DELIVERABLE

Project Acronym:	**EuDML**
Grant Agreement number:	**250503**
Project Title:	**The European Digital Mathematics Library**

Deliverable 4.1 – EuDML global system functional specification

Revision: 1.1

Authors:

José Borbinha (IST)

Aleksander Nowinski (ICM)

Wojtek Sylwestrzak (ICM)

Gilberto Pedrosa (IST)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	PU
C	Confidential, only for members of the consortium and the Commission Services	

19/12/2010

Revision History

Revision	Date	Author	Organisation	Description
0.01	28.03.2010	Wojtek Sylwestrzak	ICM	First draft framework
0.09	25.08.2010	Wojtek Sylwestrzak	ICM	Second draft framework
0.10	12.10.2010	José Borbinha	IST	Report outline
0.20	28.10.2010	José Borbinha	IST	First consolidated version
0.30	18.11.2010	Aleksander Nowiński	ICM	Revision and editing
0.40	28.11.2010	José Borbinha	IST	Extended restructure
0.90	6.12.2010	José Borbinha	IST	Version for internal review
0.91	6.12.2010	Wojtek Sylwestrzak	ICM	Copyedit
0.92	06-12-2010	Gilberto Pedrosa	IST	Revision
1.00	08-12-2010	José Borbinha	IST	Final version
1.01	14-12-2010	José Borbinha	IST	Final version - review
1.10	19-12-2010	José Borbinha	IST	Version for delivery

The document versions are controlled by the EuDML SVN version control system. The above revisions are for informational purpose only and do not reflect the actual SVN versions.

Statement of originality:

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Document Index

1. Introduction

The purpose of this document is to define the main global functional specification of the EuDML system. This document expresses two **views**, covering the description and alignment of the **concerns** of two main **viewpoints**[1]:

- **Architecture of Processes**: A high level functional specification of the **business processes** of the system, described by a general context and related scenarios.
- **Architecture of Information**: A conceptual representation of the high level **information entities** of the system.

The concepts or architecture, view, concern and viewpoint follow the definitions given by the IEEE-1471 – IEEE Recommended Practice for Architectural Description for Software-Intensive Systems[2]. The use of the terms **architecture of processes** and **architecture of information** to name respectively the views for processes and information entities are inspired by the thinking and practice of **enterprise architecture**[3].

It is important to stress that the analysis and design here presented are conceptual and mainly functional. It is focused on the high-level perspective and context of the system, and therefore it is not intended to impose or even propose concrete technical or non-functional requirements for the services, applications or underlying technology, as those concerns will be the covered in the deliverable "D4.2 - EuDML global system functional specification and design". However, there are expressed assumptions about that.

It is also important to stress that the contents of this document might be revised during the work to develop the D4.2. In that sense the main purpose of this document is to support and guide the project's development towards the delivery of D4.2.

To express the architectural perspectives here addressed, this document uses two classes of statements:

- **Rule**: A **rule** is intended to be a business rule, defined explicitly according to the understanding of the general objectives of the EuDML project and the strategy to reach them. Rules must motivate requirements. **It might not be always immediately obvious how to turn a rule in a clear requirement or set of requirements, nevertheless their statements are very important to help in defining the overall context of the system and contributing to an overall shared perception of it**. Each rule is numbered with the style "(Rule x)".
- **Requirement**: A **requirement** is a statement about a necessary characteristic or attribute that the system must perform. Therefore, the requirements must be especially considered for all the decision concerning the design, development and deployment of the system. Requirements also must be traceable to a specific property of the system or functionality of one or more of its services. **Even if not clearly stated, a requirement has to be always understood as something that the system <u>must</u> perform, so its consequence must be clearly perceived.** Each requirement is numbered with the style "[Req. x]".
- **Open Issue:** An **open issue** is an issue identified at the moment of writing this document, that was considered potentially interesting for EuDML but for which it was not possible to reach a comprehensive understanding of its relevance or feasibility, therefore it is here only to serve as a document for future addressing. In this sense **an open issue is neither a rule nor a requirement, but only an issue for <u>eventual</u> future addressing**. Each open issue is numbered with the style "{Open x}".

[1] These concepts follow the definitions given by the IEEE-1471 – IEEE Recommended Practice for Architectural Description for Software-Intensive Systems (http://standards.ieee.org/findstds/standard/1471-2000.html; http://en.wikipedia.org/wiki/IEEE_1471)

[2] http://standards.ieee.org/findstds/standard/1471-2000.html; http://en.wikipedia.org/wiki/IEEE_1471

[3] http://en.wikipedia.org/wiki/Enterprise_architecture

Rules, requirements and open issues are expressed in plain text, with references to diagrams when relevant. The diagrams are formal UML[4] diagrams, namely Use Case diagrams to represent the Architecture of Processes and Domain diagrams to represent the Architecture of Information.

The remaining of this document is organized as follow:

- Chapter 2 describes the concepts making the EuDML Architectural Framework. The focus is on the concepts of Architecture of Processes and Architecture of Information, the main focus of this document, complemented by the the Architecture of Services (to be detailed in the deliverable D4.2);
- Chapter 3 describes the Architecture of Processes, which is the main purpose of this document;
- Chapter 4 describes the Architecture of Information (relevant for a first time reader of this document, in order to better understand the Architecture of Processes, it might be advised to give at least a fast overview of this chapter before reading the Chapter 2);
- Chapter 5 makes reference to the main assumptions taken in this document, presented as a record to be considered for D4.2;
- Finally, Chapter 6 describes the process to be followed from now for the further steps for maintenance and update of this document and the alignments with the rest of the EuDML architectural design.

[4] http://www.uml.org/

2. The EuDML Architectural Framework

The conceptual modelling of the EuDML system is expressed according to an **architectural framework**[5] stressing the following views[6]:

- **Architecture of Processes** – The behaviour of the system is modelled, in its highest perspective, as UML Use Case. At a more detailed level, these use cases are specified in an architecture of processes, modelled as BPMN[7] or UML Activity diagrams[8], accordingly to the convenience and its effectiveness.
- **Architecture of Information** – The architecture of information represents the main **information entities** created, transformed and manipulated in the EuDML system, as well as their relations. The architecture of information is modelled by UML Classes Diagrams, used with the purpose of representing a domain[9]. It is important to understand that an Information Resource is a business concept, independent of its technical representation, which can assume many forms. For example, an "Author" can be represented in a technical component as an XML file according to a specific schema in one component, and in another component as according to other XML schema, and in a third component stored in a relational database according to the local specific table's structure. However, independently of all those representations, the Information Resource "Author" must be always the same (it will have implications and specific requirements for the naming and the management of replications of these entities).
- **Architecture of Services** – The architecture of services is a structural and formal modelling of the structure of the system. This architecture is expected to follow the basic principles of a SOA – Service Oriented Architecture, and of the REST model when pragmatically relevant, and is represented as UML Components Diagrams[10].
- **Architecture of Applications** – Technical applications represent a specific technical solution, reused or developed within the project and sported by a specific technology. This architecture also must comprise a **Catalogue of Components.** The deployment of a specific application can be realized by one or more technological components. For example, a specific DBMS can be used, as a unique component, to support more than one technical application. The components are the lowest level entities represented, for example, in UML Components diagrams. Technological components are implemented using specific technology and have specific technological interfaces to make them possible to be integrated with other components to build applications.
- **Technological Architecture** – The technological architecture concerns the description of the fundamental technology and techniques used by the technological components, applications and executing environment (such as the operating system and the communication technology, middleware platforms, etc.).

This document provides high level definitions of the architecture of processes and of the architecture of information, while expresses assumptions about the remaining views[11].

[5] http://en.wikipedia.org/wiki/Enterprise_Architecture_framework
[6] The concept of "views" here considered follows the same concept in the *ANSI/IEEE 1471-2000, Recommended Practice for Architecture Description of Software-Intensive Systems* (http://www.iso-architecture.org/ieee-1471/)
[7] http://www.bpmn.org/
[8] http://en.wikipedia.org/wiki/Activity_diagram
[9] http://en.wikipedia.org/wiki/Domain_model
[10] http://en.wikipedia.org/wiki/Component_diagram
[11] In a more correct "enterprise architecture" set of views, also an "organizational architecture" should be conceptualized. However, for now, in the EuDML Architectural Framework, and for the sake of simplicity, the definition of "system" restrict only to its technological components.

3. Architecture of Processes

The EuDML generic Architecture of Processes is represented in the Figure 1, as a high level UML Use Case diagram. The remaining of this section provides a detailed description of each entity represented in this architecture, made of actors and use cases.

In this conceptualization, the actors interact with the system according to perceived use cases. The use cases modelled in this document make the actual representation of the highest level architecture of processes. Following is provided a description of each use case[12].

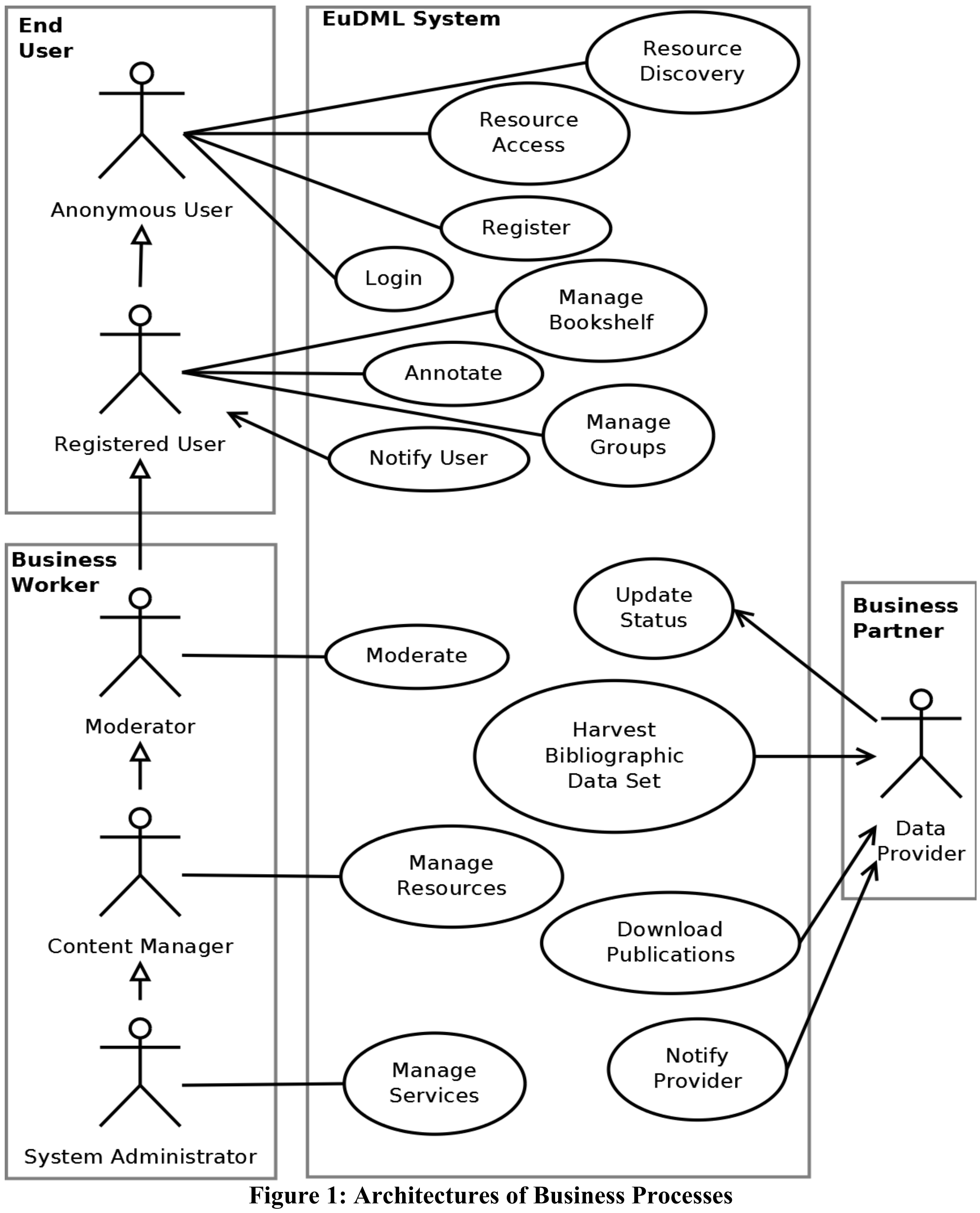

Figure 1: Architectures of Business Processes

12 More details of each use case can be developed in future releases of this general specification, or in the specific context of the Work Package where it will be designed or implemented.

3.1. Actors

These EuDML actors, already visible in the Figure 1, are detailed in Table 1.

Class of Actor		Role
End User	Anonymous User	A general class of users, which are not known to the system, but that take action to use the publicly available system services.
	Registered User	A user that is registered in the system and that is expected to log in using proper valid credentials This class of users represents the ideal intended user of the system.
Business Worker	Moderator	A worker responsible for responding to users requests concerning the management of the contributions and interactions of the EuDML community of End Users. That might comprise, for example, actions for moderation of discussions involving annotations and groups, pointing to the Content Manager errors detected by the users in the Information Resources, etc.
	Content Manager	A worker responsible for managing the system's Information Entities and its processing, including the monitoring of the actions related with the Data Providers, the ingestion of their contents, and the processing of that contents.
	System Administrator	A role for managing the overall system operation (the services) and related technological and communications infrastructure. It is a common and well known role in computer systems, comprising for example databases management, data and application backup, etc.
Business Partner	Data Provider	An external entity with which there is a formal or implied agreement to reuse metadata and publications in the EuDML services.

Table 1: EuDML Actors

Accordingly to these descriptions and the general objectives for the EUDML system, there are the following rules and requirements:

(Rule 1) EuDML Actors: The EuDML system's actors can be conceptualized in three major classes: End User, Business Worker and Business Partner

(Rule 1.1) End User: are those actors for which the system is primarily intended.

(Rule 1.1.1) End User as human or machine: an End User may be a human or another external system.

(Rule 1.1.2) Anonymous User: an End User which had not provided with success any accepted authentication is by default an Anonymous User.

(Rule 1.1.3) Registered User: an End User which provided with success an accepted authentication becomes a Registered User.

(Rule 1.1.4) The system must provide a set of relevant and attractive services for not only for Registered Users but also for Anonymous Users.

(Rule 1.1.5) The system must provide as many as possible authentication methods for an Anonymous User to prove identity and be recognized as Registered User.

(Rule 1.2)	**Business Worker: are those actors that have to engage in specific actions to maintain the system and its related technical services working, active and updated[13].**
(Rule 1.2.1)	**Business Workers are humans: all the Business Workers are expected to be human beings.**
(Rule 1.2.2)	**Moderator: a Business Worker who is expected to be highly knowledgeable of the mathematical context and community (ideally, he should be an expert and reputable member of the community).**
(Rule 1.2.3)	**Content Manager: a Business Worker who is expected to be knowledgeable of the EuDML technical system and organizational framework. He is expected to be reasonable knowledgeable of the mathematical context and community.**
(Rule 1.3)	**Business Parter: A Business Partner is an actor that contributes content to the system (metadata and publications), which is expected to occur under a specific previously agreed collaborative compromise[14].**
(Rule 1.3.1)	**Data Provider Agreement: Each Data Provider must have a cooperation agreement with the entity duly representing EuDML.**
(Rule 1.3.2)	**Every Content Manager must be aware of the cooperation agreements with Every Data Provider.**

3.2. Resource Discovery

Accessing resources in a digital library requires an End User to be able to find those resources. The capability of the system so support the use case for Resource Discovery with a high level of excellence will have important impact on the overall project success.

This section details the Resource Discovery use case, illustrated with the diagram in Figure 2.

(Rule 2)	**The amount of data expected within EuDML requires effective search and browsing operations, which will provide End User best possible feel and effectiveness.**
(Rule 3)	**Search and browsing must be possible in multiple indexes, each related with one entity of the Information Architecture, or with relevant attributes of these.**

[Req. 1]	**Indexes for Search and Browsing must comprise attributes from the Publications' Bibliographic Data, extracted from unstructured attributes, or extracted from the full-text of the Publication, comprising at least:**
[Req. 1.1]	**Title: an index of titles of publications.**
[Req. 1.2]	**Date: an index of publishing dates.**
[Req. 1.3]	**Authors: an index of names of authors.**
[Req. 1.4]	**Publisher: an index of names of publishers.**
[Req. 1.5]	**Subjects: an index of subjects, at least one based on the MSC– Mathematical Subject Classification system.**

[13] As stated before, those actors eventually also could be conceptualized as part of the architecture of the system, representing the “organizational architecture”...

[14] In future iterations of the system's design the concept of Business Partner might be extended to more classes of actors than the data providers. That will be especially the case of eventual existing systems to which EuDML might provide services or content, which are scenarios not incompatible with the EuDML vision.

[Req. 1.6]	**Formula: an index of mathematical expressions.**
[Req. 1.7]	**Annotations: an index of the full-texts in annotations.**
[Req. 1.8]	**Full-Text: an index of full texts extracted from the contents of the publications.**

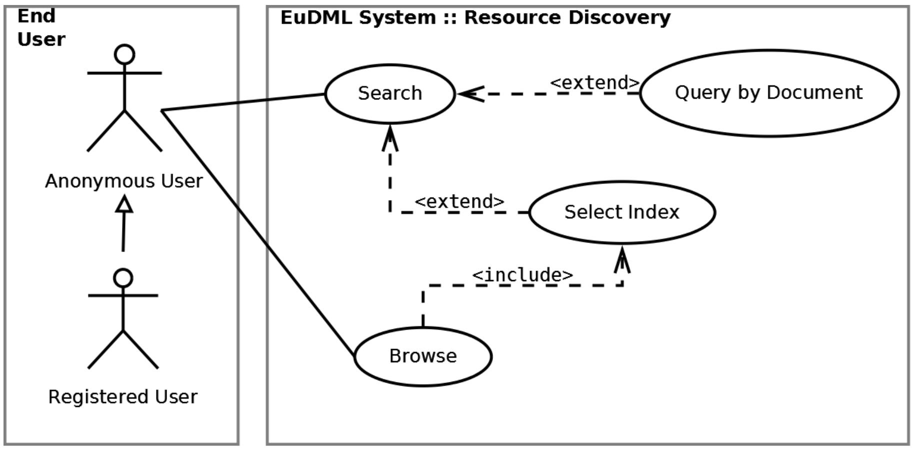

Figure 2: Search and Browse

{Open 1} Unexpected Data Attributes: In some cases it can happen that the original Bibliographic Data received from the Data Providers might contain attributes potentially relevant for searching which might be not previously identified as relevant for indexing. A case might be for example an attribute for annotations, which is not part of the expected schema but that the Data Provider decided to use as an extension. To be able to make use of those unexpected attributes, the consortium should consider to research on methods for also indexing those attributes (eventually, with techniques for information extraction).

As a consequence of the [Req. 1] there is a specific requirement for the systems' interfaces for Searching and Browsing:

[Req. 2]	**The EuDML system must provide interfaces suitable for human and remote systems as End Users.**
[Req. 2.1]	**Visual interface for humans: The system must provide for the End User a visual interface for human users, compatible with the most current HTML browsers.**
[Req. 2.2]	**Services interface for systems: The system must provide for the End User services interface for remote systems, with services' functional interfaces and information schemas properly described and published.**

3.2.1.Search

Search will be provided in the following options:

[Req. 3]	**The Search service must support multiple options:**
[Req. 3.1]	**A simple search must be provided ("google-like") with mechanisms for best possible simple query results in all the existing indexes.**
[Req. 3.2]	**An advanced search must be provided, with at least boolean queries on multiple indexes, for which the indexes defined in the [Req. 1] must be considered .**

[Req. 3.3]	**In queries for search the system must support, as arguments, free text provided by the user, or references to any other Information Resources.**
[Req. 3.3.1]	**When in the arguments of a query is an Information Resource, the search must be performed according to a similarity measure to be defined between the space of that specific Information Resource and the spaces of the other existing Information Entities.**
[Req. 3.3.2]	**The system must support the presentation of the results of a search operation in more than one index in two ways, depending of the specific scenario.**
[Req. 3.3.3]	**The system must be able to merge all the results in a unique results list (for which a generic ranking mechanism must be defined).**
[Req. 3.3.4]	**The system must be able to present the results in independent faceted lists, according to the searched indexes.**

The [Req. 3.3.1] intends to make it possible to search, for example, for a publication similar to other specific selected publication (which is represented in the Figure 2 by the use case “Query by Document”).

It also intends to cover the case of searching for formulae similar to a specific selected formula.

3.2.2.Browse

Browse will be provided in the following options:

[Req. 4]	**It must be possible to browse in all the <u>relevant</u> indexes listed in [Req. 1], namely:**
[Req. 4.1]	**Browse on Dates: It must be possible to browse on Dates of publication.**
[Req. 4.2]	**Browse on Authors: It must be possible to browse on Authors.**
[Req. 4.3]	**Browse on Publishers: It must be possible to browse on Publishers.**
[Req. 4.4]	**Browse on Subjects: It must be possible to browse on Subjects.**
[Req. 4.5]	**Presentation of browsing indexes: The presentation of each browsing index must be given in a unique list (for which case a specific ranking method must be defined).**
[Req. 4.6]	**Browsing on Publications: since publications can be made of other publications, like in journals that can have multiple issues and each issue be made of several papers, it must be given a special attention to browsing in this context.**
[Req. 4.7]	**Linking between indexes: it must be possible to browse between all the indexes and derived spaces, as for example browse from a certain Author to all its Publications, browse the Bookshelves where a certain Publication is mentioned, etc.**

{Open 2} Browse on formulae: EuDML intends to innovate on searching for mathematical formulae. This will represent an exceptional challenge, with a high risk of successful implementation. However, it also is understood as an important chance for EuDML to innovate and make a significant difference in the Mathematical world. Among others, this issue also might imply the consortium to research on how to rank mathematical expressions, so the results of a search can be presented in visual human readable indexes in an effective way. If the consortium succeeds in this area, this also might open the door to browsing in the space of mathematical s, which might be useful not only to help the End User selecting arguments for searching, but also to explore mathematical formulae in a mode of <u>serendipity</u>.

3.3. Resource Access

[Req. 5]	**From an access point of view, the End Users must be able to see detailed information of instances of relevant classes of Information Resources that are part of the EuDML Information Architecture.**
[Req. 5.1]	**The instances of the Information Resources suitable to be accessible by End Users can have access controls that must be respected by the system.**
[Req. 5.2]	**An End User must be able to access to Publications.**
[Req. 5.3]	**An End User must be able to access to Authors.**
[Req. 5.4]	**An End User must be able to access to Publishers.**
[Req. 5.5]	**An End User must be able to access to Subject.**
[Req. 5.6]	**An End User must be able to access to Registered Users.**
[Req. 5.7]	**An End User must be able to access to Bookshelves.**

In this moment the ontology for access control to Information Resources in EuDML it not completely defined, as a large part of it will depend of the negotiations with the Data Providers. Anyway, a generic model could be predicted according to the following principles:

- Public: All those Information Entities available for all End Users of the system, including Anonymous Users.
- Controlled: All those Information Resources available only to Registered Users, according to their specialized class (by default, a Content Manager must be able to have granted access to all the instances of Information Resources)

{Open 3} It is not clear in this moment if it also might be necessary to consider supporting specific access control lists for individual or specific lists of Registered Users, or even to Anonymous Users according to specific rules (an example might be an editor that offers access to its Publications to all the End Users accessing from a specific IP – Internet Protocol address, or range of addresses). This is clearly an open issue for future addressing.

Finally, it is important not to forget that the services support Resource Access must be aware of the requirement [Req. 2].

3.4. Register

The Registering use case scenario involves registering an End User into the system to allow log in and become a Registered User actor.

Registration involves the following steps:

1. Creating user credentials
2. Providing additional profile information (mandatory and optional, like real name, scientific title, institutional affiliations).
3. Customizing a publicly visible profile of the user (this will be the information to be available to all the End Users, and, accordingly, to be indexed for searching).

Creating user credentials may be processed according to two basic operation scenarios:

- Registering a new user with newly created credentials to be managed by the EuDML system: this scenario involves a minimal user registration, including asking for user email, new password and email confirmation of the registration.
- Registering a user who uses other external federation login system. This scenario assumes that the user has already some federation login. Currently considered are:
    - OpenId
    - Google Accounts (a variant of OpenId)

With federation the user does not have to create new user name and password. Managing separate account for each site is often assumed as not comfortable and repulsive for the user. Instead federation credentials will be used to authenticate user in the system.

Other steps of registration process (providing personal informations, customizing profile etc. may be performed immediately by the user or performed later.

[Req. 6]	**All the sensitive personal information to be delivered to any class of End Users, (including that structured or that presented in the public web pages of the service) that is understood as suitable to be used for future spam usage (especially email addresses) must be protected against techniques of automatic information extraction.**
[Req. 7]	**A Registered User must be able to edit all the data associated to its profile.**

3.5. Login

An Anonymous User becomes a Registered User by presenting authentication credentials to the system, depending on authentication method. The methods envisaged in EUDML are:

- User name/password
- Federation login

[Req. 8]	**In the minimum, at least directed input of user name and password must be supported to Anonymous Users for authentication.**

3.6. Manage Bookshelves

A Bookshelf is a list of Information Resources selected from the system which is managed and maintained by a user or a group of users. The purpose is to offer to the Registered Users a mechanism to create and maintain personal collection of Information Resources to bookmark, annotate and share.

[Req. 9]	**Bookshelves: Any Registered User can create one or more Bookshelves; owning a Bookshelf means to be able to edit it or even to remove it. To edit a Bookshelf means to be able to remove or to add to it one or more Resource, Subscriber or Annotation.**
[Req. 9.1]	**The creator of a Bookshelf must be automatically its first subscriber and owner; to subscribe a Bookshelf means to have it listed in the profile of the user and have it searchable and read access to all its content.**

[Req. 9.2]	**An owner of a Bookshelf must be always a subscriber of the same one Bookshelf.**
[Req. 9.3]	**An owner of a Bookshelf must be able to invite any other Registered User to subscribe the Bookshelf.**
[Req. 9.4]	**By default, any subscriber of a Bookshelf must be able to see the identity of all the other subscribers (except of Moderators and Content Managers, according the [Req. 9.11]).**
[Req. 9.5]	**An owner of a Bookshelf must be able to invite any other subscriber to share the ownership of that Bookshelf.**
[Req. 9.6]	**By default, a Bookshelf must be defined by the system as Private, which means that only its subscribers can search and access it.**
[Req. 9.7]	**Any owner must be able to define a Bookshelf as Public, making it searchable and accessible to all the End Users of the system (anonymous as well), without the need of explicit subscription.**
[Req. 9.8]	**Any owner must be able to change the definition of a Bookshelf from Public to Private.**
[Req. 9.9]	**By default, all Moderators must became automatically owners of all the Public Bookshelves**
[Req. 9.9.1]	**The Moderators must loose automatically the ownership of a Public Bookshelf if it is turned Private by any owner (including a Moderator).**
[Req. 9.10]	**By default, all Content Managers must become automatically owners of all the Bookshelves subscribed by two or more Registered Users, independently of whether they are Public or Private.**
[Req. 9.11]	**By default, the automatic subscription of a Bookshelf by a Moderator or a Content Manager must be hidden from all the other subscribers, which means that they can access all the content of the Bookshelf but they cannot edit it neither be seen by the other subscribers (thus, restricting their ownership privileges) .**
[Req. 9.12]	**A Moderator or a Content Manager can change in any moment their visibility in a Bookshelf they subscribe, becoming visible to the other subscribers; in that mode, the Moderator or the Content Manager regain their full ownership privileges.**
[Req. 10]	**When a user is removed from the system, all the Bookshelves owned by it as their unique owner are also removed.**

3.7. Annotate

(Rule 4) Annotation: An Annotation is a text added by a Registered User and commenting something relevant relating to one or more Information Resources.

[Req. 11]	**Concerning Annotations, each Information Resource must be in each moment in one of two possible states: Annotations On or Annotations Off.**
[Req. 11.1]	**Concerning Annotations, any Content Manager can, at any moment, change the state of any Information Resource.**
[Req. 11.2]	**By default, any Information Resource is in the state Annotations On.**
[Req. 11.3]	**Concerning Annotations, any owner of a Bookshelf can, at any moment, change the state of that Bookshelf.**
[Req. 11.4]	**Concerning Annotations, a Moderator or a Content Provider can take from the other owners that are only Registered User the right to change the state of the respective Bookshelf to Annotations On, retuning it back also at any moment.**

[Req. 11.5] **An owner of a Bookshelf always must be able to add or remove an Annotation to it.**

[Req. 11.6] **Create an Annotation: The system must offer a simple user interface mechanism for a user to explicitly create a new Annotation to any existing Information Resource.**

[Req. 11.6.1] **An annotation must be able to be defined by its creator as Public (anyone can search and access it) or Private (the Annotation is not indexed, and only the creator can access it).**

[Req. 11.6.2] **The creator of an Annotation must be able to change, in any moment, that Annotation from Public to Private and vice-versa.**

[Req. 11.7] **Since an Annotation is also an Information Resource, any Annotation must be able to be part of one or more threads of discussions, understood as sequences of Annotations involving any Information Resources.**

3.8. Moderate

Discussions supported by Annotations can be complex, demanding for example an intervention of a Moderator.

An example of a relevant scenario for this can be an Annotation from a subscriber complaining about abusive language of other subscriber in other Annotation. In a scenario like this the subscriber also might want that complain to be visible only to the Moderator, or visible to everyone. But more scenarios where Annotations can be relevant might exist, which raises new specific requirements:

[Req. 12] **Classifying Annotations: The system must support a pre-defined set of classes of Annotations, thus making it possible for a user to select the class notation when creating, which must trigger a specific behaviour from the system according to functional requirements defined for each class of Annotations.**

As examples, the following scenarios are expected to request a moderation:

- The Moderator must be able to act to correct the problem immediately:
  - Unintentional mistakes
  - Intended abusing or offensive annotations, comments, etc.
  - Intended spam
- Scenarios where the correction must be made by a Data Provider or an internal service of the system, and thus the Moderator must be able to act by annotating the problem and report it to the Content Administrator, so that it can be corrected in the next round of harvesting or internal processing:
  - Wrong original Bibliographic Data (as for example a wrong title): This is a problem of quality of data, so if a Content Administrator confirms it, that administrator must be able to act by reporting it to original Data Provider.
  - Wrong Association or Data Augmentation (as for example a wrong link for a publication): This is a problem of quality of the system, so if a Content Administrator confirms it, that administrator must be able to act by reporting it to the System Administrator (as it might be related with a software “bug” or a problem of quality in a specific internal service).

As a consequence, in this moment the following classes of annotations could be envisaged:

[Req. 12.1] **Regular Annotations: This is the implicit class of any annotation, which must not require any other specific behaviour. Users must not be asked to do anything specific to classify an annotation of this class.**

[Req. 12.2] **Special Annotations: To create an Annotation of this class the user must state it explicitly (by selecting it from a list of options, for example); in this case the system must issue a special warning for the Moderators, claiming their attention to this Annotation. Several classes of Annotations might exist in the system, namely:**

[Req. 12.2.1] **Report Error: This class of Annotations intends to claim the attention of a Moderator to some possible error in the system related with any Information Resource (like for example a mistake in a title or in an author); these Annotations must be visible to all the other subscribers, and as a consequence of it a Moderator is expected to take action in clarifying it or reporting it to a Content Manager.**

[Req. 12.2.2] **Public Complaint: This class of Annotation, which is also visible to all the other subscribers, intends to claim the attention of a Moderator to other Annotations where some user is expressing a socially unacceptable behaviour.**

[Req. 12.2.3] **Private Complaint: This class of Annotations, which is not visible to any of the other subscribers that are only Registered Users, intends to claim the attention of a Moderator to other Annotations where some user is expressing a socially unacceptable behaviour.**

3.9. Manage Groups

As EuDML wants to offer more than just plain library, users shall be able to organise themselves in Groups.

(Rule 5) **Conceptually, a Group is in itself a Registered User, just representing an aggregation of Registered Users. This way, any Group must be understood by the system as a regular Registered User, and any Registered User acting as a member of a Group must be seen as the whole Group.**

The idea of a Group is to create it spontaneously on user demands and allow easy group management for users, with no moderator nor personnel attention needed. Groups depending on privacy and subscription settings may provide amount of different functionalities and be a very flexible tool. Examples might be a closed research community, a lecture or seminar reading list, a public reading channel on certain topic, etc.

[Req. 13] **Any Registered User must be able to create a Group, becoming by that way the owner of the Group**

[Req. 13.1] **Each Group must be able to be defines as Private (it is not visible to non-members) or Public (it is visible to any End User).**

[Req. 13.2] **When a Group is Public, the policy to accept new Members also must be defined, which can be Self-subscription or Invitation. A Self-subscription of a Group means the users can take the initiative to subscribe the Group, and its subscription is immediately accepted; an Invitation means that the user only can take the initiative to subscribe in the following of an invitation from an actual owner.**

[Req. 13.3] **When a Group is Private, the subscription of new Members must be made only in the following of an Invitation from an actual owner.**

[Req. 13.4]	**An owner of a Group can invite any other Member of the Group to share the ownership. A Group can have more than one owner.**
[Req. 13.5]	**Only an owner can edit the attributes of the Group (as for example, create a Bookshelf or an Annotation on behalf of the Group).**
[Req. 13.6]	**Any Member of a Group must be always able to leave a Group, with no constrains.**
[Req. 13.7]	**Groups are lightweight, there is no limit on number of Groups in the system, and a user may join as many Groups as wanted.**

(Rule 6)	**A Group is also an "Information Resource", which can be indexed by one or more "Subject", and thus made searchable and browsing (this means to revise the Information Architecture). In this case the system must recognize when a Group is Public or Private, and behave accordingly (i.e., Private groups are visible only to its Owners or Members).**

3.10.Notify User

Registered users can manage a profile and preferences that makes it possible to the EuDML system to notify them of novelties, which might comprise selective dissemination of information (notify the user of new results in previously stored searches), new actions in active groups, etc.

[Req. 14]	**The system must be able to notify a Registered User by email.**
[Req. 15]	**A Registered User must be able to manage the email notifications, as for example disable notifications, filter notification event types, limit the frequency of notifications (for example, one per week), receive digest messages only, etc.**

3.11.Moderate

A Moderator is a Registered User, who has extended privileges to perform certain moderation actions. It is not yet decided who will perform this role (consortium members, society, experienced users), and it will probably depend on the project stage – in early phase system administrators and operators will do it, and at the project end probably it will involve more community effort.

A Moderator user is expected:

- To be a human being
- To be responsible for responding to users requests for moderation and annotation control (offensive language, spam)
- May remove or block Annotations in to Information Resources
- May perform other moderation activities, to be identified.

It must also be stated, that due to specific character of the mathematical community and limited, professional audience of the EuDML it is not expected to have massive problems with offensive language nor other social rules violations. Therefore, it is not expected that moderation will require an intensive commitment.

3.12.Manage Resources

The Content Manager is the actor responsible for the management of all the Information Resources. In this moment that is understood as performing the use cases represented in the Figure 3:

- **Ingest Bibliographic Data Set** means, in a normal scenario, to schedule the processes to harvest the data from the Data Providers. In alternative scenarios it might mean to ingest that data in the system manually.
- **Import Contents** is the analogous to the previous use case, but now for publications.
- In the following of the ingestion of new data and publications, the system must apply a full set of automatic or assisted processes to augment the data, as for example to extract mathematical formulae. **Review Data Augmentations** is the use case for a Content Manager to analyse, evaluate and eventually confirm or reject those results. This use case also must comprise the execution of methods to normalize the new values of the attributes.
- After augmentation of the data, methods also must be executed to establish associations between the new Data Set and the Data already existing in the system. This is, for example, when the name of an author in the new Data Set will be matched against those already registered in the system, to detect if the author already exists or is new. **Review Resources Associations** is the use case for a Content Manager to analyse, evaluate and eventually confirm or reject those results.

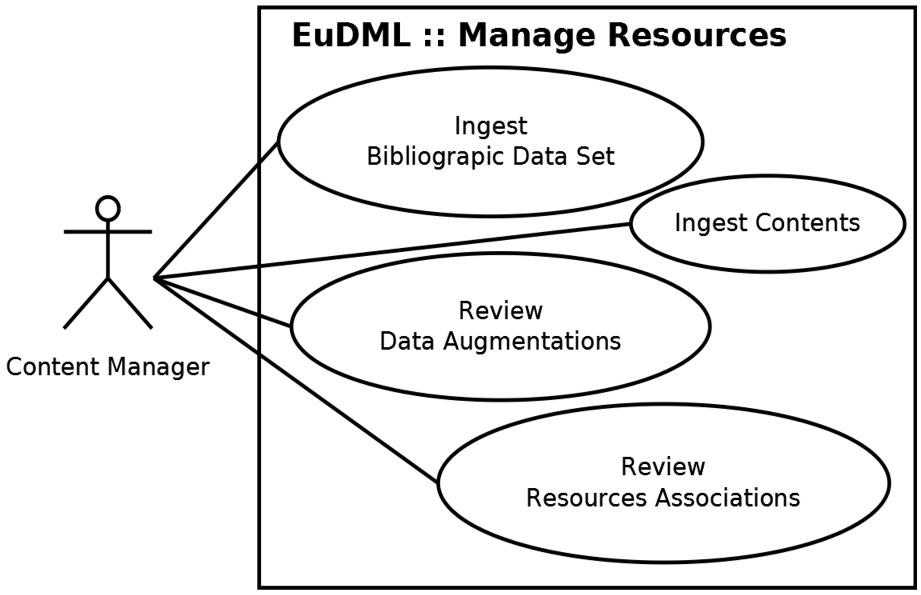
EuDML :: Manage Resources
Ingest Bibliograpic Data Set
Ingest Contents
Review Data Augmentations
Review Resources Associations
Content Manager

Figure 3: Use case Manage Content

3.13.Manage System and Services

Managing the system comprises a common set of tasks involving management of the underlying system infrastructure and application environment, such as:

- starting and shutting down the overall system (or only specific services)
- monitoring system status
- performs system upgrades

- maintaining and accessing system logs
- monitoring underlying structure (database, file system, server)

Full scale operation of the system requires separate and important task of providing system backup and recovery. It is important to understand, that as the amount of user generated data will grow, it will become more and more important to protect this work. For example, the intention of providing users capabilities of annotating scientific work is to share the knowledge and create new added value, and this achievements must be preserved with care. Therefore backup and recovery task are more specific use cases of this scenario.

3.14.Harvest Bibliographic Data Set

The Bibliographic Data Sets are harvested from the Data Providers, ideally by OAI-PMH (but other techniques might be considered along the project's lifetime). It is assumed that the Bibliographic Data Sets can be changed by the Data Providers at any moment, so they might have to be harvested regularly.

[Req. 16]	**The system must be able to take action to harvest a Bibliographic Data Set from a Data Provider.**
[Req. 16.1]	**The harvesting of the Bibliographic Data Sets must be scheduled according to the agreements with the Data Providers.**

3.15.Harvest Publications

Each Bibliographic Record is expected to refer to one Publication. Depending of the agreement with the Data Provider, that Publication also can be harvested.

[Req. 17]	**The system must be able to take action to download Publications from the Data Providers.**

The main purpose for this use case is to make it possible to process the publications locally, for data augmentation, full text indexing, etc.

However, it also might be relevant for local permanent storage and primary source of access to the publication by the End Users, depending of the existing agreement with the Data Provider.

3.16.Notify Provider

[Req. 18]	**The system must be able to notify the Data Providers of the results of harvesting actions, requests from the Content Administrators about errors in metadata or in publications, etc.**

4. Architecture of Information

The generic EuDML Architecture of Information is represented in the Figure 4, as an UML Domain Diagram. The definition of each element and of its relations with the other elements are described below, in the remaining part of this section.

It is important to stress that this is a conceptual representation of the Architecture of Information, representing the Information Entities to be considered in the description of the Architecture of Processes and in the services to be described in the Architecture of Services.

Aligned with the principles of a good information architecture the following initial requirements and rules are presented.

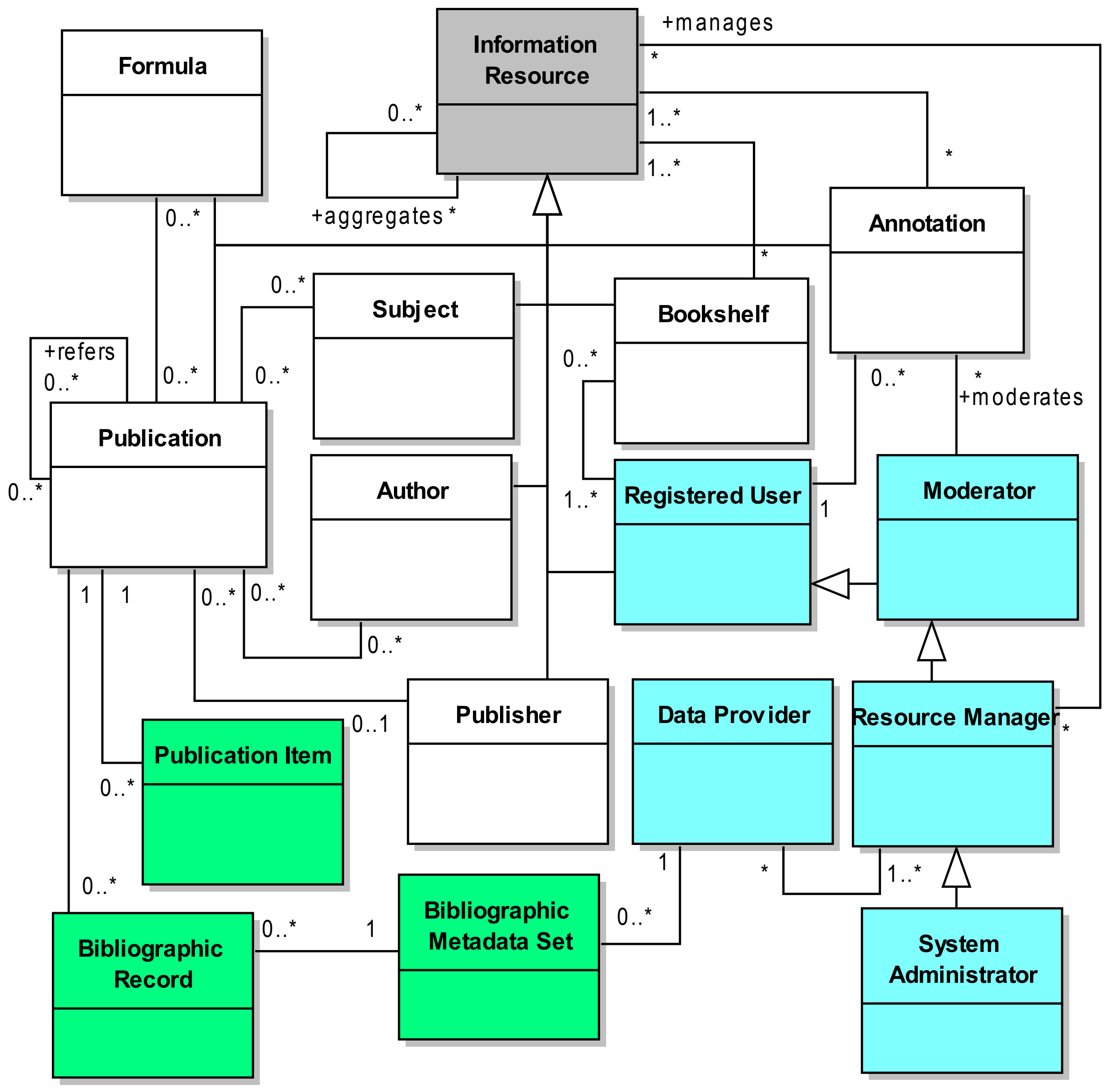

Figure 4: Information Domain

The concept of Information Resource is a generic abstract concept, not exposed to the End Users, but relevant in the information architecture to support assertions that are common to a set of other concepts exposed to the End Users: Author, Subject, Publisher, Publication, Formula, Bookshelf and Annotation. The main focus of this document is to define requirements related to these concepts.

The concepts of Bibliographic Metadata Set and Bibliographic Record are closely related to the focus of the WP3.

The concept of Publication Item refers to each possible item for each publication with full-text to be processed in EuDML.

The remaining entities (Data Provider, Registered User, Moderator, Resource Manager and System Administrator) are the conceptual representation in the information architecture of the End Users, Business Workers and Business Partners (essential to manage the related attributes).

4.1. About Users

A number of Information Entities represented in the information architecture are representations of the system's actors, already described in the Table 1, and thus with no need here for more details.

That is the case for Registered User, Moderator, Resource Manager, System Administrator and Data Provider.

4.2. Information Resource

An Information Resource is any class of structured information with potential relevant value for the End Users.

4.2.1. Classes of Information Resources

(Rule 7)	**The classes of Information Resource in this moment considered are:**
(Rule 7.1)	**Publication – A Publication may be an article, a journal, a book, a conference proceedings, etc. A Publication must have well-known set of fields and attributes, which definition is in scope of the WP5. A Publication can be part of other Publication, as for example articles are parts of journal issues, which in their turn are parts of a journal volumes. Each of those Information Resources may also have specific metadata, like issue number, Editor-in-Chief etc.**
(Rule 7.1.1)	**A Publication can refer to other publications identified in its bibliographic references.**
(Rule 7.1.2)	**A Publication only found as a bibliographic reference in other Publication also becomes a Publication referenced in the EUDML information architecture. However, the use cases for Resource Discovery and Access must make clear to the End User that fact. For example, in these cases these publication have not Bibliographic Records harvested from the Data Providers, but only a bibliographic description created by the EuDML system (usually, extracted from the Bibliographic Record or from the full text of other Publication).**
(Rule 7.2)	**Author – The author of a Publication, as found in the Bibliographic Data**
(Rule 7.3)	**Publisher – The publisher of a Publication, as found in the Bibliographic Data**

(Rule 7.4) Subject – Publications may be classified within a subject classification, and this classification has to be known and defined within the system. Classification, such as the Mathematical Subject Classification, is composed of categories, which form a category tree. Each publication which is classified within certain classification has assigned one or more categories within specific classification.

(Rule 7.5) Formula – A representation of a mathematical expression

(Rule 7.6) Bookshelf – An entity mainly intended to be shared by the End Users and aggregating one or more Information Resources.

(Rule 7.7) Annotation – An entity mainly intended to be shared by the End Users and making an explicit reference to one or more Information Resources.

(Rule 7.8) Registered User (see 3.1.)

(Rule 7.9) Group – a group of Registered Users, which by this way can be aggregated to make them look like a unique Registered User (see 3.9.).

A clear statement must be made about the fact that existence of all these classes of Information Resources must not imply automatically that all their instances will be freely available for access or even discovery; the policies about that must be defined for each class and always taken in the correct consideration by the EuDML services.

4.2.2.About Identifiers of Information Resources

It must be assumed that multiple instances of the same Information Resource might be created in uncoordinated scenarios, especially when multiple services of the system act in different moments. For example, the system might create two different instances of two Authors that at some moment are identified as being about the same person. This also may occur for Publishers and Publications (this scenario is plausible assuming that a Data Provider will provide Publications resulting from digitization of historical works) and especially for Formulae (mathematical expressions). There will be several consequences of that fact, which it is necessary to be aware of:

- Each of these instances will receive its own unique identifier[15] in the moment of its creation in the system, which might be shared among the services and eventually exposed to the outside, which implies to exclude the hypothesis of merging one instance in the other and removing the first one.
- Even if the Information Resource is the same, each of these instances can have different information contents. For example, one instance of an Author could have the date or birth , while in the other instance it is not present but a date of death is instead. More complex yet, both the instances can have the same attributes filled with data, but they can be different (like two different birth dates).
- Realizing that two Information Resource instances are really about the same Information Resource is valuable knowledge that must be registered and managed. An elegant solution for that is to assume that an Information Resource can aggregate other Information Resources. This way the rules to apply to this scenario must be:

(Rule 8) When two instances of the same Information Resource are detected a new instance must be created with its own unique identifier, to aggregate the original instances;

[15] A unique identifier is an identifier assigned to only one entity, and that therefore never can be reused to be assigned to any other identifier. This must not be confused with the fact that a same resource cannot have more than one identifier, which is something that always have to be supported as part of the fundamental infrastructure to effectively consolidate information.

	in the future, if more such instances are discovered, they only need to be associated to the same aggregation.
(Rule 9)	**All the EuDML services must be aware that aggregations might exist, and be conceived to take the best advantage of them when relevant. However, for simplicity:**
(Rule 9.1)	**The instance created to manage this aggregation must have its own attributes filled with data copied from the aggregated instances.**
(Rule 9.2)	**As this will depend of the specific class of the Information Resource, specific aggregation techniques must be defined for each class (which is especially relevant for when two or more instances have attributes with conflicting data).**
(Rule 9.3)	**A special attention must be given to the resolution of identifiers. For example, assuming there is an aggregation X aggregating the resources Y and Z, when a service requests other service to solve Y for him (which in most cases means a kind of "get" call of the first service to the second, with an identifier as argument and expecting to receive back as a reply an information structure with the data of a related Information Resource), it must be clearly defined and justified, service by service, if the returned information structure will be the one corresponding to Y or the one corresponding X.**
(Rule 9.3.1)	**A possible way to address this problem in a generic way might be for those services to request for those "get" call one more argument stating if the requested information structure should be the original one or the aggregation. This means that, for the previous example, the caller service could ask explicitly to be retuned to it Y, or to be returned the aggregation where Y belongs, if it exists.**

4.3. Bibliographic Record

Digital libraries are basically oriented on bibliographic records searching, browsing and presentation, and EuDML is not exception.

A number of Bibliographic Record formats are expected on input in EuDML. To make it possible to get the best of any record in any moment, the original Bibliographic Records must be stored and preserved in the system, available to any service willing to make good use of them.

However, all the imported Bibliographic Records also will be translated to a generic common format, for convenience of the services willing to avoid to deal with that heterogeneity. The definition of that format is an issue for the WP 3.

(Rule 10)	**The term "metadata" is usually, in the scope of the community of Digital Libraries, understood as a designation for any instance of an information entity, especially for bibliographic entities. This often conflicts with the use of the same term in Computer Science and Engineering, where it refers to information about the definition of the structures of these information entities, and not the instances of these entities. Due to this conflict, the standalone use of the term "metadata" must be avoided in EuDML. In alternative, the term to refer any instance of bibliographic entity must be "Bibliographic Record", and the term to refer to a set of Bibliographic Records must be "Bibliographic Data Set".**

4.4. Bibliographic Data Set

A Bibliographic Data Set is a collection of Bibliographic Records provided by a specific Data Provider.

The conceptual and schematic definition of these entities is the main focus of the WP3.

5. Assumptions

EuDML project is focused on Publications and Bibliographic Metadata. Therefore its information architecture model is concentrated on providing effective representation for processing.

Current achievements in information space management allow on the fly conversion of various bibliographic and other metadata formats and there is no need to stick for internal metadata format to perform all operations. Although on the other side it must be kept in mind, that the final goal of the project is to offer the user the best possible view on the system's contents. Effective presentation of the bibliographic metadata, bookshelves, etc. requires certain assumptions on presented data stored within the system.

5.1. Overall assumptions

It must be understood, that EuDML is an integration, not a research project. Therefore it does not assume a full starting from scratch, but rather an adapted integration of existing software applications when possible. This implies that certain architectural assumptions already have been made, and are not expected to be changed. This will allow rapid development and providing a working system early.

5.2. Data storage

All the structured data to be shared among the services, especially the instances of entities of the Information Resources, will be stored in XML formats, within the system, in a centralized data storage service.

This must comprise the entities created by the service of the system, but also the Bibliographic Records given by the Data Providers. It is important that they are available for reuse at any moment.

Wherever required, specialized services must provide the necessary format conversions and data extraction capabilities. This assumption will have serious implications for the system's architecture, as it requires mapping the relations among information entities to be stored and maintained externally within index and other services.

5.3. Authoritative and non-authoritative data

Within the system a number of services will store internally the same conceptual entity. For example, the a central data storage service will store an original Bibliographic Data Record with all its original attributes, but a search service might store locally the same Bibliographic Data Record with only the attributes required for its indexes, while a semantic network service might store it locally with extra attributes created locally.

It is important to have control over quality and coherence of the data in this scenario so the concept of authoritative data is introduced:

(Rule 11)	**Authoritative form is a representation of an Information Entity controlled by the centralized data storage service and that can be shared by all the services.**
(Rule 12)	**Non-authoritative form is a representation of an Information Entity controlled by only one specific service of the system, and that thus that must be shared only with the centralized data storage service.**
(Rule 13)	**Any time a service of the system changes an attribute of an Information Entity that also might be relevant to any other service, the service where that change occurred must submit the change to the centralized data storage service**
(Rule 14)	**Any time a service of the system is requiring the most recent version of an Information Entity it must request it only from the centralized data storage service.**

This approach implies, that any enhancements in the data generated within the project must be reflected back in the central data repository.

The advantage of this centralized approach is that it makes it much easier to keep the Architecture of Information consistent, without the need to consider loop dependencies in data derivations among the services.

5.4. Assumptions on the Technological Architecture

5.4.1. Operating System Independent

The software which will be the outcome of the project will must not be dependent on any specific operating system features.

EuDML will be not developed as a unique software suite, but as a system integrating services. Therefore the system is expected to be operated in a certain environment, but unnecessary assumptions and architectural restrictions must be avoided, since they might result in enforcing a certain fixed environment (such as an operating system version).

The services will be developed in Java, which is a cross platform language

It is unavoidable to choose certain database engines, but probably this will be the only decision which may enforce any specific system's configuration.

5.4.2. Browser independent

EuDML may not rely on any browser-specific features on the client side. This implies, that the basic functionality of the system must be available for each possible browser (including very simple ones, like text mode and embedded), and each modern browser should provide full user experience of the system. This includes especially support for each form of mathematical equations in content and bibliographical metadata as well as in the annotations.

5.4.3. Open software

Software which will form core of the EuDML system will be developed and available on open licence not only to the project partners, but also to other projects and initiatives. Although this may not be an architectural assumption, it has deep consequences for the process of selection of external existing components for reuse, selection of programming tools, as well as standards to be adopted.

5.4.4.XML

An XML will be used as the basic representation language for storage and interchange of instances of the information entities within the system,. It will be used not only as exchange format for bibliographic metadata, but to store bookshelves, user profiles and other informations as well.

5.4.5.SOA and REST

The system will be based on a number of services communicating with each other. To meet the performance expectations and avoid unnecessary focus on the communication layer, a pluggable service access layer will be used. This will allow to adopt and exploit existing environments in an efficient manner, avoiding the effort spent on direct protocol tuning. Depending on the deployment scenario the inter-service communication will use SOAP protocol, HTTP invoke or direct Java invocation (if run within the same service container), or REST (which might be especially relevant for web interface access to service layer, to provide easy and lightweight access for complex services).

5.5. Assumptions on the Architecture of Components

As EuDML is an integration project, the development of basic service components will be avoided, and the effort will focus on high level service composition instead. Therefore the following service components will be used, or are considered parts of the system.

5.5.1.YADDA Services Set

YADDA offers a basic service set necessary to set up a scalable digital library back-end base within short time. YADDA services will be used to provide the basic functionalities necessary for the web interface implementation, such as: bibliographic metadata and content storage, search/indexing service, user management and relation browsing service. This will allow to start up rapidly with a working core of the platform and invest the time in extended functionalities rather than basics of the system.

The following YADDA suite services are expected to be used in the project:

- YADDA storage service – to be used as the Data Repository for metadata, content, and annotation storage
- YADDA search service – a flexible indexing and search service, currently ported to Solr
- YADDA structured browse – a simplified database view dedicated for paged browsing views

- YADDA user directory – an engine for storing users, roles and groups
- YADDA AAS – an authentication/authorization component
- YADDA workflow manager – processing component, which allows to perform indexing and other data processing tasks on repository content

Those services will be selected and adopted to perform specific low-level tasks.

5.5.2.REPOX – Data Harvesting Manager

REPOX is a component to manage harvesting processes of data transfer from the Data Providers. In the scope of EuDML that will comprise specifically the support to the use case Harvesting Metadata.

In this sense REPOX will be the first immediate component to receive the Bibliographic Data Sets from the Data Providers, which must be after that stored in the Data Repository.

REPOX also is relevant to interoperate with the specific service to support the use case Download Publications, precisely by providing the necessary information taken from the Bibliographic Data Sets provided by the Data Providers.

5.5.3.MDR – Metadata Registry

The EuDML MDR is a Metadata Registry developed according to the ISO/IEC 11179. In this sense it is important to stress that this component is intended to support a service that will not store Information Resources, but the metadata about their representations!

The main purpose of this service is to allow the registration of data schemas and the maintenance of those registrations, as also to register mappings between those registered schemas.

It also allows the exporting of the registered information. For example, schemas can be exported in XSD, and mappings in XSL suitable to be applied to data instances in XML files structured according to those XSD.

5.5.4.SOLR / LUCENE – Indexing and Searching

Search service will be most important service within the system, as usability study have proved. EuDML will use most advanced open source techniques to provide reliable and accurate search results. YADDA Search Service will be used, which is based on Apache Lucene and Solr, state of the art search engines, which – enveloped with proper service interface – will support all expected user search functionalities.

6. Follow-up Processes

6.1. About the EuDML software development process

The EuDML software development process will follow elements of an agile methodology, which is believed to be the most suitable for an initiative where many services and components will be modelled in detail and developed by different partners.

As the project goes forward, two main processes will run in parallel, which will lead finally to create the desired system. As the system specification becomes more detailed, it will evolve first into the deliverables ***D4.2: EuDML global system functional specification and design*** and latter on in ***D4.3: EuDML global system functional specification and design – Revision***.

The development process will be managed taking advantage of the project's wiki, to which all partners will have access and so problems may be identified as soon as possible. Also this encourages shorter development cycles, as it requires less formal structure of the specification and documents, which is usually more efficient than traditional complex and inflexible documentation. Meanwhile the development of the software and orchestration of the services will be performed according to the plan, and necessary frameworks will be created for other work packages. The exchange of information between development and specification flow will be bidirectional and immediate, so any design flaws or wrong decisions may be identified correctly. This also will allow to define and implement immediately frameworks for other work packages (7, 8 and 9) avoiding expensive delays.

6.2. Alignment with the other Work Packages

WP4 is expected to consolidate the requirements and design contributions from all the other work packages.

The use cases for End User actors here described have now to be aligned with the non-functional requirements expressed in ***D6.2 User Interface Design*** and ***D6.1 Usability Study Report***, which define useful basic design guidelines for the user interfaces, and provide information on typical usage patterns of mathematical libraries.

The global Bibliographic Data format and specific information on the partners collections are addressed in the WP3, namely in ***D3.1 - Report on available collections and metadata***, ***D3.1*** *annexes* ***- Complementary description of metadata schemas***, ***D3.2 EuDML metadata schema - initial version***, and ***D3.2 annexes - Best practices recommendations and XML examples***. Those must be now taken in consideration for the concrete implementation of the Information Architecture.

The requirements for the data processing services and stored data formats internal to those services are reflection of the outcome of the deliverables ***D7.1 A State of the art on Augmenting Metadata Techniques and Technology*** and ***D8.1 Association Analyser Implementation: State of the Art***. Also as the WP7 and WP8 demo deliverables are expected to have significant impact on the revision of the specifications.

6.3. Follow-up within the WP4

Next step within the WP4 is to provide a detailed architecture specification and use cases. That will include:

- Architecture of Services: Detailed description of the EuDML conceptualization of services, including the description of the lower-level services.
- Architecture of Applications: Description of how the services will be supported by the applications; description of the data formats within the services.
- Technological Architecture: Define inter service communication protocols and other communication issues.
- Align the End User use cases with the Deliverables D6.2 and D6.1.

As the system specification developed during the demo deliverable will mature, this content will be aggregated and finally delivered as ***D4.2 EuDML global system functional specification and design***, which will be referred by all development activities within the project.