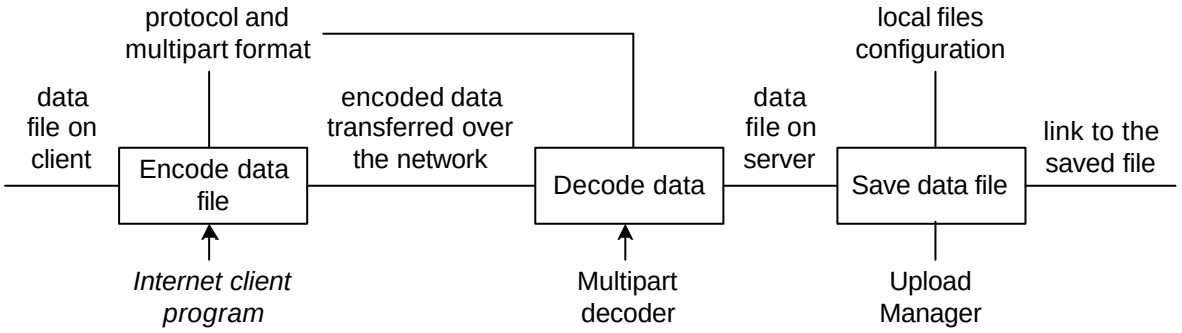
MAIN ASPECTS

- Leveraging the capability of internet client programs to accept data files as form fields (for example, Web browsers)
- Centralized management of the saved file to enable access control for the uploaded data files

DECOMPOSITION

The functions I.1.9 “If available, download the data”, I.1.10. “If available, view the data itself in an appropriate viewer” and I.1.14. “Attach to the record a data file” are interdependent since the data, uploaded by the latter, can be downloaded and viewed by others. When a data file is uploaded to the server it is first encoded by the client program into the format allowing to send it over the network. On the server side this message has to be decoded, used the reverse algorithm and stored in the local file system of the server or in a database. Having saved the decoded data file on the server, the Upload Manager component produces a hyperlink that can be used then for accessing the uploaded data.

DIAGRAM (FUNCTIONAL)



COMPONENTS INVOLVED

- II.3.12. Multipart Decoder
- II.3.13. Upload Manager

FEATURE:	**II.2.6.**	EDITOR
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

The feature enables creating and editing metadata records within the LIS.

FUNCTIONS

- II.1.11. Edit and save new metadata records
- II.1.11.1. Create metadata record
- II.1.11.2. Load metadata record in the editor
- II.1.11.3. Edit metadata record
- II.1.11.4. Save metadata record
- II.1.11.5. Close record in the editor
- II.1.19. Modify or delete existing metadata records

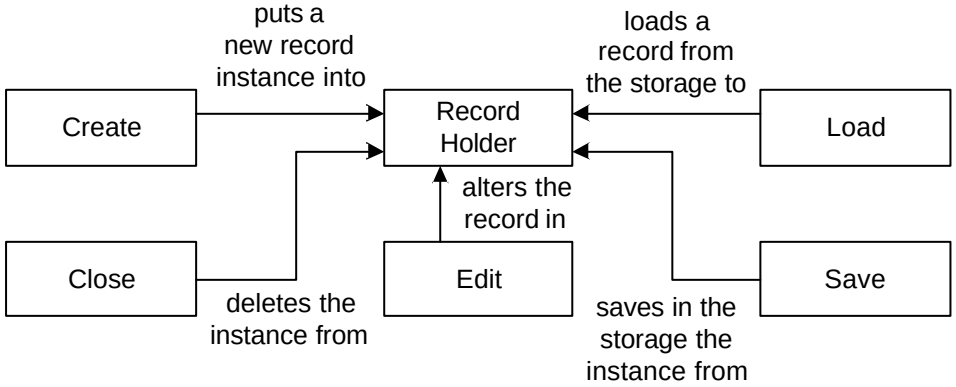
MAIN ASPECTS

- Weak dependency on the used system platform (stand-alone or Web-based application, thin or “fat” client)

DECOMPOSITION

All the sub-functions of the function I.1.11. “Edit and save new metadata records” have a common point of interconnection, namely the working record. Function II.1.11.1 “Create metadata record” puts a new instance to the working record, function II.1.11.2. “Load metadata record in the editor” loads the working record from the storage, function II.1.11.5. “Close record in the editor” removes the instance from the working record, function II.1.11.4. “Save metadata record” writes the working data into the storage. And finally, the function II.1.11.3 “Edit metadata record” allows a user to alter data in the working record. Encapsulating the working record into a separate module named “Document Holder” makes the implementation of these sub-functions independent on the type of the application. In the case of a stand-alone application or “fat” client the working record is saved somehow in the main memory, usually in the form of class member of editor object. Whereas for a Web-based thin client records should be apparently stored inside the HTTP session.

DIAGRAM (COMMAND AND DATA FLOW)



COMPONENTS INVOLVED

- II.3.2. Document Holder (as Record Holder)

FEATURE:	**II.2.7.**	RECORD TEMPLATES
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

Enables creating a metadata record from an existing template and saving edited record as a template. The same functionality could be achieved by allowing copying of records within the storage management. However, this approach would be not straightforward for the user, saying in other words, the user would not be encouraged to use temple technique if it is not explicitly declared.

FUNCTIONS

- II.1.11.6. Create metadata record from Template
- II.1.11.7. Save metadata record as Template

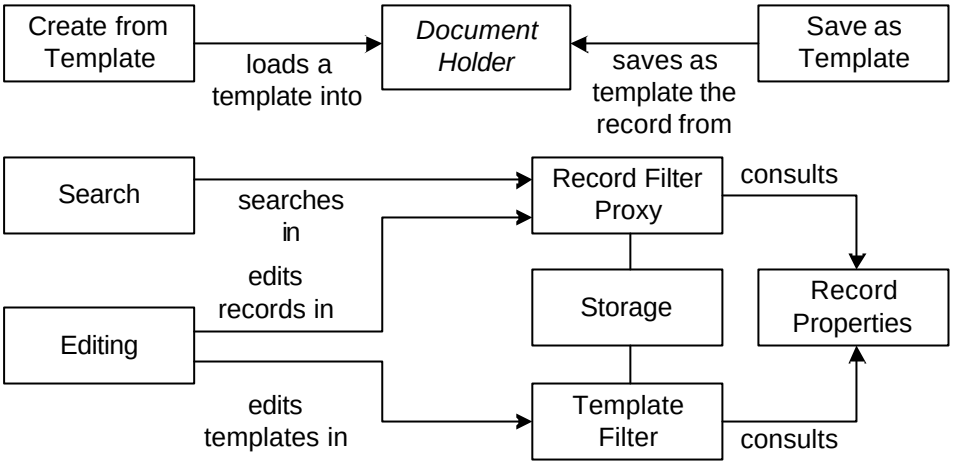
MAIN ASPECTS

- Opportunity to create a new record from an existing template rather than from the scratch
- Producing template from the working record

DECOMPOSITION

Dependency between functions II.1.11.6 “Create metadata record from Template” and II.1.11.7. “Save metadata record as Template” is in the fact, that any record saved as template becomes accessible for the creating. In other words, these two functions share some template storage. Record templates are actually records, which somehow marked to be considered as templates. Using such markers, we can store templates in the same storage where all other metadata records are stored and therefore, leverage transparent access control mechanism. However, other functions operating with the storage should not be affected by record templates, i.e. record templates must be invisible for them. “Create from Template” and “Save as Template” are actually mirrored “Load” and “Save” functions with the difference that they operate with templates (marked records) instead of normal records. “Create new” function emits prepared instance of empty record to the “Edit” function. In order to make template mechanism transparent to other functions, two proxy components for the storage are introduced. They perform mutual exclusive filtering: one filters only records, the other – only templates. Note that these two proxy components have the same interfaces but different roles otherwise other modules could not distinguish between them. More precisely, record filter proxy overtakes the same role as the storage and template filter proxy has its own one. If the template markers were stored directly inside the records, it could cause problems because other components would have to be aware of it. The better way is to store these markers inside a separate storage. This storage is maintained by the Record Properties component.

DIAGRAM (CONTROL AND DATA FLOW)



COMPONENTS INVOLVED

- II.3.1. Metadata Storage (as Record Filter Proxy and Template Filter)
- II.3.14. Record Properties

FEATURE:	**II.2.8.**	MULTIPLE RECORDS
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

Allows several record to be edited (to be opened for the editing) simultaneously.

FUNCTIONS

- II.1.11.8. Choose record

MAIN ASPECTS

- Possibility of editing several records at the same time
- Transparence of the multiple records mechanism to the editing functions

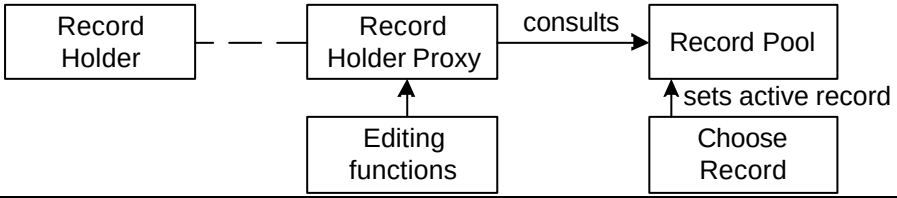
DECOMPOSITION

There are two functions affected by this feature: “Create metadata record” and “Load metadata record”. Comparing the behavior of these functions with and without the feature, we can see that

- “Create” function puts a new record instance into the Record holder. Without the multiple records support, the new instance replaces the one that was in the record holder. With the multiple records support, the record in the holder is moved in some pool, so it is available for further operations.
- “Load” function loads a record from the storage and puts in into the Record holder. Without the multiple records support, the loaded instance replaces the one that was in the record holder. With the multiple records support, the record in the holder is moved in some pool, so it is available for further operations.

In order to implement the behavior differences the proxy component can be introduced. This component mediates the communication between the Record Holder and “Create” and “Load” functions. It consults the pool of records that are available for main editing operations. New function “Choose record” actually does the following: the record in the holder is returned to the pool and chosen record goes from the pool to the holder. In the scenario, this function may precede such functions as “Save”, “Close”, “Edit”, “Save as Template”. However, function “Choose record” cannot communicate with the Record Holder Proxy since a proxy has the same role as the component, accessed via this proxy. Record Holder Proxy does not provide service for choosing record function. Another component, Record Pool is needed. Record Pool possess a pool of records and one of these records may be marked as active by the “Choose record” function. Record Holder Proxy, providing the same service as the Record holder, consults the Record Pool and performs all operations only with the currently active record in the pool. It is up to developer to decide if the Record holder Proxy should use the original Record holder beside the Record Pool. That is why their connection is shown with a dashed line. It means also that all other proxies that are applied upon the Record Holder, must be applied between this Record Holder Proxy and editing functions, otherwise they might be disabled.

DIAGRAM (CONTROL AND DATA FLOW)



COMPONENTS INVOLVED

- II.3.2. Document Holder (as Record Holder Proxy)
- II.3.3. Record Pool

FEATURE:	**II.2.9.**	MULTIPLE RECORD TYPES
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

Enables creating several types of record. These record types may have different structure but must be able to be transformed into some universal format. This universal format is needed in order to enable searching across all record types with the same search mechanism.

FUNCTIONS

- II.1.11.9. Choose record type

MAIN ASPECTS

- Support for several record types
- Transparency of multiple record types mechanism to editing functions

DECOMPOSITION

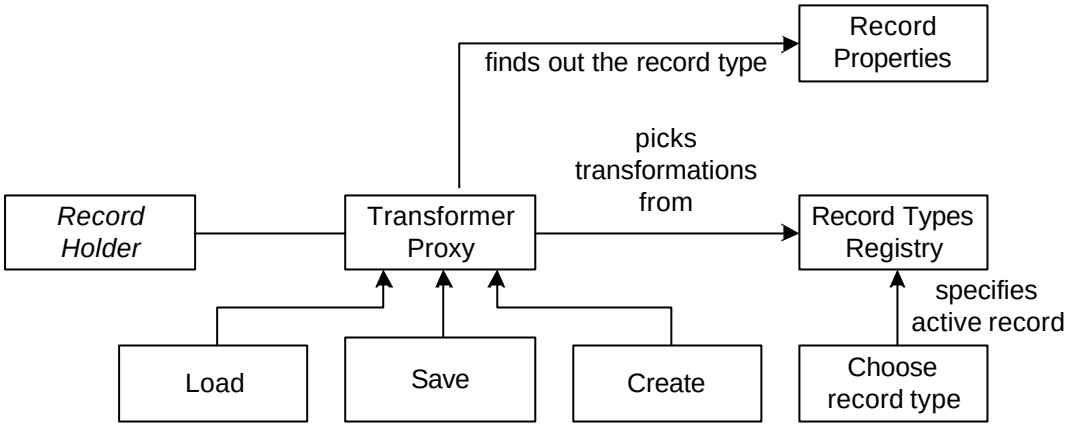
Functions, affected by this feature are “Create metadata record”, “Load metadata record” and “Save metadata record”:

- “Create” function with the multiple record types support is always preceded by another function: “Choose record type”. The latter specifies a type of the record to be created.
- “Load” function. As long as the universal format is used for the metadata storage and record type specific format – for editing, any record that is being loaded from the storage into the Record holder, needs to be transformed. Transformation depends on the record type. Without the multiple record types support the universal format coincides with the format for editing, so no transformation is needed.
- “Save” function. As long as the universal format is used for the metadata storage and record type specific format – for editing, any record that is being saved into the storage, needs to be transformed. Transformation depends on the record type. Without the multiple record types support the universal format coincides with the format for editing, so no transformation is needed.

Obviously we need two transformation components – for transforming record from the universal format to the format for editing and back. However, these two component might have the same implementation, just different configurations. As long as “Create” function produces new empty record instances in one format, say in the universal format, it is to be transformed into the format for editing as well. In order to implement behaviour differences for the functions, mentioned above, a proxy component for transformations can be introduced. This proxy component can mediate, firstly, the communication between “Create” function and the Record Holder, and secondly, between “Load” and “Save” functions and the Metadata Storage. Another component is needed to perform the function “Choose record type”: Record types registry. It stores the transformation for all available record types where one of these transformation may be marked as active. Active transformation affects only “Create” function, specifying the record type to be assigned when a new record is created.

When a metadata record is loaded from the storage or stored into it, it is important to find out the right record type to apply the transformation. Information about the record type can be stored in the Record Properties, so other components do not have to be aware of the record type.

DIAGRAM (CONTROL AND DATA FLOW)



COMPONENTS INVOLVED

- II.3.2. Document Holder (as Transformer Proxy)
- II.3.4. Record Types Registry

FEATURE:	**II.2.10.**	ACCESS CONTROL
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

Restricts access to the metadata store and uploaded data files according to the information in User Profiles. In other words, guards the Metadata Storage and the Upload Manager etc.

FUNCTIONS

- II.1.13. Define the access rights to download or view a data file
- II.1.21. Create, modify, delete user profiles with specific access rights

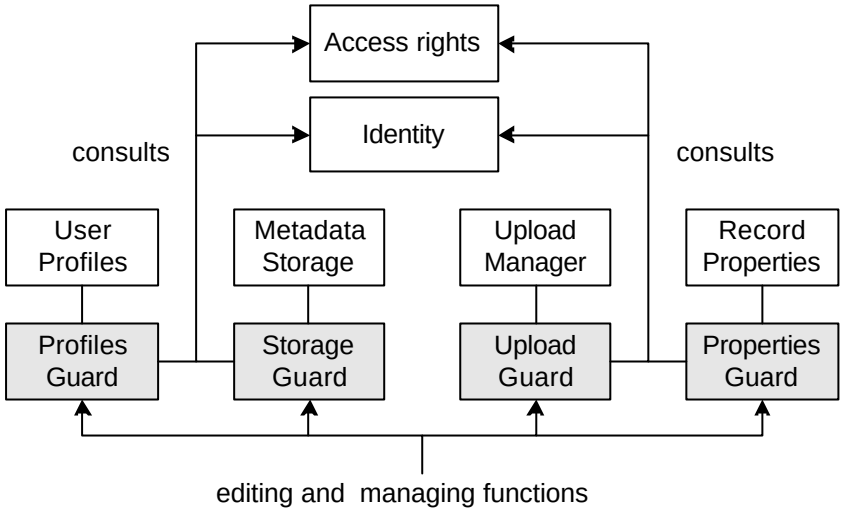
MAIN ASPECTS

- Guarding the components, allowing access to the information, managed by the system (metadata records, uploaded data files)
- Guarding the Record Properties component, since it allows access to the information, crucial for the system
- Guarding the User Profile
- Transparency of the access control mechanism for other components of the system

DECOMPOSITION

In order to perform full-value access control, it is necessary to observe the usage of all components to be guarded as well as component that are capable to change the access control settings. Access to the resources is granted respecting the user rights settings. Each access operation is performed on behalf of certain user. Before the actual performing, the access rights of this user must be checked to find out if he/she has an authorization to perform such operation. Consequently, all inter-component communication that might contain access operations, must be revised by the access control mechanism. Four proxy components are introduced for transparent mediating the communication of Metadata Storage, Upload Manager, Record Properties and User Profiles components correspondingly. These proxies consult other two components to check the necessary permissions. One of them – “Identity” components keeps the information about the user, currently working with the system. The other – “Access Rights” contains policy information for the access control. Here is specified, which right every user has.

DIAGRAM (CONTROL AND DATA FLOW)



COMPONENTS INVOLVED

- II.3.15. Identity
- II.3.16. Access Rights
- II.3.1. Metadata Storage (as Profiles Guard for User Profiles Registry)
- II.3.1. Metadata Storage (as Storage Guard)
- II.3.13. Upload Manager (as Upload Guard)
- II.3.14. Record Properties (as Properties Guard)

FEATURE: **II.2.11.**	FORUM
CHAPTER: **II.2.**	FEATURE SPECIFICATION

DESCRIPTION

The feature groups together functions, concerning the participation in the forums, designated to a specific theme.

FUNCTIONS

- II.1.15. Access one or several for a related to the coastal zone management
- II.1.16. Post new message to a specific forum
- II.1.17. Read previous messages posted to a specific forum
- II.1.20. Create or delete a new forum

MAIN ASPECTS

- Intuitive interface
- Flexibility of the forums

SUGGESTION

Since the forum does not have to contain features specific for Local Information System, it is apparently the best solution to reuse already existing open-source forums. The advantages are:

- Almost no cost for developing the forum. It needs to be installed and set up properly
- New features, appearing in the forums can be introduced quickly

One of such open-source solutions is “phpBB Creating Communities”. Below is the excerpt from the site http://www.phpbb.com/

phpBB is a high powered, fully scalable, and highly customisable open-source bulletin board package. phpBB has a user-friendly interface, simple and straightforward administration panel, and helpful FAQ. It is based on the powerful PHP server language and your choice of MySQL, MS-SQL, PostgreSQL or Access/ODBC database servers.

Key features:

- Supports popular database servers
- Unlimited forums and posts
- Multiple language interface
- Private or public forums
- Powerful search utility
- Private messaging system
- Complete customisation with templates

If employed for the LIS, the forum should be tailored in order to support single sign-on.

COMPONENTS INVOLVED

- II.3.19 Single sign-on forum component

FEATURE:	**II.2.12.**	INTERNATIONALISATION
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

The feature makes possible to render the user interface in different languages.

FUNCTIONS

- II.1.4.1. Specify the language of keyword
- II.1.18. Select the language for the criteria of search, the interface, and the glossary

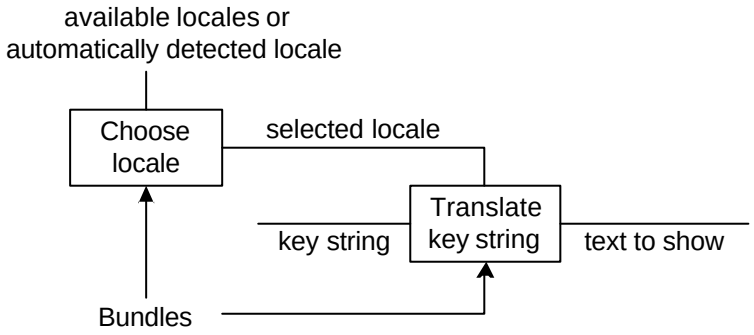
MAIN ASPECTS

- Rendering all user interface information in the selected local language
- Ease of adding new languages and translations

DECOMPOSITION

The usual solution for the internationalization lie in using resource bundles. Resource bundles are capable to translate string keys into strings in the specific language. In order to add new language to the bundles, one needs to create a file with the name, corresponding to the language name or its standard abbreviation. The current language, preferred by the user may be either explicitly chosen or automatically detected from the system or client program settings. The introduced component, Bundles, allows the user to choose the current locale (language), or the system to set automatically detected one. Then this component is used whenever string keys have to be translated into the text in the current language.

DIAGRAM (FUNCTIONAL)



COMPONENTS INVOLVED

- II.3.17. Bundles

FEATURE:	**II.2.13.**	USERS
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

The feature groups together functions, needed for the user management and aims at:

- organising restricted access to the resources, stored in the system
- providing contact information about the users of the system
- realizing inter-user communication in the forums

FUNCTIONS

- II.1.21. Create, modify, delete new users within each user profile

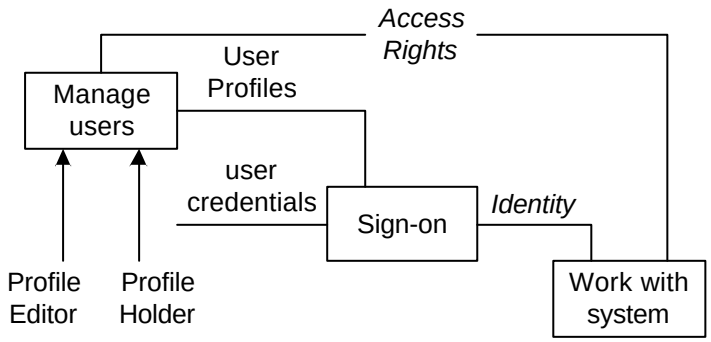
MAIN ASPECTS

- Single sing-on for different applications and components of the LIS through the use of central user management

DECOMPOSITION

Function II.1.21. “Create, modify, delete new users within each user profile” implies the existence of a component, that is used to store user profiles. This component can be implemented in the same way as the Metadata Storage. Similarly to the metadata records, user profiles are pieces of structured or semi-structured information. Like metadata records, user profiles can be edited by the Editor component – Profile Editor. Since the Editor component always needs a Document Holder, Profile Holder should be introduces for this purpose. Having the user profiles information, the system can perform identification and authentication of users. The idea of single sign-on for all parts of the system is implemented in the Identity component. Using the Identity, every component in the system can find out which user is working with the system and obtain detailed information about him/her with the help of User Profiles.

DIAGRAM (FUNCTIONAL)



COMPONENTS INVOLVED

- II.3.1. Metadata Storage (as User Profiles Registry)
- II.3.5. Editor (as Profile Editor)
- II.3.2. Document Holder (as Profile Holder)
- II.3.15. Identity

FEATURE:	**II.2.14.**	TRANSFER
CHAPTER:	**II.2.**	FEATURE SPECIFICATION

DESCRIPTION

The feature represents the possibility to transfer metadata records and associated data (if available) towards the Eurosion server.

FUNCTIONS

- II.1.23. Regular transfer of metadata, and if available data themselves, towards EUROSION server

MAIN ASPECTS

- Interoperability of the central Eurosion server and Local Information Systems

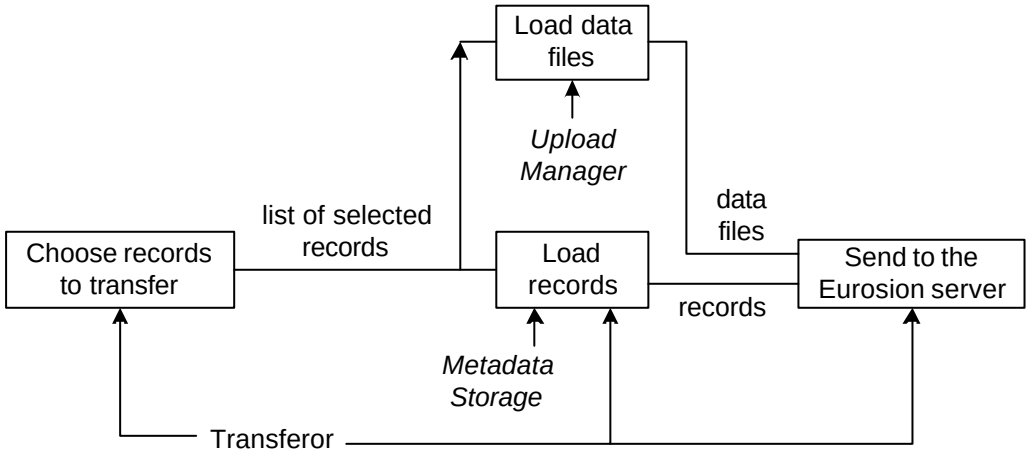
DECOMPOSITION

The operation of the metadata transfer consists of three main steps:

- Choose the records to transfer
- Load selected records from the storage
- Load data files if available
- Upload (send) records and data files to the Eurosion server

Upload operation implies that the Eurosion server has the corresponding functionality. Transferor component acts as an auxiliary addition on top of Metadata Storage and Upload Manager functionality.

DIAGRAM (FUNCTIONAL)



COMPONENTS INVOLVED

- II.3.18. Transferor
- II.3.1. Metadata Storage
- II.3.13. Upload Manager

COMPONENT SPECIFICATION

COMPONENT:	**II.3.1.**	METADATA STORAGE
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates the functionality concerning storing and retrieving metadata records in LIS and User Profiles. It performs mainly functions of ID management, record creation, storing, retrieving and removing, as well as querying of record in the storage. The interface described below serves as a basis for several components: Metadata Storage itself, its proxies (Storage Guard, Record Filter Proxy), filter – Template Filter, component for User Profiles Registry and its access control proxy – Profiles Guard. Storage Guard restricts the access to the resources in the storage, Record Filter Proxy filters only records (not templates) from the storage, whereas Template Filter – only templates.

INTERFACE

```
interface Storage {

  // Creates a new unique identifier in the context of the metadata storage

  string createId();

  // Creates a new empty record with the given id

  Record createRecord(string recordID);

  // Retrieves a record from the storage

  Record getRecord(string recordID);

  // Returns a list of the ids for all records stored in the storage

  sequence string listRecords();

  // Removes the record from the storage

  void removeRecord(string recordID);

  // Stores the provided record into the storage

  void storeRecord(string recordID);

  // Runs a query against the storage and return the list of ids of found records

  sequence string query(string query);

}
```

ROLE

MetadataStorage

This role is assigned to several components: Metadata Storage itself and its proxies: Record Filter Proxy and Storage Guard.

TemplateFilter

The role for the template filter.

UserProfiles

This role is assigned to the User Profiles Registry component and its access control proxy – Profiles Guard.

CONFIGURATION

- Contact information for the database, where the metadata records are stored (only for the Metadata Storage itself)

HUMAN RESOURCES

Metadata Storage (User Profiles Registry)	5 person days
Filter (Record Filter Proxy, Template Filter)	2 person days
Storage Guard (Profiles Guard)	5 person days

COMPONENT:	**II.3.2.**	DOCUMENT HOLDER
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality concerning the temporary storage to keep a metadata record (or a query, or a user profile) during editing. In the interface below, documents are shown as Document type. Document type might be the parent for Record type, used in the Metadata storage interface, or be even equivalent with it.

INTERFACE

```
interface DocumentHolder {

  // Puts a document into the holder.

  void setDocument(Document document);

  // Gets the document that is currently in the holder. Returns no value (NULL) when there is no document in the holder.

  Document getDocument();

  // Removes the document from the storage.

  void removeDocument();

}
```

ROLE

Since Document Holder is used by the metadata editor as well as by the query editor and by user profiles editor, it has three corresponding roles:

RecordHolder

This role is also shared by the proxy component named Record Holder Proxy that serves for multiple record support as well as by the proxy component Transformer Proxy allowing Multiple record types.

QueryHolder

ProfileHolder

CONFIGURATION

This component might have a configuration that, however, dependent on the implementation. For example, for the Web thin client it could contain the name of the session attribute to store the document.

HUMAN RESOURCES

Document Holder (Record Holder, Query Holder, Profile Holder)	3 person days
Record Holder Proxy	2 person days
Transformer Proxy	5 person days

COMPONENT:	**II.3.3.**	RECORD POOL
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality concerning the maintenance of the record pool, allowing simultaneous editing of several records. The pool contains a set of records with assigned ids. One of these records can be marked as active.

INTERFACE

```
interface RecordPool {

  // Puts a new record into the pool.

  void putRecord(string recordId, Record record);

  // Gets the list of all records stored in the pool. Returns a sequence of record ids.

  sequence string getRecords();

  // Retrives a record with given id from the pool.

  Record getRecord(string recordId);

  // Removes the record with given id from the pool.

  void removeRecord(string recordId);

  // Makes one of the records in the pool active.

  void setActiveRecord(string recordId);

  // Gets the active record.

  Record getActiveRecord();

  // Removes the active record from the pool.

  void removeActiveRecord();

}
```

ROLE

RecordPool

CONFIGURATION

This component might have a configuration that, however, dependent on the implementation. For example, for the Web thin client it could contain the name of the session attribute to store the pool.

HUMAN RESOURCES

Record Pool	3 person days

COMPONENT:	**II.3.4.**	RECORD TYPES REGISTRY
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality concerning transformations of the edited metadata records into the unified storage format and back. It contains a map (hash table), where two transformations (to the unified format and back) are assigned to every record type. Transformations are denoted in the interface with a type “Transformation”. Depending on the technologies, used for record transformations, implementations of this type might be references to some external transformation engine, compiled transformation etc.

INTERFACE

```
interface RecordTypesRegistry {
  // Retrives “load” transformation for given record type. “Load” transformation converts a record from the unified storage format into the editing format.
  Transformation getLoadTransformation(string recordType);
  // Retrives “save” transformation for given record type. “Save” transformation converts a record from the editing format into the unified storage format.
  Transformation getSaveTransformation(string recordType);

}
```

ROLE

RecordTypesRegistry

CONFIGURATION

For every record type

- Reference to the transformation from the format, used for editing records of this type towards the unified storage format
- Reference to the transformation from the unified storage format towards the format, used for editing records of this type

HUMAN RESOURCES

Record Types Registry	7 person days

COMPONENT:	**II.3.5.**	EDITOR
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality concerning search query construction, and editing of metadata records and user profiles. It is involved when a user fills the search form to specify his/her desired search criteria, as well as when a user edits a metadata record or manages the user profiles. Search query, metadata records and user profiles are observed as documents that are processed by the editor. A document might have a hierarchical structure, including elements and attributes, where elements usually represent non-terminal nodes and attributes – leaves. Another difference between them is that the elements with the same name may occur within the parent element, whereas the attributes must have unique names within the parent. Every element and attribute in the document can be accessed using path expressions of the following form:

1. PathExpression :- PathStep (“/” PathStep)*

2. PathStep :- (ElementReference | (“@” AttrubuteName))

3. ElementReference :- ElementName [“[“ Number “]”]

Every path expression is a sequence of one or more path steps, delimited by slash. Every path step is either an element reference or an attribute name, preceded with the “@” sign. Element reference comprises the element name and optional ordinal number in brackets that denotes the number of the element in the parent one, starting with “1”.

Current document for editing resides inside the Document Holder (Record Holder or Query Holder) component.

INTERFACE

```
interface Editor {

  // Inserts an element within the specified parent element and after specified sibling. If parent is not specified
(NULL), the element is inserted as a root. If preceding is not specified (NULL), the element is inserted as the first
element within the parent.

  void insertElementAfter(string parent, string preceding, string name);

  // Inserts an element within the specified parent element and before specified sibling. If parent is not specified
(NULL), the element is inserted as a root. If preceding is not specified (NULL), the element is inserted as the last
element within the parent.

  void insertElementBefore(string parent, string following, string name);

  // Removes the specified element.

  void removeElement(string element);

  // Sets the value of the element.

  void setElementValue(string element, string value);

  // Inserts an attribute within the specified parent element and sets its value (or replaces existing one).

  void setAttribute(string attribute, string value);

  // Removes the specified attribute.

  void removeAttribute(string attribute);

}
```

ROLE

QueryEditor

RecordEditor

ProfilesEditor

CONFIGURATION

- Role name of the Document Holder component. Since the editor component might be used in different context (i.e. as a Record Editor, as a Query Editor and as a Profile Editor), it must be properly assigned to the appropriate document holder.

HUMAN RESOURCES

Editor	8 person days

COMPONENT:	**II.3.6.**	QUERY TRANSLATOR
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality needed for transforming queries, constructed with the query editor, into the storage-specific (in other words, understandable by the storage) format or language. Queries to the storage are supposed to be strings and queries produced by the query editor could be picked up from the Query Holder in the form of serialized document.

INTERFACE

```
interface QueryTranslator {

  // Translate the query.

  string translate(string query);

}
```

ROLE

QueryTranslator

CONFIGURATION

This component can have a configuration if represents a generic translator that can work with several kinds of metadata storages. Although this way of implementing the query translator is not encouraged, since it increases the complexity of the query translator component with each new supported metadata storage. The better idea is to provide metadata storages and query translator for them always together, in pairs.

HUMAN RESOURCES

Query Translator	5 person days

COMPONENT:	**II.3.7.**	TRANSLATORS REGISTRY
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component serves as a registry for diverse geographical coordinate transformations. Many coordinate translator components might be registered in the Local Information System, since the users may use different coordinate reference system to specify spatial coverage or other data, containing geographical coordinates. This component acts as a facade and a multiplexing component for several translator components. It might also be able to find chained multi-step transformations, where the direct transformation between two coordinate reference systems is not available.

INTERFACE

```
interface CoordinateTransformer {

  // Facade method for transforming coordinates from one reference system to another. InitRS is an initial
reference system, targetRS – target reference system, coordinate – coordinates to transform.

  sequence double transform(string initRS, string targetRS, sequence double coordinates);

}
```

ROLE

TranslatorsRegistry

CONFIGURATION

- Coordinate reference systems to refer
- Coordinate translators configurations. Every such configuration contains a reference to the implementation of a specific translators (all translators have the same interface).

HUMAN RESOURCES

Translators Registry	3 person days

COMPONENT:	**II.3.8.**	COORDINATE TRANSLATOR
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality concerning transformations of geographical coordinates between coordinate reference systems. There might be arbitrary number of coordinate translators. They all have the same interface, whereas their implementations differ. Each implementation represents the transformation between two coordinate reference systems.

INTERFACE

```
interface CoordinateTranslator {

  // Transforms a sequence of coordinates from one reference system to another.

  sequence double transform(sequence double coordinates) ;

}
```

ROLE

Components of this type do not have roles since they are not managed by the component container (the reason for that the number of such components could be quite large and some other efficient management mechanism rather than general-purpose container should be employed).

CONFIGURATION

Configuration of this type of component is a part of Translators Registry component configuration:

- Initial coordinate reference system
- Target coordinate reference system

HUMAN RESOURCES

Coordinate Translator	10 person days

COMPONENT:	II.3.9.	THESAURUS SOURCE
CHAPTER:	II.3.	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality concerning the access of the hierarchical thesauri, i.e. geographical thesaurus, containing information about administrative regions and their geographic coordinates, and thematic thesaurus, containing information about keywords occurring in metadata records. It provides the means for thesauri traversal and acts as a facade for the thesaurus object model, which may include several objects (like thesaurus, term, group of terms etc.). Along with the hierarchical information, any additional information about the terms can be provided, for example, name of the term. If this information is available in different languages, the Thesaurus Source component has to consult the Locale component to find out how to render all region-dependent values.

INTERFACE

```
interface ThesaurusSource {

  // Gets children objects of the specified one as a list of ids. When no parent is specified (NULL), ids of topmost objects returned.

  sequence string getChildren(string parentId);

  // Gets parent object of the specified one.

  string getParent(string childId);

  // Gets additional information about the object in the thesaurus (name, keyword id etc.). For example, getInfo("12", "name").

  string getInfo(string id, string infoName)

}
```

ROLE

GeographicThesaurusSource

ThematicThesaurusSource

CONFIGURATION

- Access information of the database (or directory, file), where the thesaurus is stored.
- Thesaurus schema, defining additional information to every term.

HUMAN RESOURCES

Thesaurus Source (Geographic Thesaurus Source, Thematic Thesaurus Source)	15 person days

COMPONENT:	**II.3.10.**	THESAURUS BROWSER
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality concerning interactive browsing of the thesaurus. It maintains the current state of the browser and lets the user change it through the interaction. Current state is actually a map (hash table), containing a boolean attribute for every term in the thesaurus. This attribute shows whether the term is expanded in the browser (whether its children terms are visible). In order to simplify the front-end part of the thesaurus browser, the following interface has an operation that generates the logical representation of the thesaurus respecting "expanded" attributes of the terms. This logical representation can then be transformed into viewable form. Call-backs are configured though call-back template. This template can be parameterised with the data, containing in the thesaurus, to form a call-back command.

INTERFACE

```
interface ThesaurusBrowser {

  // Gets the value of expanded attribute of a term.

  boolean isExpanded(string id);

  // Sets the value of expanded attribute of a term.

  void setExpanded(string id, boolean expanded);

  // Generates the logical representation of the thesaurus respecting "expanded" attributes of the terms.

  any getLogicalView();

}
```

ROLE

Since there might be several thesaurus browsers in the system, they should have different roles and be associated with right thesaurus sources. Association with thesaurus sources is specified in the configuration.

GeographicThesaurus

ThematicThesaurus

CONFIGURATION

- Role name of the thesaurus source
- Call back template (including role name of the associated editor)

HUMAN RESOURCES

Thesaurus Browser (Geographic Thesaurus, Thematic Thesaurus)	10 person days

COMPONENT:	**II.3.11.**	MAP VIEWER
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality allowing interactive delimiting a geographical region on the map. It maintains the current state of the map viewer and allows a user to change it during the interaction. The current state of the map viewer comprises the coordinates of currently visible part of the map and coordinates of the bounding box. All actions that the user may perform with the map viewer are manipulating these coordinates. They are presented in the interface as auxiliary operations based on main operations for coordinate manipulations.

INTERFACE

```
interface MapViewer {

  // Gets the coordinates of the entire map.

  double[4] getEntireMap();

  // Gets the coordinates of the currently visible part of the map.

  double[4] getMap();

  // Sets the coordinates of the currently visible part of the map.

  void setMap(double[4] map);

  // Gets the coordinates of the bounding box

  double[4] getBoundingBox();

  // Sets the coordinates of the bounding box

  void setBoundingBox(double[4] boundingBox);

  // Auxiliary operation: zoom in. Makes the part of the map, selected by the bounding box, visible.

  void doZoomIn();

  // Auxiliary operation: zoom out. Increases the scale of the map, moving the currently visible part to the centre
and making adjacent space visible.

  void doZoomOut();

  // Auxiliary operation: move up. Scrolls the map up, making the part of the map above the current visible.

  void doMoveUp();

  // Auxiliary operation: move down. Scrolls the map down, making the part of the map below the current visible.

  void doMoveDown();

  // Auxiliary operation: move right. Scrolls the map to the right, making the part of the map to the right from the
current visible.

  void doMoveRight();

  // Auxiliary operation: move left. Scrolls the map to the left, making the part of the map to the left from the
current visible.

  void doMoveLeft();

  // Other auxiliary operations are possible..

}
```

ROLE

MapViewer

CONFIGURATION

- Map content and boundaries (for example, in the form of references to shape files)
- Call-back template (including role of the associated editor)

HUMAN RESOURCES

Map Viewer	7 person days

COMPONENT:	II.3.12.	MULTIPART DECODER
CHAPTER:	II.3.	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality for decoding multipart messages coming from the internet client over the network and producing normal data files from them. The implementation depends on the protocol and multipart algorithm, used by the client program. In the interface declaration type inputStream is used to depict the multipart encoded content stream and plain file content stream. The Multipart Decoder acts as a wrapper for the multipart streams. The “inputStream” interface might look differently depending on platform, programming language and Input/Output library.

One implementation of the multipart decoder for Java programming language can be found at: http://www.servlets.com/cos/index.html. It is free for use in non-commercial development projects. For a commercial project, permission is granted to use the packages provided that every person on the development team for that project owns a copy of the book *Java Servlet Programming* (O'Reilly) in its most recent edition.

INTERFACE

```
interface MultipartDecoder {

  // Creates a wrapper  for the multipart input stream.

  inputStream decode(inputStream multipart);

}
```

ROLE

MultipartDecoder

CONFIGURATION

- Buffer size for the decoder

HUMAN RESOURCES

Multipart Decoder	3 person days

COMPONENT:	**II.3.13.**	UPLOAD MANAGER
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality concerning the management of uploaded data files. Since these data files might be used in different components of the system, their metadata should be managed centrally. Not only storing of data files but also access to them must be performed via this component, to enable the full-value access control mechanism.

INTERFACE

```
interface UploadManager {

  // Store a data file in the local file system or in a database. File is given as an input stream. Returns the URL of the created file.

  string uploadFile(string filename, inputStream file);

  // Access the data file with the given name.

  inputStream downloadFile(string filename);

}
```

ROLE

UploadManager

This role is also shared by the Upload Guard component that ensures the access control over the uploaded data files.

CONFIGURATION

- Base directory in the local system or connect information for the database where data files must stored
- Template of the URL that is returned by the “uploadFile” operation

HUMAN RESOURCES

Upload Manager	5 person days
Upload Guard	5 person days

COMPONENT:	II.3.14.	RECORD PROPERTIES
CHAPTER:	II.3.	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulate functionality concerning the maintenance of the record properties. These properties may contain additional metadata about the records, used by other components. Such metadata can include:

- Template markers (to distinguish normal records from templates)
- Record ownership (to assign every record to the user that created it)
- Record type markers (to find appropriate transformation to the unified storage format for the record)

It is not encourage to store such information inside the records itself. As such, template markers component implements a two-dimensional map (hash table), where a string attribute is assigned to every record id and property name.

INTERFACE

```
interface RecordProperties {

  // Gets the value of the property for the record with given record id.

  string getProperty(string recorded, string property);

  // Sets the value of the marker for the record with given record id.

  void setProperty(string recordId, string property, string value);

}
```

ROLE

RecordProperties

This role is also shared by the Properties Guard component that is responsible for the access control.

CONFIGURATION

Depending on the implementation, configuration could specify, where the record properties must be stored (file in the file system, database).

HUMAN RESOURCES

Record Properties	3 person days
Properties Guard	3 person days

COMPONENT:	**II.3.15.**	IDENTITY
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component keeps the identity of the user that is currently working with the system. This component is used by the access control mechanism to find out on behalf of which user all operations are performed. This component actually only contains a reference to the corresponding user profile, stored in the User Profiles Registry.

INTERFACE

```
interface Identity {

  // Retrieves the identity of current user in the form of his/her id. Detailed information about this user can be then obtained with the help of User Profiles Registry.

  string getCurrentUserId();

}
```

ROLE

Identity

CONFIGURATION

Depending on the implementation and on the type of application, configuration can contains, for example, advice for where the identity should be stored.

HUMAN RESOURCES

Identity (as well as authentication mechanism)	5 person days

COMPONENT:	II.3.16.	ACCESS RIGHTS
CHAPTER:	II.3.	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality concerning the management of access rights of users towards records, created by other users, data files, uploaded by other users and other relevant information. Access rights can be represented as a matrix, where each cell contains the permissions of one user towards the resources of another one. Every permission has a name and all access control components are aware of permission names they are sensitive to. Access Rights component is implicitly connected with the Identity component, so whenever access right view or change is requested, the acting user is checked against the matrix.

INTERFACE

```
interface AccessRights {

  // Checks if the first user has permission with given name towards the second one.

  boolean isPermitted(string userId1, string userId2, string permission);

  // Set the permission with given name for the first user towards the second one.

  void setPermission(string userId1, string userId2, string permission, Boolean value);

}
```

ROLE

AccessRights

CONFIGURATION

Depending on the implementation, configuration might contain general policy settings, applied to the access control mechanism.

HUMAN RESOURCES

Access Rights	3 person days

COMPONENT:	**II.3.17.**	BUNDLES
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality concerning multi-lingual user interface for the Local Information System. It keeps the current selected locale and translates key string into text strings in the current language. Translations of key strings are stored separately for every language in the form of file (resource bundles), so they can be easily managed.

INTERFACE

```
interface Bundles {

  // Sets the current locale.

  void setLocale(string locale);

  // Finds out the current locale.

  string getLocale();

  // Translates a key string according to the current language.

  string translate(string key);

}
```

ROLE

Bundles

CONFIGURATION

For every locale:

- Reference to the translation file.

Alternative configuration might contain only the path with translation files. In this case, file names are must be chosen with respect to corresponding locales.

HUMAN RESOURCES

Bundles	4 person days

COMPONENT:	**II.3.18.**	TRANSFEROR
CHAPTER:	**II.3.**	COMPONENT SPECIFICATION

DESCRIPTION

The component encapsulates functionality needed for performing regular data transfer from the Local Information System towards the Eurosion server. Operations, related to the transfer, performed mainly by Metadata Storage, Upload Manager components and Eurosion server itself (in the way that it provides an open port for the incoming transfer and accepts data from Local Systems). Transferor component contains a set of auxiliary operations that help to simplify user interface and other development related to the Transfer feature. Since transfer operations are normally long-lasting, it would be a good idea to call the operation in the interface below in the asynchronous mode.

INTERFACE

```
interface Transferor {

  // Loads records and corresponding data from Metadata Storage and Upload Manager and send it to the
Eurosion server. The parameter is a sequence of record ids, chosen for the transfer.

  void transfer(sequence string recordIds);

}
```

ROLE

Transferor

CONFIGURATION

- Reference (for example in the form of URL) to the port on the Eurosion server for incoming transfer

HUMAN RESOURCES

Transferor	5 person days

COMPONENT:	II.3.19.	SINGLE SIGN-ON FORUM COMPONENT
CHAPTER:	II.3.	COMPONENT SPECIFICATION

DESCRIPTION

The component serves for enabling single sign-on for the thematic forum and other parts of the Local Information System. When a user enters the system, he/she provide his/her credentials (user name, password) for the sake of identification and authentication. Once having logged in, the user should not have to make it again. If the thematic forum is implemented upon the basis of an existing forum solution, the support of single sign-on must be extra considered.

The interface and configuration depend on the forum system that is chosen for the Local Information System and should be determined during the translation from the generic to the specific guidelines.

HUMAN RESOURCES

Single sign-on forum component	5 person days

CORPORATE DESIGN DEFINITION

When talking about the appearance of the Local Information System, the following objectives of the Graphical User Interface must be taken into account:

- Interface needs to be logically clear and understandable. In other words the action flow of the interface should be predictable. This goal is archived through the well-designed logical structure of the interface
- Interface should be optimised for the type of work it is created for. This goal also concerns the action flow and can be archived during the fine-tuning of the system, when the feedback from the users is processed
- Screen should be used as optimal as possible. This goal is archived by using view and perspective paradigms
- Cognitive aspects of the user interaction should be respected. The content within one view or perspective should be organized in the way that the most important things would be clearly visible and accessible. This goal is archived by the proper usage of interface design patterns and guidelines.
- Ergonomic aspects of the user interaction must be taken into account. The colours and their combinations, text sizes and styles must be as little annoying and irritating as possible. This goal is archived by using styles, including text and background colours, fonts etc.

Consequently, the main component of the user interface development are:

- Views and perspectives
- Action flow (navigation)
- Styles
- User Interface Design Patterns and Guidelines

Some of these component are generic whereas the others need to be defined for every Local Information System specifically. The following table groups the component mentioned into 2 categories: generic and specific.

Generic	**Specific**
Views	Perspectives
Action flow	Styles
User Interface Design Patterns and Guidelines	

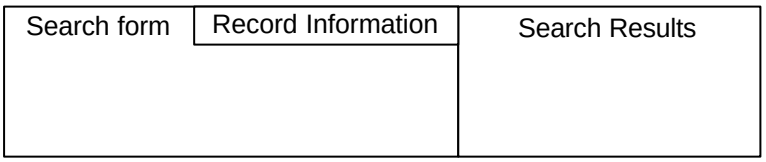
Views and perspectives

Views provide alternative presentation of or ways to navigate the information in the System. View usually contains one structural unit (list, tree etc.) that includes elements of the same or similar type (controls, text pieces etc.) or semantics. A view might appear by itself or stacked with other views. Perspective is a way to compose views together. Perspective determine the layout of the user's screen. According to the set of functions, defined for the Local Information System, the following views can be defined:

- Introduction (splash)
- Main menu
- Search form
- Map Viewer

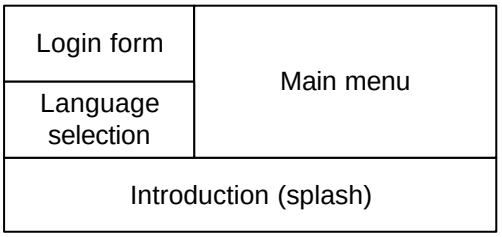
- Thesaurus Browser
- Glossary
- Search Results
- Record Information
- External file viewer (for downloaded data files)
- Metadata record editor
- List of forums
- List of topics in a specific forum
- List of posts in a specific topic
- New post
- Content of the metadata storage
- Language selection
- Login form
- Access rights management
- User profile
- Transfer metadata view
- Help

Views should be organized into perspectives when the specific Local Information System is designed. Some examples of perspectives are given below (these examples are schematic, so they do not advice size proportions of the particular views):



Search perspective

The Search perspective consists of 3 views, two of them (Search Form and Record Information) are stacked on the left side of the screen and the other (Search Results) is placed on the right side.



Welcome perspective

The Welcome perspective consists of 4 views, two of them (Login form and Language selection) place on the top left corner of the screen, since they are most important for the user that has just started to work with the system.

For the specific Local Information System, the set of perspectives and their content should be defined as show above.

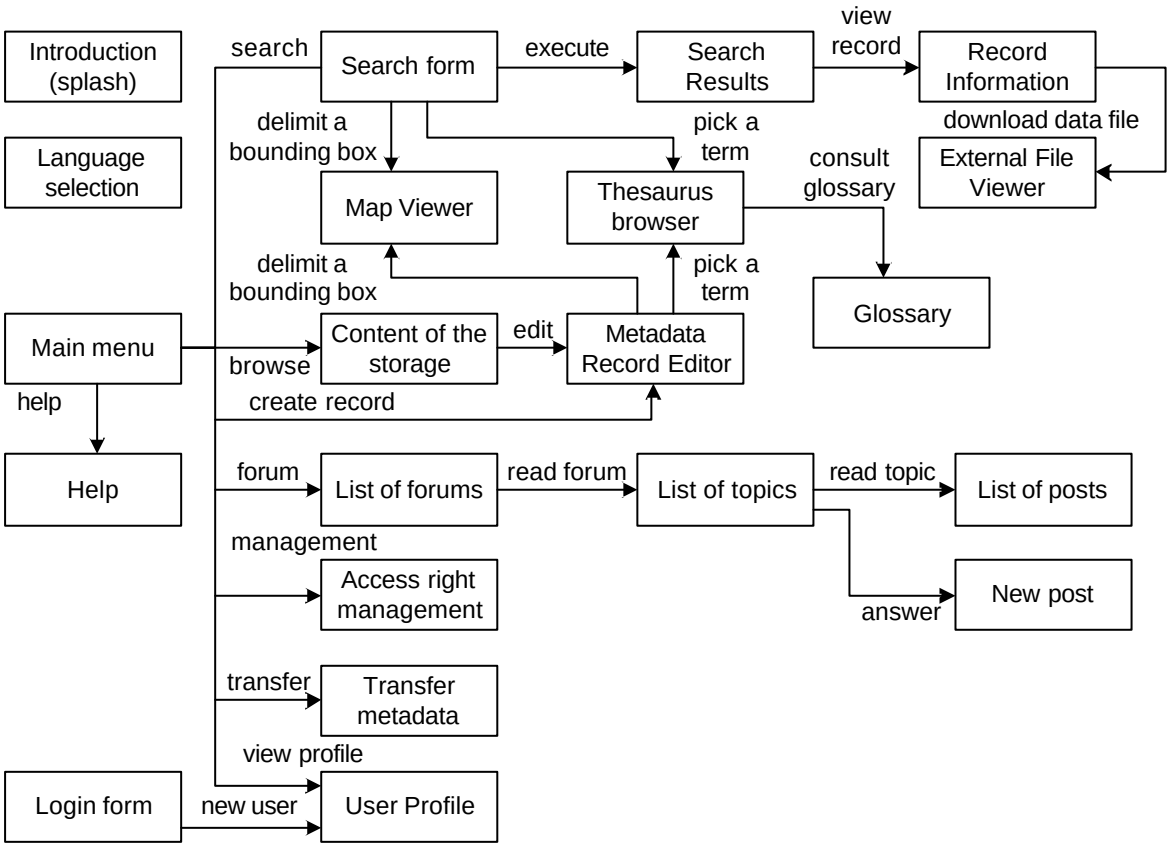
Action flow (navigation)

Action flow determines how the user can navigate within the system to perform the work. Action flows affects the usability of the user interface significantly and therefore, must be carefully designed. Navigational capabilities of the system can be modelled using the transition diagram, which shows,

how the user can move from one view to another, performing actions. On this diagram, boxes represent views and arrows represent action that the user can perform on these views. Leftmost boxed depict the views that will apparently appear on the first, Welcome perspective. Actions that are performed on the main menu, cause the perspective change. The following perspectives can be considered:

- Search
- Browse
- Forum
- Management
- Transfer
- Profile

These perspective might contain actions, causing other perspectives (for example, Browse perspective contains an action that allows to edit a metadata record, therefore, Edit perspective follows) or encompass all necessary views in themselves.



Styles

Styles comprise of several characteristic properties, such as:

- Text and background colours
- Text fonts
- Images
- Table styles

In order to keep the user interface flexible and maintainable, it is recommended to use logical style instead of associating certain physical style (e.g. colour, font) with widgets. When logical styles are associated with widgets, the external mapping of logical styles to physical styles should be created. Changing this mapping, one can easily tune the user interface appearance.

User Interface Design Patterns and Guidelines

User Interface Design Patterns and Guidelines must be follow to leverage the diverse experience concerning the User Interface Design, that is publicly available.

USER MANUAL CREATION

FUNCTION:	**II.5.1.**	DEFINE GEOGRAPHIC CRITERIA OF SEARCH BY TYPING A GEOGRAPHICAL LOCATION
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Defining geographic criteria of search by typing a geographical location

DESCRIPTION

This function can be used to directly enter a geographical location. There are four input fields named west, east, south, and north, all of which have to be filled out in order to specify a bounding box of the location which the user wants to select. The units for the entries are longitude and latitude.

To delete the selected geographical location, the input fields can be simply cleared out or the clear button can be used.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the number and identifier of the input boxes according to the implementation
- Change the reference description according to the implementation
- Change delete mechanism according to the implementation

FUNCTION

- II.1.1. Define geographic criteria of search by typing a geographical location

SUBFUNCTION:	**II.5.1.1**	SPECIFY REFERENCE SYSTEM
FUNCTION:	**II.5.1**	DEFINE GEOGRAPHIC CRITERIA OF SEARCH BY TYPING A GEOGRAPHICAL LOCATION

TITLE

- Specifying reference system

DESCRIPTION

This function can be used to change the reference system for entering geographical locations. If there is no reference system selected all values are in longitude and latitude. If another reference system is desired it can be selected from the list of all supported reference systems.

ADAPTIONS

- Provide a screen shot of the implementation
- List and describe all supported reference systems
- Change the description of the default reference system according to the implementation

FUNCTION

- II.1.1.1 Specify reference system

FUNCTION:	**II.5.2.**	DEFINE GEOGRAPHIC CRITERIA OF SEARCH BY SELECTING A LOCATION IN A GEOGRAPHICAL THESAURUS
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Defining geographic criteria of search by selecting a location in a geographical thesaurus

DESCRIPTION

Instead of typing in the coordinates of a geographical location, the system provides an easy way for selecting well known geographical locations, e.g. countries and parts of them. All well known geographical locations are organized in a thesaurus and represented by a tree-like structure.

The desired geographical location can be picked from this tree. If a location consists of further parts, in this case there is a “+” left beside the identifier of the location. By clicking on “+” the parts are made visible in the tree and the “+” changes to a “-“. The “-“ can be used to hide the parts of a displayed geographical location to provide more room for the identifier of other locations. A click on the identifier of a geographical location itself selects it as search criteria and displays it in the search form.

In the search form, the delete button behind the displayed identifier of the geographical location can be used to delete the selected region.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation
- Change the description of the default reference system according to the implementation
- Change delete mechanism according to the implementation

FUNCTION

- II.1.2. Define geographic criteria of search by selecting a location in a geographical thesaurus

FUNCTION:	**II.5.3.**	DEFINE GEOGRAPHIC CRITERIA OF SEARCH BY DELIMITING AN AREA ON AN INTERACTIVE MAP
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Defining geographic criteria of search by delimiting an area on an interactive map delimiting

DESCRIPTION

The most flexible way to select a geographical location is to select it from a map. The system provides an integrated map server for this task. The actual part of the map is presented as a graphical view on the screen and it can be scrolled in any direction by clicking on the border of the map.

By clicking on the magnifying glass with the "+" sign the map can zoomed in and by using the magnifying glass with the "-" sign with map can zoomed out. To centre the map on a specific point from the map the hand icon can be used.

To specify the geographical location the rectangle icon is used. After clicking on it first the upper left corner of the bounding box then second the lower right corner of the bounding box can specified. The coordinates of the selected bounding box are then used by the system as search criteria and displayed in the search form.

To delete the bounding box of the geographical location as a search criteria the delete button behind the displayed coordinates can be used.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation
- Provide some additional information on the limitations of the map server such as resolution, coordinate reference system, covered area etc.
- Change delete mechanism according to the implementation

FUNCTION

- II.1.3. Define geographic criteria of search by delimiting an area on an interactive map delimiting

FUNCTION:	**II.5.4.**	DEFINE THEMATIC CRITERIA OF SEARCH BY TYPING FREE KEYWORDS
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Defining thematic criteria of search by typing free keywords

DESCRIPTION

It is possible to search the metadata records using free textual keywords. This is useful if the required controlled keywords (e.g. from a thesaurus) are not available or the metadata records are not indexed by controlled keywords.

There exists an input field in which all free keywords can be entered separated by spaces. The entered keywords are then compared with the content of the titles and abstracts of the metadata records.

The deletion of the free keywords as search criteria is carried out by simple clearing of the input field.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the descriptions according to the limitations of the free keywords implementation such as number of keywords, the metadata fields which are to be compared with the free keywords etc.
- Change delete mechanism according to the implementation

FUNCTION

- II.1.4. Define thematic criteria of search by typing free keywords

FUNCTION:	**II.5.5.**	DEFINE THEMATIC CRITERIA OF SEARCH BY SELECTING KEYWORDS IN A THEMATIC THESAURUS
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Defining thematic criteria of search by selecting keywords in a thematic thesaurus

DESCRIPTION

This function allows searching the metadata records through keywords, which are controlled by an underlying thesaurus. This is a very powerful tool enabling to effectively search for relevant information because the terms in the thesaurus make up a “broader-narrower” relationship. Therefore it is possible to find metadata records indexed by a more specific keyword by selecting a more global one.

The use of a thesaurus also considerably enhances the chance to find relevant information because of the limited number of terms, which unlike to the free text keywords, are known to the system users. All thesaurus terms also have a unique language independent identifier, which makes it possible to search the metadata records in the default language of the user to find results in other languages as well.

The only disadvantage of the controlled search terms is that the metadata records HAVE to be indexed by them otherwise they are useless, which means some overhead in creating the metadata records.

All terms in the thesaurus are organized in a tree-like structure and are displayed in this tree form on the screen. The desired keyword can be picked from this tree. If a term consists of other parts, there is a “+” left beside the identifier of the term. By clicking on “+” the narrower terms are made visible in the tree and the “+” changes to a “-“. The “-“ can be used to hide the narrower terms from a displayed keyword to provide more room for the identifier of other terms. A click on the identifier of a term itself selects it as search criteria and displays it in the search form.

In the search form the delete button behind the displayed identifier of a selected keyword can be used to delete it.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface the according to the implementation
- Change delete mechanism according to the implementation

FUNCTION

- II.1.5. Define thematic criteria of search by selecting keywords in a thematic thesaurus

FUNCTION:	**II.5.6.**	CONSULT THE DEFINITION OF A SPECIFIC TERM VIA AN INTERACTIVE GLOSSARY
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Consulting the definition of a specific term via an interactive glossary

DESCRIPTION

All terms which are organized in a thesaurus and which are displayed in a tree-like structure are attached to a glossary which describes these terms further. To activate this function it is only necessary to move to pointer with the mouse over the interesting term and to wait for a few seconds. Then the definition of this term is displayed on the screen besides the term itself. To hide the definition of the term after it is displayed the pointer has to be moved to a place with no terms underneath. Some definitions of the terms have further terms inside which are hyper linked to their definition. This definition can be displayed by clicking on the term. Therefore it is possible to browse the glossary from term definition to term definition.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface the according to the implementation
- If hyperlinks are not implemented strike out the description of browsing the term definitions

FUNCTION

- II.1.6. Consult the definition of a specific term via an interactive glossary

FUNCTION:	II.5.7.	FIND OUT THE LIS DATA REPOSITORY WHICH DATA SOURCES ARE MATCHING THE SELECTED CRITERIA
CHAPTER:	II.5.	USER MANUAL CREATION

TITLE

- Using the search form to find metadata records

DESCRIPTION

Searching in the system repository for relevant metadata records that match certain specific criteria is one of the main objectives and functionalities of the system. The search criteria can be entered in the search form and then executed by clicking on the search button.
The search form includes sections to specify what, where, when and who is addressed by the metadata records. The input fields can be simple text fields (e.g. for defining free keywords) or complex tree-like structures (e.g. for defining controlled keywords when using a thesaurus).
The use of the different input fields is described under their own function description.
Additionally fully or partly filled out search form can be stored for later use by clicking on the store button. Each user can have his/her own stored search forms independently. The delete button deletes an already stored search form and the new search button clears only the search form to begin a new search. To load a stored search form, the user has to change to the user profile and select the stored search form there.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface the according to the implementation
- Remove the storing of a search form if not implemented by the LIS

FUNCTION

- II.1.7. Find out the LIS data repository which data sources are matching the selected criteria

FUNCTION:	**II.5.8.**	SELECT A DATA SOURCE IN A LIST OF ITEMS MATCHING THE SELECTED CRITERIA AND VIEW ITS RELATED METADATA
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Viewing the details of found metadata records

DESCRIPTION

After the execution of a specified search all matching metadata records are displayed as a list on the screen. In this list every found metadata record is listed by its title. Now it is possible to browse among this list of metadata records and show their full content. For this purpose the user has to click on the title of a metadata record in the list, then the content of this selected metadata record is shown ion a separate part (window) of the windows. The result list of all found metadata records is still visible, so the user can select another interesting metadata record at will. In this case the details of the prior selected metadata record are replaced by the content of the new one.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface accordingly to the implementation

FUNCTION

- II.1.8. Select a data source in a list of items matching the selected criteria and view its related metadata

FUNCTION:	**II.5.9.**	IF AVAILABLE, DOWNLOAD THE DATA
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Downloading the data linked to the metadata record

DESCRIPTION

Each metadata record describes real data and is associated with a real data object. If the data object is available for the system (e.g. it is stored in the system repository or a valid URL is provided) and the user has the appropriate access rights to retrieve it then it is possible to get the data by clicking the download button following the title of the metadata record in the result list. The data object is stored in the local file system.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface accordingly to the implementation

FUNCTION

- II.1.9. If available, download the data

FUNCTION:	**II.5.10.**	IF AVAILABLE, VIEW THE DATA ITSELF WITH AN APPROPRIATE VIEWER (WORD, EXCEL, ETC.)
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Viewing the data linked to the metadata record

DESCRIPTION

Each metadata record describes real data and is associated with a real data object. If the data object is available for the system (e.g. it is stored in the system repository or a valid URL is provided) and the user has the appropriate access rights to retrieve it then it is possible to view the data by clicking the download button following the title of the metadata record in the result list.

The data object is then automatically viewed by the right viewer (depending on the content type, e.g. word for a DOC file) if the viewer is present by the client operating system. If no appropriate viewer is present then the user has the choice to download the data object and store it in the local file system.

If the data object should be downloaded in any case ignoring its content type and presence of the right viewer the download button can be clicked on with the right mouse button instead of the left one. Now a menu opens and the user can select the function to store the data object. If you have a system with only one mouse button, please refer to the user manual of your client software for how to store the content connected to a hyper link.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface accordingly to the implementation

FUNCTION

- II.1.10. If available, view the data itself with an appropriate viewer (word, excel, etc.)

FUNCTION:	**II.5.11.**	EDIT AND SAVE NEW METADATA RECORDS
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Creating and modifying metadata records

DESCRIPTION

The system provides a powerful editor for creating new metadata records or to modify existing ones. The editor has functions to create new metadata records from scratch, choosing the type of the described data records, create new metadata records from templates, load existing metadata records into the editor, edit the content of metadata records, save metadata records, save metadata records as templates, close metadata records in the editor view and choose the described data records.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation

FUNCTION

- II.1.11. Edit and save new metadata records

FUNCTION:	**II.5.12.**	ATTACH TO THE RECORD A SET OF KEYWORDS FROM A THESAURUS
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Creating and modifying metadata records

DESCRIPTION

In the metadata editor a thesaurus can be used for indexing. The keyword thesaurus is organized in a tree-like structure and allows browsing and selecting the keywords for a metadata record. The desired keywords can then be picked from this tree.
If a keyword has further narrowing terms then there is a “+” left beside the identifier of the keyword. By clicking on “+” this terms are made visible in the tree and the “+” changes to a “-“. The “-“ then can be used to hide the narrowing terms again to get more room for other keywords. A click on a term itself selects it as a indexing keyword and displays it on list in the metadata record.
In the list of the already selected keywords the delete button behind a displayed identifier of a keyword can be used to delete it from the list.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation
- Change delete mechanism according to the implementation

FUNCTION

- II.1.12. Attach to the record a set of keywords from a thesaurus

FUNCTION:	**II.5.13.**	DEFINE THE ACCES RIGHTS TO DOWNLOAD OR VIEW A DATA FILE
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Access rights for downloading or viewing data objects

DESCRIPTION

The system supports users, roles and rights to control the access of data objects linked to metadata records. Every metadata record is linked to a data object. The creator of the metadata record normally supplies the data object to which it is linked. Therefore the metadata creator is seen as the owner of the data object and as a default the owner of a data object can always access it. There exist some special roles like administrator and manager. The system administrator has access to all data objects in the system, where the manager has access to all objects, except of the one where the owner is the administrator.
The administrator can modify the access rights for all other users of the system.

ADAPTIONS

- Change default roles and rights according to the implementation
- Provide a table with all default rights and roles

FUNCTION

- II.1.13. Define the access rights to download or view a data file

FUNCTION:	**II.5.14.**	ATTACH TO THE RECORD A DATA FILE
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Attaching a data object to the metadata record

DESCRIPTION

The metadata editor allows attaching a data object to a metadata record which is currently modified. To do this the browse button can be used. After clicking on it a standard file open choose dialog appears on the screen and it is possible to browse thru the local file system. If the appropriate file is found it can be selected and attached to the metadata record by clicking the open button.
To delete the link between the metadata record and the data object simply the input field data file has to be cleared out.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation
- Change delete mechanism according to the implementation

FUNCTION

- II.1.14. Attach to the record a data file

FUNCTION:	**II.5.15.**	ACCESS ONE OR SEVERAL FORA RELATED TO COASTAL ZONE MANAGEMENT
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Accessing forums

DESCRIPTION

The System provides access to several forums related to coastal zone management, so the user can see the discussions at these forums. On the topmost level the list of all available forums is shown. The following information about a forum might be shown in the list:

- Name of the forum. Gives an idea about the content
- Number of topics (threads) in the forum
- Number of messages posted to the forum. Can server as an indicator of forum's popularity
- Last message posted to the forum. Shows the presence of "live" topics in the forum

When one of the forums is selected, the list of topics in this forum is shown. Now the following information is displayed on the screen:

- Name of the topic (thread). Usually contains a question
- Number of answers. Shows the activity of the users answering the question of the topic or giving their comments
- Number of views. Shows how many users have read the topic.

From this view it is possible to select interesting topics and read their content.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation

FUNCTION

- II.1.15. Access one or several fora related to coastal zone management

FUNCTION:	**II.5.16.**	POST NEW MESSAGE TO A SPECIFIC FORUM
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Posting new messages to a specific forum

DESCRIPTION

After selecting a forum the system provides a new button for posting new messages to this forum. When the button post new message is clicked on a n input form is shown consisting of the input fields is shown. The first input field is for the title of the topic, the second input field is for the message according to the topic. A last click on the post button submits the new message to the forum. Because of the fact that the user is already logged in it is not necessary to enter the user name or role for the new message.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation

FUNCTION

- II.1.16. Post new message to a specific forum

FUNCTION:	**II.5.17.**	READ PREVIOUS MESSAGES POSTED TO A SPECIFIC FORUM
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Reading previously posted messages

DESCRIPTION

The forums are organized in the form of topics and topics can contain messages. After a topic of a specific forum was selected and displayed it is possible to read the messages according to this topic. The messages contain the user which has created the message, the date and timestamp of the message and the message itself.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation

FUNCTION

- II.1.17. Read previous messages posted to a specific forum

FUNCTION:	**II.5.18.**	SELECT THE LANGUAGE FOR THE CRITERIA OF SEARCH, THE INTERFACE, AND THE GLOSSARY
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Selecting the language for the user interface

DESCRIPTION

The language for the user interface can be selected by the user. Possible languages are English, German and French. The selected language will be used by all system components of the user interface including defining search criteria, glossary, metadata editing etc.
To select a language it is sufficient to click on the flag of the according nationality.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the list of possible languages according to the implementation
- Provide a table with the descriptions and flag of all possible languages

FUNCTION

- II.1.18. Select the language for the criteria of search, the interface, and the glossary

FUNCTION:	**II.5.19.**	MODIFY OR DELETE EXISTING METADATA RECORDS
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Modifying or deleting existing metadata records

DESCRIPTION

Existing metadata records can be loaded into the metadata editor in order to modify its content or it can completely deleted. Before modifying or deleting a metadata record first a search for the desired records is necessary. In the result list of the found metadata records there are two buttons shown. The first button called edit loads the metadata record into the editor for fulfilling the modifications, the second button called delete is foe deleting the metadata record once and for all.

The access rights of the user is check so that no user can modify or delete metadata record for which he has no rights. If a metadata record is deleted the according data object is deleted as well, if it is stored in the system repository.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation

FUNCTION

- II.1.19. Modify or delete existing metadata records

FUNCTION:	**II.5.20.**	CREATE OR DELETE A NEW FORUM
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Creating or deleting a new forum

DESCRIPTION

When the list of all forums is displayed on the screen it is possible to create new forums or the delete existing ones. To delete an existing forum simply click on the delete button following the title of the forum. To create new forums select the new forum button on top of the list. The creation of new topics and deletion of exisiting topics are analogous to the forums, but only available when the topics of a specific forum are displayed.
To create or delete forums or topics the user has to have the necessary rights which will be checked by the system.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation

FUNCTION

- II.1.20. Create or delete a new forum

FUNCTION:	**II.5.21.**	CREATE, MODIFY, DELETE USER PROFILES WITH SPECIFIC ACCESS RIGHTS
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Managing user access rights

DESCRIPTION

The access control of the system allows easy managing of the access rights a user of the system has. There are four rights available:

- Manage
- Write
- Read
- Nothing

Manage means the right to read and write metadata records and to mange the right of other users. Write means the user can change the contents of the metadata records and read them; read means reading of the metadata records but not modifying them and nothing disallows all operations.

The rights can be set for the relation between every user and every owner of metadata records independently. To do this a table with the owner roles of metadata records as rows and the rights of the user roles as columns is displayed after invoking the access component. At every intersection of this table one of the rights can be selected.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation
- Change rights and roles according to the implementation

FUNCTION

- II.1.21. Create, modify, delete user profiles with specific access rights

FUNCTION:	**II.5.22.**	CREATE, MODIFY, DELETE, NEW USERS WITHIN EACH USER PROFILE
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Managing user profiles

DESCRIPTION

Each user can update his personal user profile for them self. After invoking the user profile a form is shown by the system with input fields for the personal data like first name, last name, street address, city email etc.
After modifying the personal data it can stored by click the button submit.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation

FUNCTION

- II.1.22. Create, modify, delete, new users within each user profile

FUNCTION:	**II.5.23.**	REGULAR TRANSFER OF METADATA, AND IF AVAILABLE DATA THEMSELVES, TOWARDS EUROSION SERVER
CHAPTER:	**II.5.**	USER MANUAL CREATION

TITLE

- Transferring metadata records and data objects to EUrosion

DESCRIPTION

This function allows transferring metadata records and linked data form the system to the EUROSION server. First a search for the desired metadata records is necessary, and then all metadata records which should transferred to EUrosion can be checked.
Submitting the operation transfers all selected metadata records and all linked data objects to the EUrosion server. To use this function the appropriate rights are necessary.

ADAPTIONS

- Provide a screen shot of the implementation
- Change the description of the user interface according to the implementation

FUNCTION

- II.1.23. Regular transfer of metadata, and if available data themselves, towards eurosion server

MANUAL OF MAINTENANCE

MANUAL OF MAINTENANCE FOR THE LOCAL INFORMATION SYSTEM ENCOMPASSES ISSUES CONCERNED WITH:

- Equipment maintenance
- System maintenance
- Components maintenance

Equipment maintenance

In order to define the content of the hardware maintenance manual, it is necessary to name the hardware components, occurring in the Local Information System implementation:

- Server computer

    a) CPU
    b) Main memory
    c) Hard drive

- Network

    a) Bandwidth
    b) Protocol

- Operating system

- Runtime environment

- Application server

- Database

Specific equipment maintenance manual is to be excerpted from the corresponding maintenance manuals of the individual parts of the equipment.

System maintenance

System maintenance lies mainly in the management of the container and components. Since the system settings is amassed in the container's configuration, the management comprises the following steps:

- When a new component is to be added:

1. Its configuration has to be included into the container's configuration to let the container recognize the component
2. If the component to add has the same role as other component(s) in the container, the proper order of their configurations within the container's configuration has to be respected
3. The implementation of the component has to be included into the runtime library of the system
4. The system has to be restarted

- When the existing component is to be removed:

1. The configuration of the component has to be excluded from the container's configuration
2. The implementation of the component has to be excluded from the runtime library of the system
3. The system has to be restarted

If any external applications (like forum system) are used in the Local Information System, there should exist some bridge (joint) components that enable integration. These bridge component are application specific and therefore must be well document when included into the system. The documentation should enlist following main issues:

- Version of the external application that is used as a part of the Local Information System

- Changes that have been made in the application (if any) for the sake of integration

- How has the external application been configured when integrated into the LIS

- Interfaces and main implementation details of the bridge component

Components maintenance

For every component implementation, following details are included into the component maintenance manual:

- The underlying technologies employed for the implementation
- Conceptual description of the algorithm
- General conditions of the usage
- Compatibility with other components

The system, where the roles of individual components are strictly separated and described, can be understood and therefore, maintained, quite well without further details. However, it is not always the case when components can be made independent from each other. They often rely on some assumption about the implementation of the other component, despite of paradigm forbiddance. When this happens, such assumptions have to be documented and provided along the component maintenance manual.

Logging

Logging is inserting log statements into the code of components. Logging is usually used for debugging purposed, however, it suites well for the system monitoring and maintenance also. Although log statements are scattered in the source code, their management must be performed centrally. Logging equips the developer with detailed context for component failures. On the other hand, testing provides quality assurance and confidence in the application. Logging and testing should not be confused. They are complementary. When logging is wisely used, it can prove to be an essential tool.

Every log statement has normally a severity level. Severity level might be one of following:

- Debugging. Level designates fine-grained informational events that are most useful to debug a system
- Informative. Level designates informational messages that highlight the progress of the system at coarse-grained level
- Warning. Level designates potentially harmful situations
- Error. Level designates error events that might still allow the system to continue running
- Fatal. Level designates very severe error events that will presumably lead the system to abort.

Analysing the log files, one can diagnose system malfunctions and locate their origins. For this sake, the log file have to be well structured. Logs may be organized into the hierarchy that might, for example, match the class hierarchy of the components and their individual parts.

TIMEFRAME AND BUDGET ESTIMATION

Human resources that are needed for the implementation of a Local Information System can be estimated on the basis of functional and component specifications. Every function in the functional specification has a an estimation of human resources, needed for the front-end implementing for this function. Similarly, for every component in the component specification a human resources estimation is given. Both estimations for the functions and for the component can be considered as credible only provided that the following general conditions are satisfied:

- A mature component technology (Apache Avalon, EJB etc.) is used on the place of the abstract component architecture, mentioned in this specification
- Component implementation are reused. If two components can be implemented in one software module (have the same value in the “Implementation” column in the Component Synopsis, Annex 2), they must be implemented so
- Recommendation about the reuse of existing solutions are followed (as in the case of Forum and Multipart Decoder)
- A mature presentation technology (HTML Browsers, Java Swing etc.) is used for the front-end implementation

The numbers in person days, presented in the functional specification and the component specification are summarized in Annex 2 “Human Resource Estimation”. Relations between the function and the features, as well as between the features and the components are depicted in the table. The procedure of the human resource estimation is as follows:

1. Choose the set functions, relevant to the Local Information System that is to be implemented –(according to the task I.1.4.1 “Comparison between the requirements, according to activities and the Technical specifications for prototyping the Internet-based applications”)
2. Estimate the approximate need in human resources for the front-end implementation. To do this, find the corresponding numbers in the very last column of the table for every function that is chosen. Sum all numbers.
3. Using the upper part of the table, determine the set of features that should be present. To do this, check grey boxes in every line, where the function is chosen. Choose features in the columns with checked boxes.
4. Having the set of features, determine the components to be implemented, using the lower part of the table. To do this, check grey boxes in every column, where the feature is chosen. Choose components in the lines with checked boxes.
5. Estimate the approximate need in human resources for the back-end implementation. To do this, find the corresponding numbers in the very last column of the table for every component that is chosen. Sum all number.
6. Estimate the overall implementation cost. Sum the results of step 2 and step 5

Having estimated the overall implementation cost, one should also consider costs of

- Project Management (10%)
- Documentation process (5%)
- Testing (20%)

ANNEXES

ANNEX 1. Human Resource Estimation Matrix

FUNCTIONS	Search	Coordinate Translation	Thesaurus	Map	Linking	Editor	Record Templates	Multiple Records	Multiple Record Types	Access control	Forum	Internationalisation	Users	Transfer	FRONT-END, person days
Define geographic criteria of search by typing a geographical location															2
SUBFUNCTION: Specify reference system															
Define geographic criteria of search by selecting a location in a geographical thesaurus															10
Define geographic criteria of search either by delimiting an area on an interactive map															20
Define thematic criteria of search by typing free keywords															1
Define thematic criteria of search by selecting keywords in a thematic thesaurus															10
Consult the definition of a specific term via an interactive glossary															3
Find out the LIS data repository which data sources are matching the selected criteria															5
Select a data source in a list of items matching the selected criteria and view its related metadata															5
If available, download the data															1
If available, view the data itself with an appropriate viewer (Word, Excel, etc.)															1
Edit and save new metadata records (1)															20
SUBFUNCTION: Create metadata records from Template and save them as Template															
SUBFUNCTION: Choose record															
SUBFUNCTION: Choose record type															
Attach to the record a set of keywords from a thesaurus															2

Define the access rights to download or view such a data file (optional)															10
Attach to the record a data file (optional)															2
Access one or several fora related to coastal zone management															3
Post new message to a specific forum															3
Read previous messages posted to a specific forum															3
Select the language for the criteria of search, the interface, and the glossary															2
Modify or delete existing metadata records															3
Create or delete a new forum															2
Create, modify, delete user profiles with specific access rights															5
Create, modify, delete new users within each user profile															5
Regular transfer of metadata, and if available data themselves, towards EUROSION server.															3

COMPONENTS

															BACK-END, person days
Metadata Storage															5
Filter															2
Storage Guard															5
Document Holder															3
Record Holder Proxy															2
Transformer Proxy															5
Record Pool															3
Record Types Registry															7
Editor															8
Query Translator															5
Translators Registry															3
Coordinate Translator															10
Thesaurus Source															15
Thesaurus Browser															10
Map Viewer															7

Multipart Decoder																3
Upload Manager																5
Upload Guard																5
Record Properties																3
Properties Guard																3
Identity																5
Access Rights																3
Bundles																4
Transferor																5
Single sign-on forum component																5

ANNEX 2. Components Synopsis

COMPONENT	INTERFACE	ROLE	IMPLEMENTATION
Metadata Storage	Metadata Storage	MetadataStorage	Metadata Storage
Record Filter Proxy			Filter
Storage Guard			Storage Guard
Template Filter		TemplateFilter	Filter
User Profiles Registry		UserProfiles	Metadata Storage
Profiles Guard			Storage Guard
Record Holder	Document Holder	RecordHolder	Document Holder
Record Holder Proxy			Record Holder Proxy
Transformer Proxy			Transformer Proxy
Query Holder		QueryHolder	Document Holder
Profile Holder		ProfileHolder	Document Holder
Record Pool	Record Pool	RecordPool	Record Pool
Record Types Registry	Record Types Registry	RecordTypesRegistry	Record Types Registry
Record Editor	Editor	RecordEditor	Editor
Query Editor	Editor	QueryEditor	Editor
Profile Editor	Editor	ProfileEditor	Editor
Query Translator	Query Translator	QueryTranslator	QueryTranslator
Translators Registry	Translators Registry	TranslatorsRegistry	Translators Registry
Coordinate Translator	Coordinate Translator	<not relevant>	Coordinate Translator
Geographic Thesaurus Source	Thesaurus Source	GeographicThesaurusSource	Thesaurus Source
Thematic Thesaurus Source		ThematicThesaurusSource	Thesaurus Source

Geographic Thesaurus Browser	Thesaurus Browser	GeographicThesaurusBrowser	Thesaurus Browser
Thematic Thesaurus Browser		ThematicThesaurusBrowser	Thesaurus Browser
Map Viewer	Map Viewer	MapViewer	Map Viewer
Multipart Decoder	Multipart Decoder	MultipartDecoder	Multipart Decoder
Upload Manager	Upload Manager	UploadManager	Upload Manager
Upload Guard			Upload Guard
Record Properties	Record Properties	RecordProperties	Record Properties
Properties Guard			Properties Guard
Identity	Identity	Identity	Identity
Access Rights	Access Rights	AccessRights	AccessRights
Bundles	Bundles	Bundles	Bundles
Transferor	Transferor	Transferor	Transferor

REFERENCES

1. Aspect-oriented software development, Web site: http://aosd.net (There is a document on this site, containing the comprehensive bibliography of resources on aspect-oriented software development)
2. AspectJ project, http://www.eclipse.org/aspectj/, programming language that is actually extension to Java, implementing aspects and related concepts.
3. M. Kande, J. Kienzle, A. Strohmeier. From AOP to UML: Towards an Aspect-Oriented Architectural Modelling Approach. Software Engineering Laboratory, Swiss Federal Institute of Technology Lausanne.
4. R. N. Taylor, G. F. Johnson, Separations of Concerns in the Chiron-1 User Interface Development and Management System. Proceedings of InterCHI `93, Amsterdam, May 1993, 367-374.
5. B. A. Myers. Separating application code from toolkits: Eliminating the spaghetti of call-backs. In Proceedings of the ACM Symposium on User Interface Software Technology, pages 95-105, Hilton Head, South Carolina, November 1991
6. Brad A. Myers. "Graphical User Interface Programming," CRC Handbook of Computer Science and Engineering - 2nd Edition. Allen B. Tucker, editor in chief. Boca Raton, FL: CRC Press, Inc., 2003.
7. A. Valikov, A. Akhounov, A. Schmidt. A Model-Transformer Architecture for Web Applications. Proceedings of TES 2002 Workshop, Hong Kong, China, August 2002, 29-37.
8. I. Sommerville, P. Sawyer. Viewpoints : Principles, Problems and Practical Approach to Requirements Engineering. Cooperative System Engineering Group, Lancaster University.
9. the Apache Avalon Project. http://avalon.apache.org/
10. E. Gamma, R. Helm, R. Johnson, J. Vlissdes. Design patterns: Elements of reusable object-oriented software., Addison-Wesley, 1996.
11. W. J. Brown, R.C. Malveau, H. W. “Skip” McCormick III, T. J. Mowbray. AntiPattens: Refactoring Software, Architectures and Projects in Crisis. John Wiley & Sons, Inc., 1998.
12. van der Vlist, E. XML Schema. O’Relly, 2002.