ECP-2008-DILI-528001

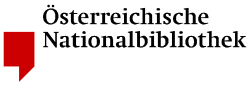
EuropenaConnect

D5.1.1 – Europeana Metadata Registry

Deliverable number/name	*D 5.1.1*
Dissemination level	*PU*
Delivery date	*3/10/2010*
Status	*v1.0*
Author(s)	*Hugo Manguinhas (IST)*

e*Content*plus

This project is funded under the *e*Content*plus* programme,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.



EuropeanaConnect is coordinated by the Austrian National Library

europeana
connect

D5.1.1 Europeana Metadata Registry

Deployment and Documentation

co-funded by the European Union

The project is co-funded by the European Union, through the ***e*Content*plus*** programme

http://ec.europa.eu/econtentplus

Österreichische Nationalbibliothek

EuropeanaConnect is coordinated by the Austrian National Library

Executive Summary

Task 5.1 in EuropeanaConnect is to build a Metadata Registry (EuMDR) for the management and publishing of the range of data models and terms used by the various European institutions that will provide content for Europeana. IST has led the task of developing the EuMDR together with UW.

The primary purpose of the EuMDR is to improve metadata interoperability by allowing the documentation of the data models of the Europeana's providers and the relationships among their attributes; along with the transformations among the various data models.

This document describes the specification, design and implementation of the EuMDR. The remainder of the document continues by introducing the generic concepts behind MDRs, followed by a description of the principal guidelines, the requirements and design for the development of the EuMDR, and finishes with its implementation and deployment.

Table of Contents

1. Introduction

Europeana will integrate multiple services and will reuse descriptive metadata created in multiple contexts and by a large spectrum of different entities or communities, each one with their own data model (schema). Moreover, most of that metadata will be created for purposes other than Europeana and thus must be aligned with Europeana's expectations. This is a problem for interoperability.

In order to achieve interoperability at this level, Europeana needs to find a common understanding of the descriptive metadata being transmitted. This understanding can be achieved by recognizing these data models and the rules modelling the relationships and equivalences between them. By cross-walking between the data models, different systems can recognize others' metadata and use it in an effective way.

The EuMDR – Europeana Metadata Registry is thus a service responsible for managing and publishing the range of data models and terms used by the various European institutions that will provide content for Europeana. The primary purpose of the EuMDR is to improve metadata interoperability by allowing:

- The documentation of the data models of the Europeana's providers and the relationships among their attributes;
- Transformations among the various data models.

In this sense, the EuMDR will also be able to manage the current or future Europeana internal data models and their mappings, namely the actual ESE – Europeana Semantic Elements or EDM – Europeana Data Model. Nevertheless, the cost for achieving interoperability must not come with a higher price. In this sense, the EuMDR is designed to rely on human intervention only where it is essential. This means automating or assisting users in performing their tasks.

The EuMDR will be integrated into the Europeana environment as a system component (ideally as an independent service) which will have as clients other services such as REPOX (detailed in M5.3.2), which this way will be able to use the data transformation scripts provided by the EuMDR to transform the original metadata into the ESE or any other requested schema.

This document describes the specification, design and implementation of the EuMDR. The remainder of the document continues by introducing the generic concepts behind MDRs, followed by a description of the principal guidelines, the requirements and design for the development of the EuMDR, and finishes with its implementation and deployment.

2. About Metadata Registries

An information model is an abstract description of how information is represented in a specific context (a business area, an organization or a community), consisting therefore in the definition of the relevant terms, the relations between them, and the vocabularies or rules to assign values to them. A data model (schema) is the representation of an information model for a specific syntax (XML Schema, SQL-DDL, etc.) and/or storage method (file, database, etc.), which defines the data elements and arrangements between them.

The information needed to describe these models (information model and data model) has been called metadata (in opposition to the term “data”, to refer to the data objects representing the instances of the information entities). Associated to that resulted the concept of **Metadata Registries** (MDR), defined as information systems to manage and publish metadata.

A MDR is a central location (e.g. repository) in an organization where metadata is stored and maintained in a controlled environment. The “registry” part implies that metadata is subject of registration within the Registry. Registration specifies the set of rules, operations, and procedures that apply to an MDR, accomplishing three main goals:

- **Identification**
- **Provenance**
- **Monitoring quality**

Identification is accomplished by assigning a unique identifier (within the registry) to each registered object. **Provenance** addresses the source of the metadata and the object described. **Monitoring quality** ensures that the metadata does the job it is designed to do. The registration and administration functions of an MDR are what separate an MDR from a database of metadata.

A MDR provides a unified view and promotes a common understanding of the information managed in an organization. It assists organizations in the sharing and exchanging of mutually agreed information. It can also promote harmonization, standardization, use, and reuse of the information throughout and between organizations.

The ISO/IEC JTC1 SC32 WG2 develops standards for metadata and related technology[1], namely the ISO/IEC 11179, Information Technology -- Metadata registries (MDR), which is the standard that defines the concepts behind MDR, addressing “the semantics of data”, “the representation of data”, and “the registration of the descriptions of that data”. It specifies the kind and quality of metadata necessary to describe data, and it specifies the management and administration of that metadata in an MDR. It applies to the formulation of data representations, concepts, meanings, and relationships between them to be shared among people and machines, independently of the organization that produces the data.

One particular kind of information that can be registered within an MDR is the rules modelling the relationships and equivalences between the data models. This information is commonly called **mapping information** and is particularly important for cross-walking between two data models.

[1] http://metadata-standards.org/

One way to assure that is to build **data transformation scripts**, by using the mapping information between the two intervenient data models.

A MDR promotes **interoperability** by using a common reference model for the registration of the data model (semantic interoperability) and the context where it should be used (pragmatic interoperability), while registering version information about the data object (dynamic interoperability) and the corresponding relations (conceptual interoperability), whether related to relationships between different versions of the same or different data models.

3. The Europeana Metadata Registry

This section provides an analysis of the information being registered, the principal participants and the processes involved in the EuMDR.

3.1. Metadata

A MDR is responsible for managing and administering Metadata. The notion of metadata described in this document, relates to the information needed to describe **Models**. This comprises the information about the structural organization of all the elements that make up the model, along with the definition of the relevant terms, the relations between them, and the vocabularies or rules to assign values to them.

Associated to this metadata (more accurately called structural metadata) are also other kinds of metadata necessary for the correct administration of the models, called administrative metadata. In the EuMDR, this metadata is divided in two forms, descriptive metadata and authoritative metadata, which are required for respectively, providing textual descriptions of the names and definitions for the elements that compose the models, and the information related to the responsibilities of the different participants in their creation and management. The combination of the structural and administrative information that form the metadata managed within the EuMDR is called an **Administered Model**. On the other hand, all the elements that form the structural organization of a model and are also subject of registration (and thus must also detain metadata associated to them), are called **Administered Elements**.

This document defines two sorts of models: **Data Models** (commonly called schema) which are representations of an information model for a specific syntax (XML Schema, SQL-DDL, etc.) or storage method (file, database, etc.) and **Mapping Models**, which describe the logical relationships between two distinct data models.

3.2. Participants

According to the ISO11179 the principal participants of MDRs are Registration Authorities, Responsible Organizations and Submitting Organizations. To be conformant with the standard, the EuMDR will maintain the same participants. Figure 1 presents an overview of all the participant organizations and their roles in the administration and management of a EuMDR.

A **Submitting Organization** (SO), is an organization responsible for providing models to be registered within the EuMDR in accordance with the procedures prescribed by the Registration Authority. A Submitting Organization is represented by **Submitters**, which are organizational units familiar with or engaged in development and operational environments, responsible for identifying and reporting new models suitable for registration. A particular kind of Submitter is a **Mapper** which detains particular expertise in multiple domains and thus is capable of establishing relationships between them.

A **Responsible Organization** is the subject matter expert for the Metadata, designated to ensure consistence of the Metadata managed by its Submitting Organizations. A Responsible Organization is represented by **Stewards**, which are specific expert points of contact responsible for coordinating the identification, organization, and establishment of registered metadata for use throughout the EuMDR within an assigned functional area.

A **Registration Authority** (RA) is an organization that desires to operate and manage the EuMDR. A Registration Authority is represented by **Registrars**, experts in registration processes,

responsible for facilitating the registration of Metadata and making that Metadata widely accessible and available to the community.

For the specific context of Europeana it is expected to exist only one Registration Authority, the Europeana itself, which will be responsible for assigning a group of Registrars with expertise in the registration of models.

How the assignment of Responsible Organizations will be processed in Europeana is not clear at this time. Which organization will be responsible for any given data model will depend on its context. For example, in the case of MARC21 data model, the Responsible Organization would be the Library of Congress (LoC), but since LoC is not part of the Europeana Consortium, the Responsible Organization within Europeana would be a workgroup composed of experts from each Library using MARC21 as its format. A different case is UNIMARC, since the UNIMARC working group is lead by a consortium member, the National Library of Portugal, which could thus be responsible for assigning the Stewards for the UNIMARC format. Similarly, the National Library of Spain could take the role of Responsible Organization for IBERMarc.

For data providers with specific data models (not shared with other data providers), a workgroup composed of experts in data formats in general could be assigned as Responsible Organization.

Finally, all Europeana data providers can be registered in the EuMDR as Submitting Organizations.

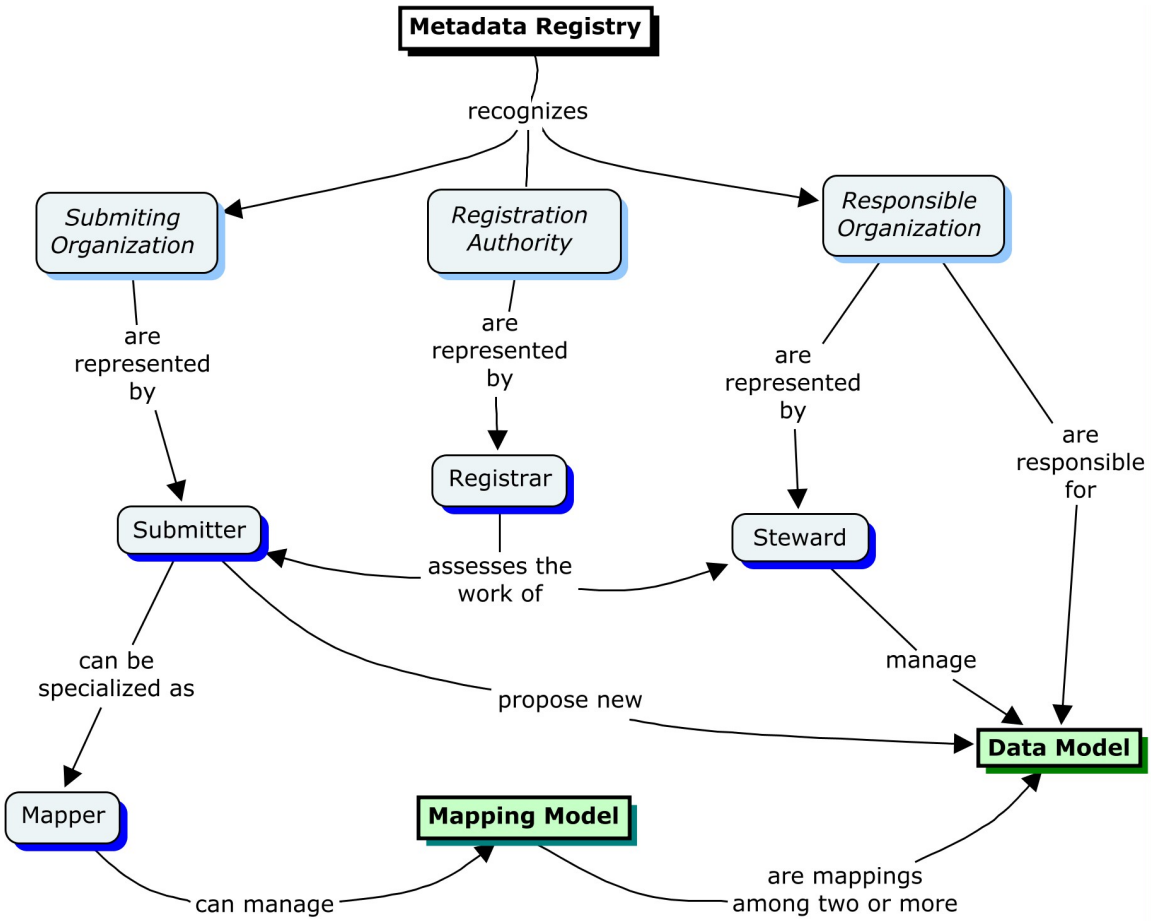


Figure 1- Conceptual map with all the possible Participants and their roles in a typical MDR.

3.3. Registration of a Model

The registration of models in a MDR requires the participation of three Participants. The registration is always initiated by the Submitting Organization, which will be from that moment on the editor of that model (any changes to the model must always be made by the Submitting Organization). On the other hand, it is the responsibility of the Responsible Organization to ensure that the model satisfies the necessary requirements to complete the registration, while the Registration Authority is responsible for assessing the work of the Responsible Organization.

In the EuMDR it is envisioned that each participant in Europeana that uses a different data format should become a Submitting Organization and register the corresponding data model. Stewards will then be appointed to assist the process of registration. Mapping models between the participant's data models and the ESE data model should also be submitted to the EuMDR.

Until a model is effectively registered in the EuMDR, it passes through a set of **registration status levels**. The registration status measures the conformity of the Administered Model with the quality requirements defined for registration. The registration status is classified according to 5 levels (see Table 1 and Figure 3) as defined in ISO11179 series of standards: Incomplete, Candidate, Recorded, Qualified, Standard and Preferred Standard.

Table 1- Description of the registration status levels for a model.

Registration Status Level	Description
Incomplete	A model with the "Incomplete" status shall indicate that the Submitter wishes to make the community aware of the existence of a new Model. This level is defined to allow for a user to have persistence of a model while it is being edited and before it is proposed for "Candidate" status.
Candidate	A model with the "Candidate" status shall indicate that it has been proposed for progression through the registration levels. The Administered Model may not contain all mandatory attribute values.
Recorded	A model with the "Recorded" status shall mean that all mandatory metadata have been completed. A model in the "Recorded" status implies that the model may be shared across domains. The contents of the mandatory metadata attributes may not conform to quality requirements.
Qualified	A model with the "Qualified" status shall mean that the model had a "Recorded" registration status and the Registration Authority has confirmed that the mandatory metadata attributes are complete and conform to applicable quality requirements.

Standard	A model with the “Standard” status indicates that it had a “Qualified” registration status and the Registration Authority confirms that the model is of sufficient quality and of broad interest for use in the community that uses this metadata register.
Preferred Standard[2]	A model with the “Preferred Standard” status indicates that it had a “Standard” registration status and the Registration Authority confirms that the Administered Model is preferred for use within the community that uses this metadata registry.
Retired	A model with the “Retired” status indicates that it had a registration status ranging from “Recorded” to “Preferred Standard” and the Registration Authority confirms that the Administered Model is no longer recommended for use in the community that uses this metadata register and should no longer be used.

[2] To be conformant with ISO11179, the “Preferred Standard” status level was maintained. Nevertheless, whether this status level relates to a preferred standard within the whole MDR, or it is dependent to the context which it is used, is still an issue to be addressed.

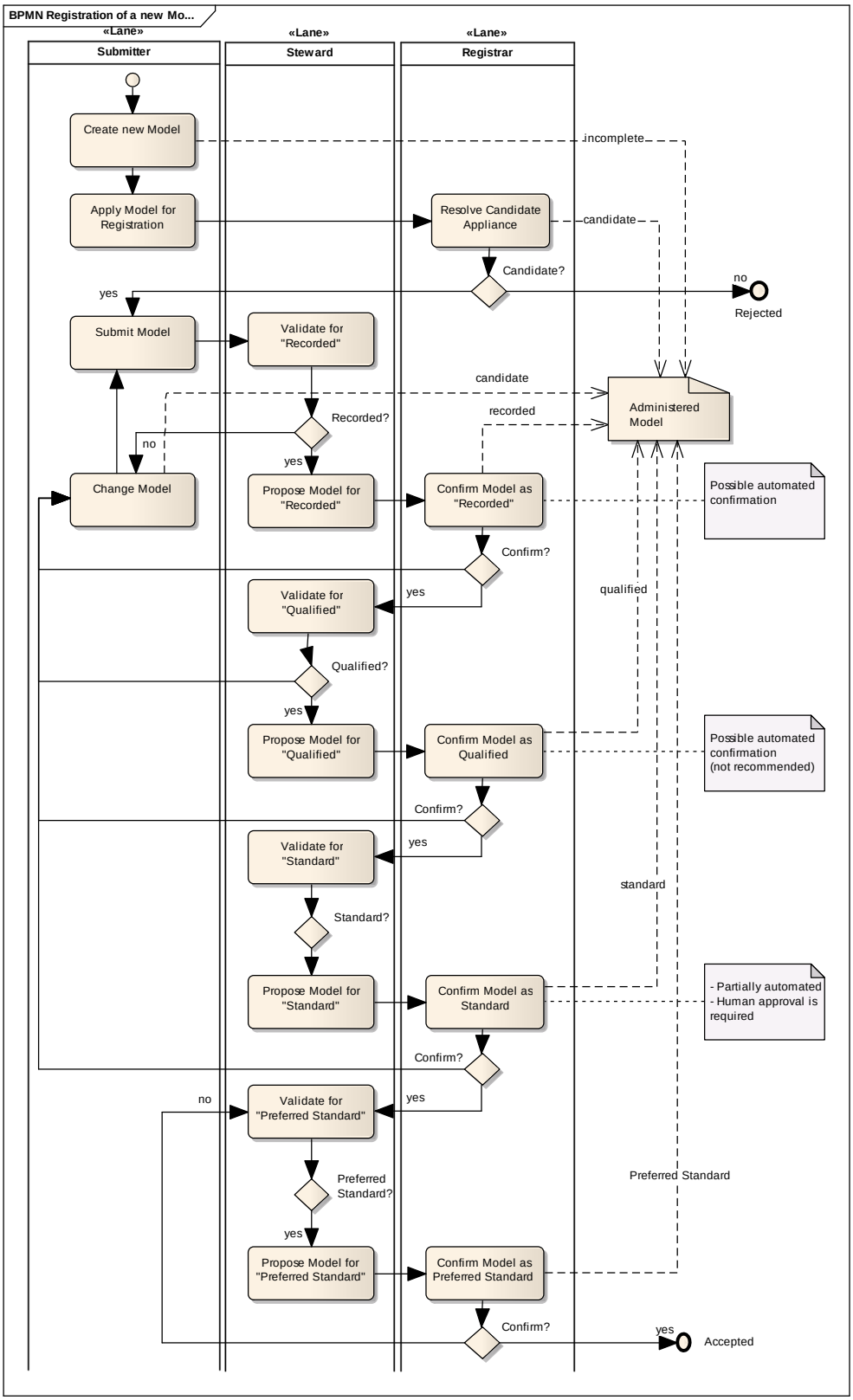


Figure 2 - Overview of the Registration Process in BPMN notation.

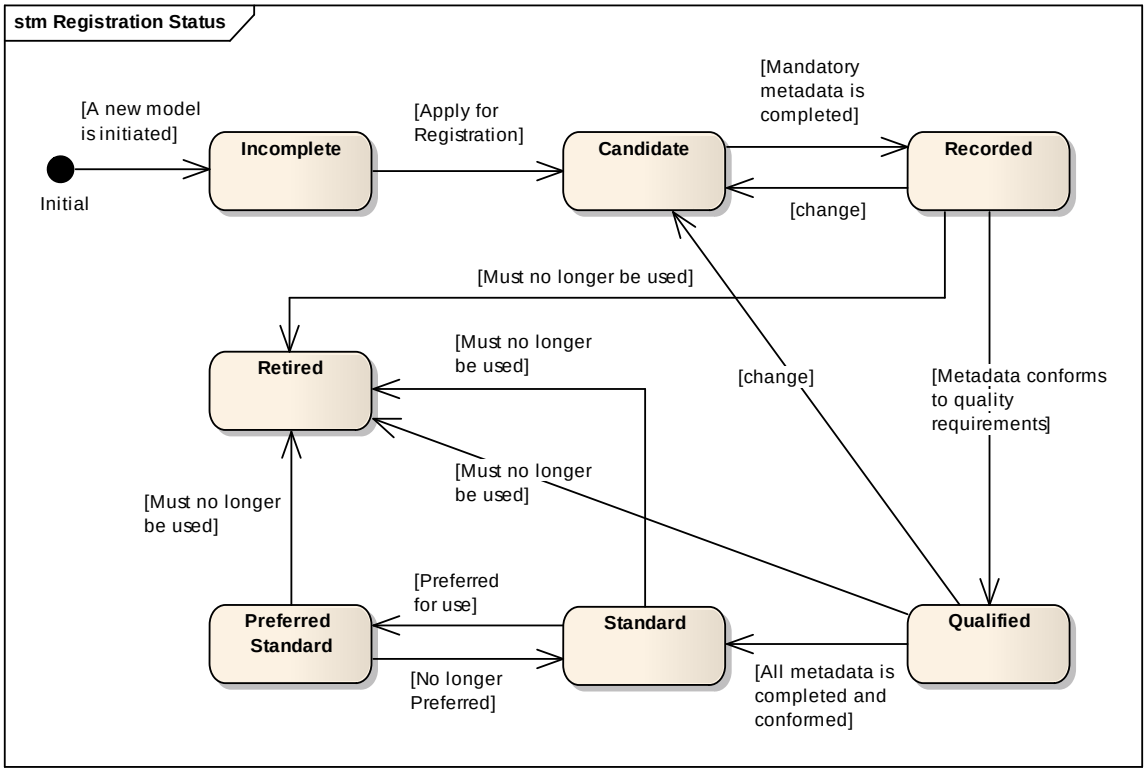


Figure 3 – State diagram showing the transitions between the registration status levels.

Any progression in the level of registration status must be confirmed by the Registration Authority. In the event that a model is not approved for progression by the Registration Authority, it shall remain at the same registration status level. All confirmations must be preceded by a validation of the proposal for progression to ensure that the model satisfies all the requirements for progression. This validation must be performed by the Responsible Organization.

To reduce the overhead over both the Responsible Organization and the Registration Authority the validations and confirmations should be automated whenever possible. Nevertheless, confirmations for progression in levels higher than recorded are not advised.

After a rejected proposal for progression, the Submitting Organization may change the model to meet the necessary requirements for achieving the next registration status level. Whenever a model with the registration status higher than “Candidate” is changed, a new version of the model is created starting at the level of “Candidate”, while the older is maintained unchanged.

3.4. Retirement of a Model

A model can be retired at any moment. The retirement of an Administered Model must only be proposed by the Submitting Organization responsible for the model, and must be subject of approval by the Registration Authority. Only data models that have reached the registration status level o “Recorded” and have no retired mapping model associated to them can be retired. Thus before retiring a data model, the Steward must make sure that all mapping models associated to it are also retired. Figure 4, presents an overview of the Retirement Process.

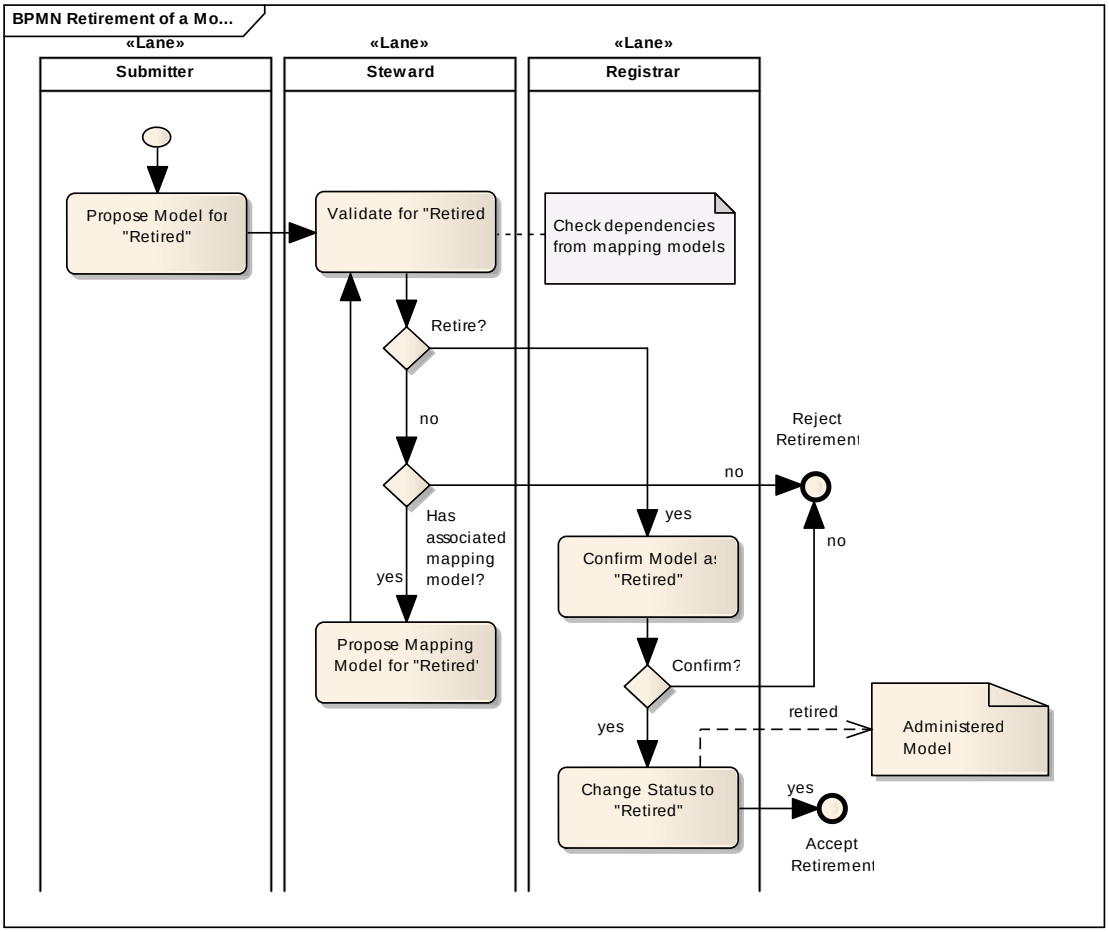


Figure 4- Overview of the Retirement Process in BPMN notation.

3.5. Superseding a Model

Successor models can be defined to supersede existing ones. The superseding of a model indicates that the Registration Authority has determined that the model is no longer recommended for use in the community, and a successor model is preferred for use instead. It is possible to supersede both data and mapping models.

The preferred model should include a reference to the replacement model when appropriate. The superseding of a model must not interfere with their registration (management and even progression in their registration status), they must continue to be managed and used by the participants in the EuMDR. On the other hand, successor models are to be managed as if they had no correspondence with their superseding models, having their own identification and metadata. A model is only recognized as superseded when exists a superseding model, with the minimum registration status of “Qualified”, registered in the EuMDR.

When superseding a data model, the new data model must not interfere with the mapping models that use the older model as source or target. The successor model must start with no mapping model defined; nevertheless, automated processes may be defined to automatically derive new mappings for the successor model.

To reduce the model registration effort, automated processes may be defined to create the successor model based on the older one, which would then be changed to reflect the newer version. This applies to both data models and mapping models.

3.6. Versioning

When a model is changed during the registration process, it must restart registration from the registration status level of “Candidate” (see Figure 3). However, since the model could be in use at the moment of change (when the registration status is higher than “Qualified”), both the old and new models must be accessible at the same moment in time, which means that the system must maintain separate versions of models during registration. The change in the model must only become effective when it reaches the same registration status level of the previous version. Models at the registration status level of “Incomplete” and “Candidate” shall not be maintained under version control.

In this sense, the versioning of a model must always be performed when a model is changed during the process of registration. It is important to note that versioning is not the same as superseding, the first is a result of correction and completion, and the second is a result of improvement (new functional requirements).

In order to avoid that all previous work performed during registration, be redone due to the change in model, the system must be able to track the changes between the two versions of the model.

3.7. Mapping Models and Mapping Scripts

Mapping Models are registered in the same way as Data Models. Nevertheless, Mapping Models can only be proposed for registration status level of “Candidate” when both the source and target data models have achieved the registration status level of “Qualified”. This is required since registration status levels lower than “Qualified” mean that the models are still in progress of registration.

Only when the Mapping Models reach a level of registration status of “Recorded”, that **Mapping Scripts** can be generated. A Mapping Script is a technological implementation of a particular mapping model (e.g. a XSLT script for a XML based data model). There can be several different mapping scripts for the same mapping model, which can be available for use by other systems to perform data transformations.

Within the context of Europeana, the REPOX system will use the Mapping Scripts provided by the EuMDR to perform transformations from each original data format into the ESE profile.

3.8. Importing Models

It is expected that a significant part of all the institutions participating in Europeana already have data models encoded for particular syntaxes (XMLSchema or DTD for XML documents, etc). When that is the case, those data models could be automatically registered in the EuMDR. Even though these data models would need to be subject of registration through its registration levels as in manual registration, it would dramatically reduce the effort of creating the data model from the beginning. This “import” functionality should be available for any Submitter as an alternative for the manual registration of a data model.

When a data model is not available for a particular data file, it should be possible to deduce the data models from its structure. This is particularly relevant for XML files, since it already exist software that can generate XML Schema files from analyzing their structure.

External Systems should also be able to import data models. This is particular important for the REPOX system, when it encounters a new data files data encoded with a particular data model that he is not aware. Nevertheless, for all data models imported through External Systems, a Submitter and a Steward must be appointed to supervise its registration.

3.9. Efficiency of Registration

The cost for Europeana and its participating institutions to completely register a Model in the EuMDR must be very low. In this sense, human interaction with the EuMDR must be reduced to the minimum required for an effective registration of Models.

Having this in mind, tasks that can be fully automated should be replaced by automatic functions. Examples of these tasks are:

- Checking if the Administered Model is conformant with the requirements defined for each Registration Status level;
- Allow automatic approval by the Registration Authority for Registration Status levels lower than “Qualified”;

Other tasks could be partially automated, by assisting the user on its work. Examples are:

- Testing a generated Mapping Script against two data files (source and target data files);
- Create a Mapping Model from the source and target data models by evaluating the rules modeling the relationships and equivalencies between the elements that compose each model;
- Create a superseding data model from an existing Administered Model.

Although a significant part of the tasks can be partially or fully automated, each participant in the registration must maintain in control of the Registration. This means that the participants should be able to interfere in the registration whenever they find it necessary.

4. Requirements

This Section presents both the functional and non-functional requirements for the EuMDR service. The functional requirements are described using “use case diagrams” as defined by the UML – Unified Modeling Language[3].

4.1. Non-Functional Requirements

Non-functional requirements define important constraints of the system. They describe the “how”, “when” and “where” of the system, that do not depend of the use cases but of the businesses and technical constraints.

R-1. Operational Requirements

R-1.1. The system should be as portable as possible, minimizing any kind of platform-specific dependencies.

R-1.2. The system should provide simple service interfaces in web service form, preferably according to the Representational State Transfer (REST) principles.

R-1.3. The system must minimize and ease the work of all its participants, whether by completely automated means or semi-automated (assisted) when automation is not possible.

R-2. Interface Requirements

R-2.1. The administrative interface must have a short learning curve and non technical terms, to be used by non technical staff at institutions using the EuMDR.

R-2.2. The user interface must provide all the necessary information to ease user interaction with the system.

R-2.3. The user interface for creating mapping models must provide a practical and preferably visual way for users to associate metadata equivalents so that the creation of mapping scripts is quicker and less error-prone.

The user interface must only offer access to both information and functionality for users with the necessary credentials.

4.2. Actors

In the context of the EuMDR were identified 6 classes of actors interacting with the System: the Read-Only user, Submitter, Mapper, Steward, Registrar and External System (see Figure 5). Two additional classes were identified from the 4 classes of actors defined in the ISO11179 series of standards, which correspond to the Mapper and External System. The Registered User shown in Figure 5, is not a concrete user of the system, it is shown only for the purpose of simplification. The same person can play more than one role in the system.

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

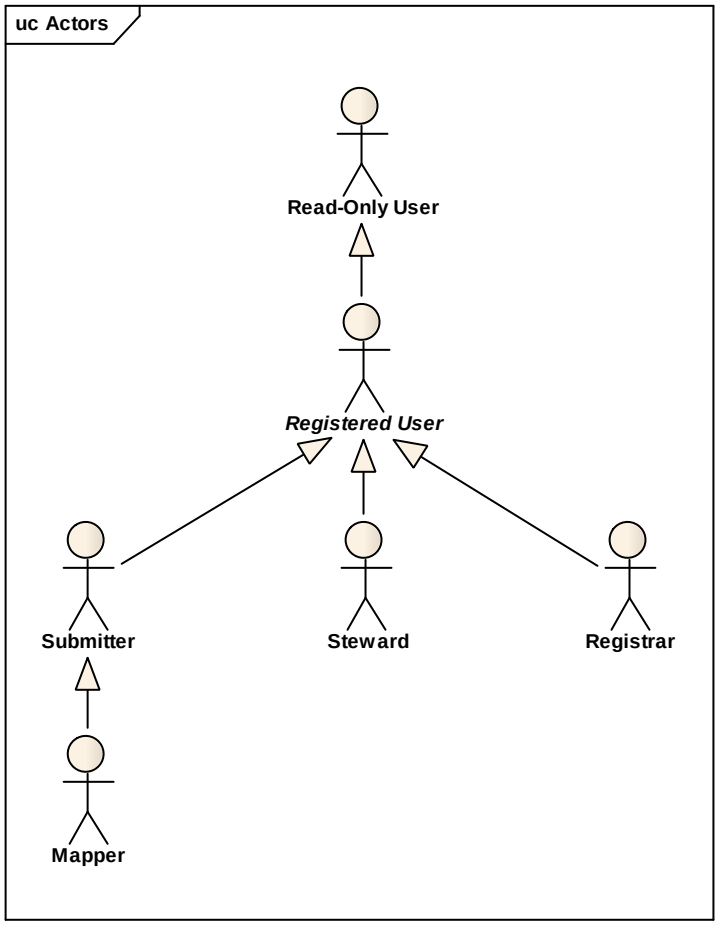


Figure 5 – Overview of the actors of the System.

4.2.1. Read-Only User

A Read-Only user is an organizational unit or individual that is approved by the Registration Authority to obtain and explore the contents of the EuMDR. A read-only user has access to the contents in the Metadata Register, but is not permitted to submit, alter, or delete contents. The policies for approving read-only users are established by the Registration Authority.

4.2.2. Submitter

A Submitter is an organizational unit within a Submitting Organization, familiar with or engaged in development and operational environments. Submitters maintain current Administered Items and are engaged to identify, describe and submit new Administered Items following the registration requirements. The Submitter can be viewed as a contact for the Submitting Organization, which may utilize any number of Submitters. Each Administered Item is associated with only one submitter.

4.2.3. Mapper

A Mapper is a submitter with particular expertise in multiple context environments, which is capable of establishing relationships between Administered Items in two different contexts.

4.2.4. Steward

A Steward is an organizational unit within a Responsible Organization, responsible for the accuracy, reliability, and currency of descriptive metadata for Administered Items at a registration status level of “Qualified” or above within an assigned area. Stewards should be responsible for

metadata within specific areas and may have responsibilities that cut across multiple areas. The Steward can be viewed as a contact for the Responsible Organization, which may utilize any number of stewards. Each Administered Item is associated with only one Steward.

4.2.5. Registrar

A Registrar is an organizational unit within the Registration Authority, expert in registration processes, responsible for facilitating the registration of models and making those models widely accessible and available to the community. The Registrar may be viewed as the contact for the Registration Authority. The Registration Authority should appoint the Registrar as its contact, and may have one or more.

4.3. Use Cases

The following subsections describe the main use cases of the EuMDR system.

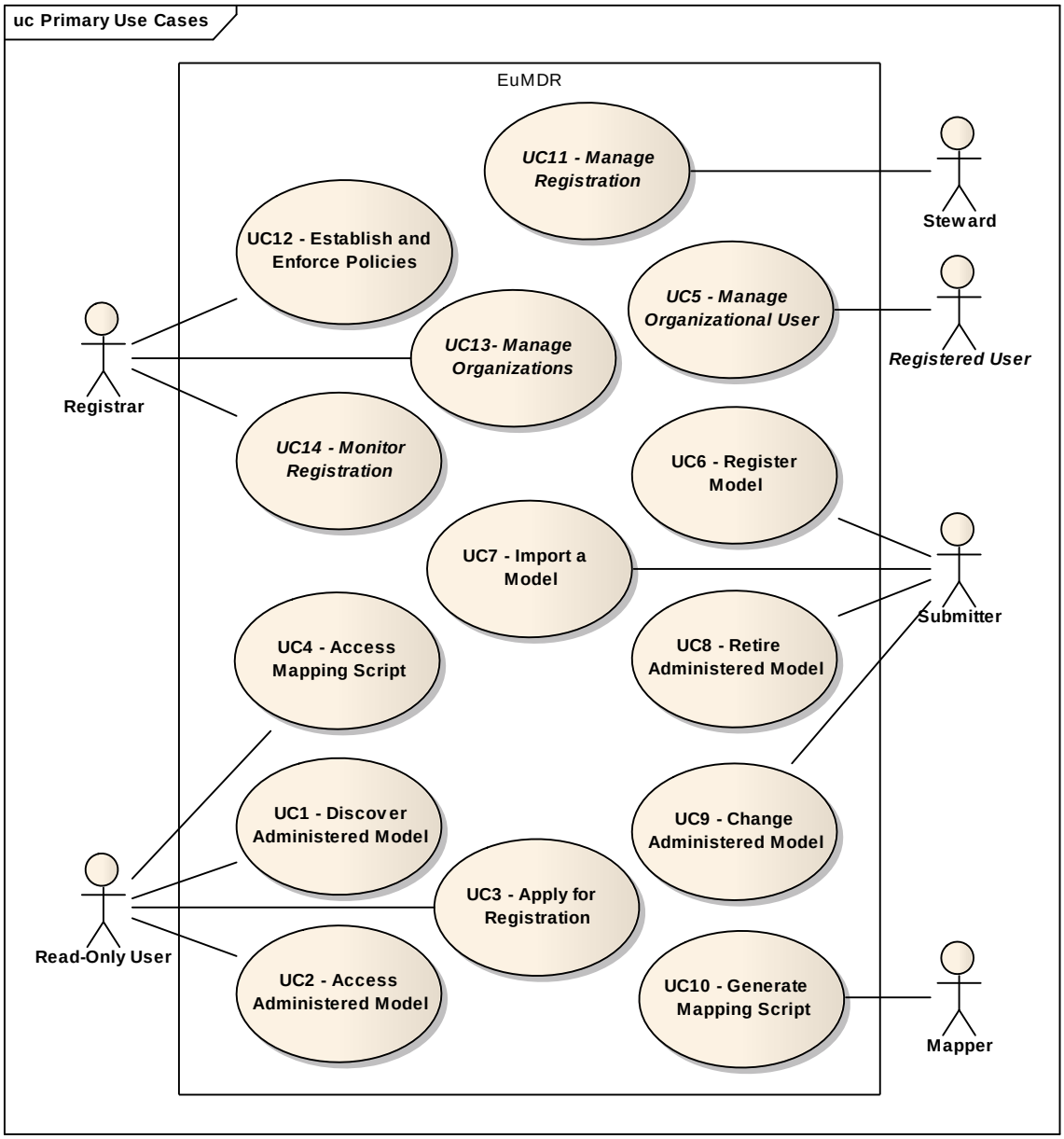


Figure 6 - Overview of the use cases defined for the EuMDR.

Use Case	**UC1 - Discover Administered Model**
Description	A Read-Only user searches or browses the EuMDR for a particular Administered Model. Searching is made through a set of search parameters that applies to both data and mapping models. Browsing is made through various indexes (e.g. using its relationship with an organizational entity or by a particular “classification” of the model, etc).
Primary Actor	Read-Only User
Basic Flow	1. Fill a search form with a set of parameters or chooses a particular index to browse for a Model; 2. Choose an Administered Model to access; 3. Include “Access Administered Model”.

Use Case	**UC2 - Access Administered Model**
Description	After selecting an Administered Model, a Read-Only user accesses all the administrative metadata associated to it, and also all the Administered Elements that compose the model. When accessing a Mapping Model, the user may access the source and target Data Models and also the generated Mapping Scripts.
Primary Actor	Read-Only User
Preconditions	An Administered Model must have been selected (via searching or browsing).
Basic Flow	1. The user asks for access to the Administered Model; a. If granted, i. The user accesses the administrative metadata associated to the Administered Model and also its component elements; ii. If is a Mapping Model, the user may access all the Mapping Scripts generated for this model. b. Otherwise, present a restrict access notification.

Use Case	**UC3 - Apply for Registration**
Description	A Read-Only user applies for registration as a Submitting or Responsible Organization. This application will be resolved by the Administration Authority.
Primary Actor	Read-Only User

Basic Flow	1. The user fills an application form with the information about the Organization he belongs, along with its contact information and the type of relationship with the EuMDR; 2. Send the form.

Use Case	**UC4 - Access Mapping Script**
Description	After selecting a Mapping Model the user may access a particular Mapping Script for a particular technology.
Primary Actor	Read-Only User (played by a person or system)
Preconditions	A Mapping Model must have been selected.
Basic Flow	1. The user asks for access to the Mapping Script; 2. The user accesses the Mapping Script

Use Case	**UC5 - Manage Organizational User**

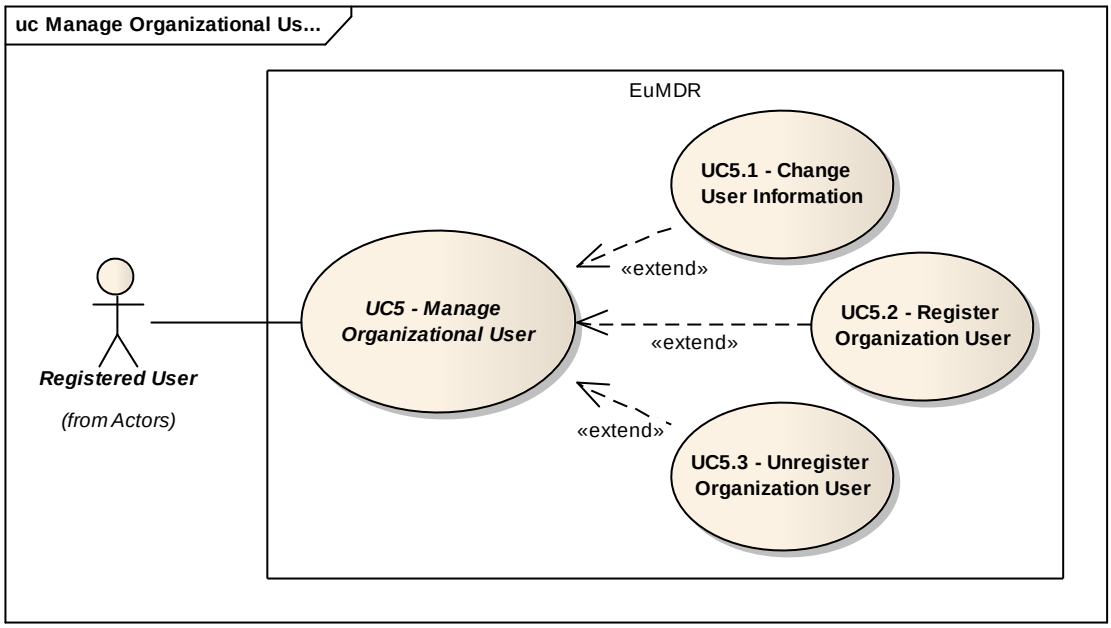


Figure 7 - Use case refinement for "Manage Organizational User".

Description	Manage the users from a given Organization.
Primary Actor	Registered User (note: not a concrete user).
Use Case	**UC5.1 - Change User Information**
Description	A Registered User changes its contact information.

Primary Actor	Registered User.
Preconditions	The user can only change its information.
Basic Flow	1. Change contact information. 2. Submit new information to the Registry.

Use Case	**UC5.2 - Register Organizational User**
Description	A Registered User may register another user for the same Organization. The new user will inherit all access privileges for all models for witch its Organization is responsible.
Primary Actor	Registered User.
Basic Flow	1. Fill the user contact information. 2. Create new user in the Registry. 3. Send a notification to the new user.

Use Case	**UC5.3 - Unregister Organizational User**
Description	A Registered User may unregister other users of the same Organization.
Primary Actor	Registered User.
Basic Flow	1. Select the Registered User to unregister. 2. Unregister user in the Registry.

Use Case	**UC6 - Register Model**
Description	A Submitter identifies a new Model appropriate for registration in the EuMDR and proposes as candidate for registration.
Primary Actor	Submitter (played by a person or system)
Basic Flow	1. Create the new Model in the Registry which becomes a new Administered Model with the registration status level as “Incomplete”. 2. Document the new Administered Model. 3. Propose the new Administered Model for the registration status level of “Candidate”. a. If confirmed and the new Administered Model is complete and conformant, namely that all mandatory metadata is completely

	documented, submit the new Administered Model for Registration. b. If confirmed and the new Administered Model still needs to be more documented (include “Change Administered Model”) c. If rejected, end registration process.

Use Case	**UC7 - Import Model**
Description	New Administered Models can be imported from existing data model implementations (e.g. DTD, XMLSchema, etc. for XML based data models). Existing Administered Model may also be updated through import.
Primary Actor	Submitter (played by a person or system)
Preconditions	A data model implemented using a particular technology must be provided.
Basic Flow	1. The data model is searched in the EuMDR. a. If it is a new Model, propose the new model for registration (include “Register Model”). b. Otherwise, update the existing Administered Model with the new model (include “Change Administered Model”).

Use Case	**UC8 - Retire Administered Model**
Description	A submitter is responsible for notifying the Registration Authority when Administered Models are no longer to be subject of registration, by proposing them for retirement.
Primary Actor	Submitter
Preconditions	Submitter must have access to the Administered Model and the registration status must be higher or equal to “Recorded”. Note that only Mappers can retire Mapping Models.
Basic Flow	1. Check access and registration status of the Administered Model. 2. Propose the Administered Model for retirement.

Use Case	UC9 - Change Administered Model
Description	The Administered Model can be changed at any time by the Submitter to satisfy a requirement defined by the Registration Authority. Nevertheless, every time it is changed, a new version must be created with the registration status level of “Candidate”. This use case can be automated in the case of an import (see “Import Model”).
Primary Actor	Submitter (played by a person or system)
Preconditions	Administered Model registration status must be higher or equal to “Candidate”. Submitter must have access to the Administered Model. Note that only Mappers can change Mapping Models.
Basic Flow	1. Check access for the Administered Model. 2. Create a new version of the Administered Model starting at registration status of “Candidate”. 3. Document the Administered Model, ensuring its completeness and conformity, namely that the mandatory metadata information is completely documented. a. Change the Structural Metadata that composes the Model. b. Change the Administrative Metadata associated to the Model. 4. Submit the Administered Model.

Use Case	UC10 – Generate Mapping Script
Description	Mappers can generate Mapping Scripts for a particular technology. The generated Mapping Scripts are published for use by external systems to perform data transformations. Before publishing, the Mapper must test the Mapping Script to assess its alignment with the source Mapping Model.
Primary Actor	Mapper
Preconditions	Mapping Model registration status must be higher or equal to “Recorded”.
Basic Flow	1. Check access for the Mapping Model; 2. Choose a technology for the Mapping Script; 3. Generate Mapping Script; 4. Assess its alignment with the Mapping Model; 5. If true, publish Mapping Script. 6. Otherwise, report bug for the Administration Authority.

Use Case	UC11 - Manage Registration
Description	A Steward is responsible for assisting in the progression and retirement of Administered Models.
Primary Actor	Steward

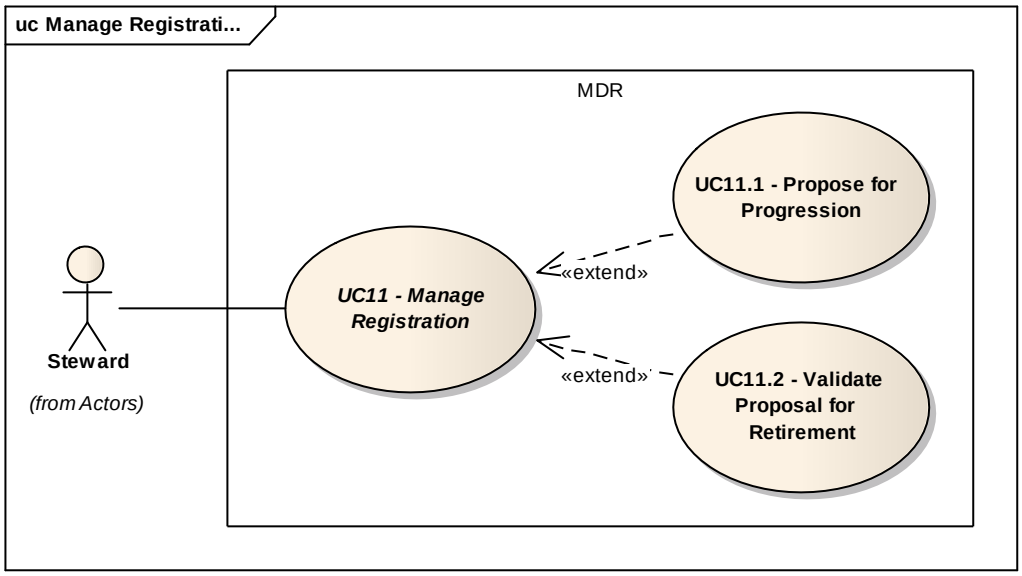


Figure 8 - Use case refinement for "Manage Registration".

Use Case	UC11.1 –Propose for Progression
Description	A Steward is responsible for validating an Administered Model against the requirements for each registration status (Recorded, Qualified and Standard) and to propose to the Registration Authority for achieving a higher registration status.
Primary Actor	Steward
Basic Flow	1. Receives notification of submission to registration. 2. While the Administered Model does not achieve the highest registration status. a. Ensures that the Administered Model at a particular registration status level fills the requirements for achieving a higher status. i. To "Recorded", ensure the completeness of the mandatory metadata. ii. To "Qualified", ensure the conformity of all metadata, while checking if they conform to the established formats. Preventing and resolving conflicts among the structured metadata of the Administered Model. iii. To "Standard", ensure that the Administered Model is properly documented and all the metadata is completed (not only the

	mandatory metadata). iv. To “Preferred Standard”, ensure the model satisfies all the requirements to be considered preferred for use (e.g. check if no other standard is also defined as preferred). b. Proposes a higher registration status for the Administered Model to the Registration Authority. i. If confirmed, continue to propose for a higher registration status. ii. If declined, notify the Submitter of the result, and explain which requirements were not met.

Use Case	**UC11.2 –Validate Proposal for Retirement**
Description	A Steward is responsible for validating retirement proposals.
Primary Actor	Steward
Precondition	Administered Model registration status must be higher than “Recorded”.
Basic Flow	1. Receives proposal for retirement. 2. Ensures that the Administered Model has no dependencies to mapping models. 3. If dependencies exist propose all dependent mapping models for retirement (Include “Retire Administered Model”). 4. Ask for confirmation from the Registration Authority.

Use Case	**UC12 - Establish and Enforce Policies**
Description	It’s the responsibility of each Registration Authority to establish its own policies, procedures and formats for populating and using the EuMDR (while maintaining the conformity with this specification). A brief description of a set of policies, procedures or formats that a Registration Authority may establish, are given below: • Define the policies for avoiding duplicate Administered Model submissions to the EuMDR. • Define the degree of acceptance for resolving proposals for achieving higher levels of registration status (e.g. relax the acceptance of proposals from Stewards for achieving Recorded registration status, by relying on their prior evaluation); • Define the policies and formats used to effect harmonization of data across the EuMDR for the participating Organizations. • Establish the criteria for registration eligibility of new Organizational Entities.

Primary Actor	Registrar
Basic Flow	1. A form is presented with a predefined set of policies, procedures and formats supported by the EuMDR. 2. Policies, procedures and formats are selected. 3. Policies, procedures and formats are established and enforced.

Use Case	**UC13 - Manage Organizations**
Description	A Registration Authority is responsible for registering and managing Organizations, whether by appointing new contacts or by accepting proposals for registration. Only the Registrar may register new Organizations.
Primary Actor	Registrar

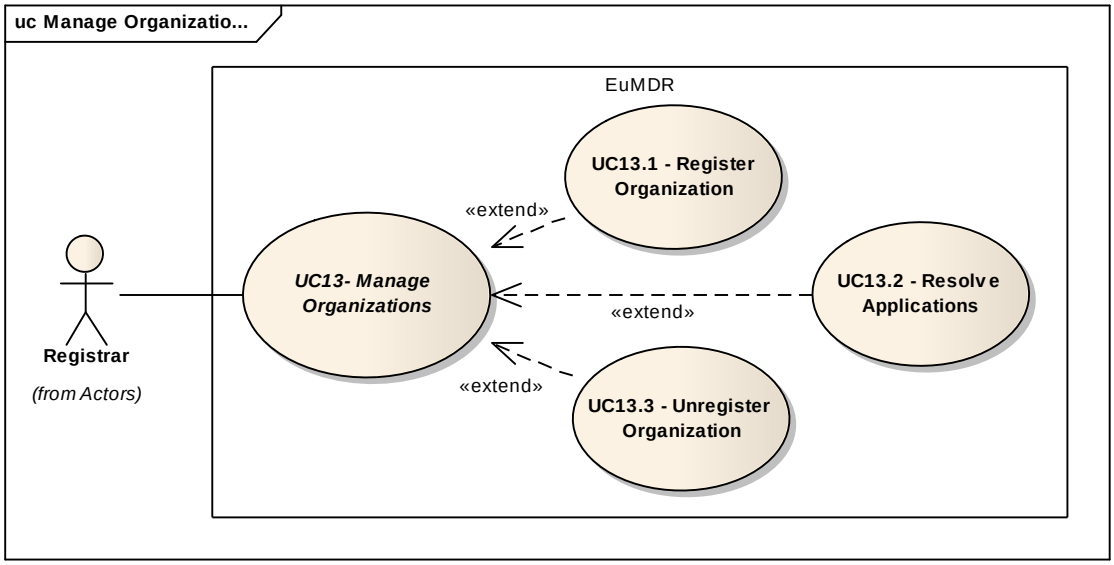


Figure 9 - Use case refinement for "Manage Organizations".

Use Case	**UC13.1 - Register Organization**
Description	The Registration Authority registers a new Organization in the EuMDR. This use case should be implemented using a user interface. However, in the context of Europeana, it can be implemented by interoperating with the Europeana Customer Relationship Management (CRM) system. The interoperation between the two systems is still an issue to be addressed.
Primary Actor	Registrar
Basic Flow	1. Fill the information of the Organization. 2. Submit to the Registry; 3. Register a new user for the Organization (Include "Register Organization User");

Use Case	**UC13.2 - Resolve Applications**
Description	The Registration Authority receives a proposal for the registration of a new Organization.
Primary Actor	Registrar
Basic Flow	1. Check the illegibility of the new Organization; 2. If illegible, accept registration (include "Register Organization") and register the new user (include "Register Organizational User");

Use Case	**UC13.3 - Unregister Organization**
Description	The Registration Authority unregisters an existing Organization from the EuMDR.
Primary Actor	Registrar
Basic Flow	1. Select the Organization to unregister. 2. Remove from the EuMDR.

Use Case	UC14 - Monitor Registration

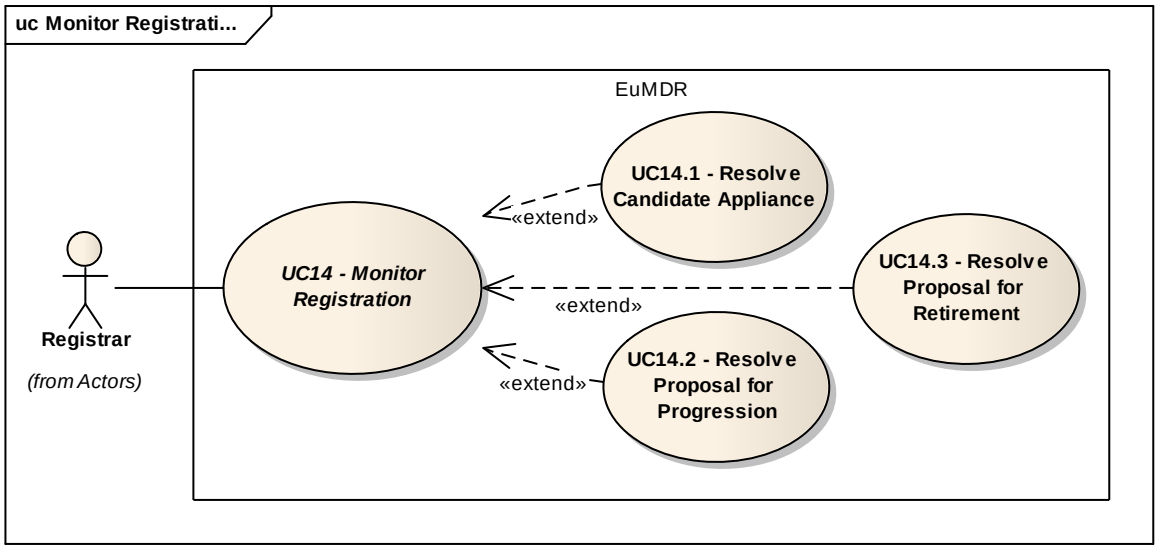


Figure 10 - Use case refinement for "Monitor Registration".

Description	A Registrar is responsible for assisting in the progression of Administered Items through the registration status levels and ensuring that each Administered Item follows the established minimum quality requirements for achieving the pretended registration status level.
Primary Actor	Registrar

Use Case	UC14.1 - Resolve 'Candidate' Appliance
Description	A Registrar receives and resolves an application for registration of Model in the Registry.
Precondition	Administered Item registration status must be 'Incomplete'.
Primary Actor	Registrar
Basic Flow	1. Ensure that the new model does not exist in the Registry; 2. If it exists reject appliance 3. If it doesn't exist, accept the model as Administered Model a. Assign an identifier or verify the proposed identifier. b. Assign a Responsible Organization. c. Assign a Submitting Organization when model comes as an import from an External System.

Use Case	**UC14.2 - Resolve Proposal for Progression**
Description	A Registrar receives and resolves proposals from Stewards for Administered Models to achieve a higher Registration Status.
Primary Actor	Registrar
Basic Flow	1. Confirms that the Administered Model at a particular registration status level fills the requirements for achieving a higher status: a. For achieving the “Recorded” status, confirms the completeness of the mandatory metadata. b. For achieving the “Qualified” status, confirms the conformity of all metadata, while checking if they conform to the established formats. c. For achieving the “Standard” status, confirms that the Administered Model is properly documented and all the metadata is completed and validated. d. For achieving the “Preferred Standard” status, confirms that the Administered Model is eligible for preferred status. 2. If confirmed, change registration status level to a higher level and send notification to the Steward. 3. If declined, maintain registration status and send notification to the Submitter.

Use Case	**UC14.3 - Resolve Proposal for Retirement**
Description	A Registrar receives and resolves proposals from Stewards for Administered Models to be retired.
Primary Actor	Registrar
Basic Flow	1. Confirms that the Administered Model at a particular registration status level has no dependencies to mapping models and can be retired. 2. If confirmed, change administration status to “Retired” and send notification to the Submitter. 3. If declined, maintain administration status and send notification to the Submitter.

5. Design of the EuMDR

This section describes the software architecture and design of the EuMDR service. The design is a consequence of the requirements, both functional and non-functional, which were identified in the previous section. It starts by presenting an overview of the internal system architecture, detailed description of the component architecture, and the interfaces and data models used in each component.

5.1. System Architecture

The EuMDR System Architecture is composed of three internal subsystems (see Figure 11), the EuMDR User Interface, the EuMDR Service, and a Repository. The EuMDR User Interface was separated from the EuMDR Service since the management of data models and mapping models may require customized user interfaces for complex data models such as bibliographic formats like MARC. This way the EuMDR User Interface may be designed, tested and improved (whether in functionality or with better user interaction) separately from the EuMDR Service, which needs to be more stable. The following tables explain in detail each subsystem and its dependencies with internal subsystems and other Europeana systems.

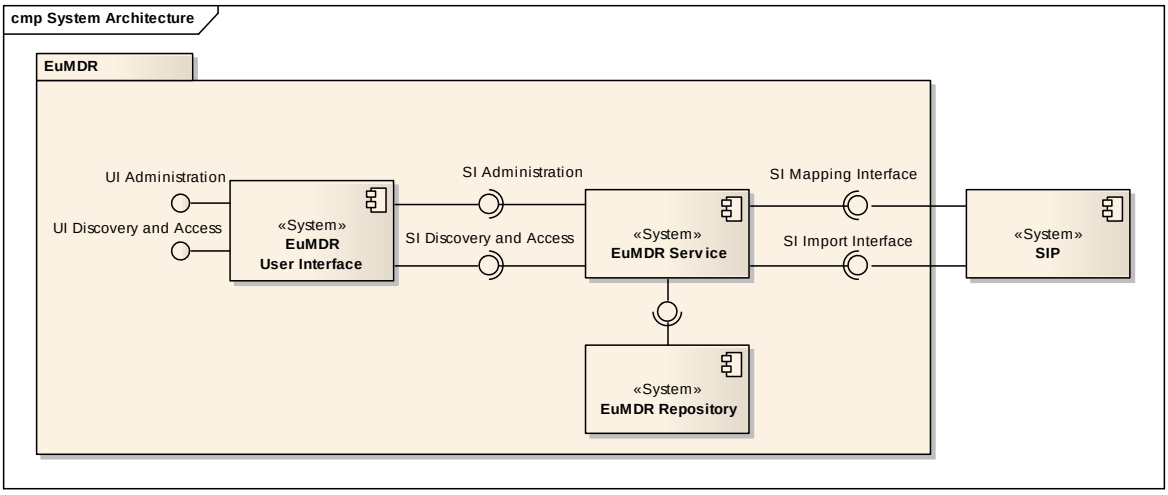


Figure 11 - Overview of the system architecture for the EuMDR.

Subsystem Description		
Name	EuMDR User Interface	
Description	Responsible for supporting all processes that require user interaction with EuMDR, mainly for management and monitoring of the registration process, management of organizations, and discovery and access of administered models. Any interaction with the service interfaces will use the current user credentials, and it is the responsibility of the EuMDR Service to check them.	
Dependencies	SI Administration	Depends on the SI Administration for management.
	SI Discovery and Access	Depends on the SI Discovery and Access, for searching the EuMDR for models and elements, and accessing their information.

Subsystem Description		
Name	EuMDR Service	
Description	Responsible for supporting all the EuMDR processes that require no user interaction.	
Dependencies	EuMDR Repository	Depends on a Repository for storing administered models, the associated metadata and artefacts that are imported or produced by the EuMDR.

Subsystem Description	
Name	EuMDR Repository
Description	Responsible for storing all administered models, the associated metadata and also for storing artefacts that are imported or produced by the EuMDR (e.g. mapping scripts generated from the mapping models, XMLSchemas, DTDs). It will be divided into 3 different logical repositories according to the nature of the information that will be stored. References between information entities that are shared by different repositories will be resolved using URIs.

5.2. Component Architecture

The component architecture describes the component parts that make up the system(s). As described in the previous section the system is decomposed in three other systems, the EuMDR User Interface, the EuMDR Service and the EuMDR Repository. The MDR User Interface is composed of three components, the Administration UI, Data Model UI, and Mapping Model UI; while the MDR Service is composed of four components, Service Support, Data Model Support, Mapping Model Support, and Access Control and Registration Manager; and the EuMDR Repository composed of only one component, the Repository, Figure 12, presents an overview of the components for each subsystem and their dependencies with each other, while the following tables describe in detail each of these components.

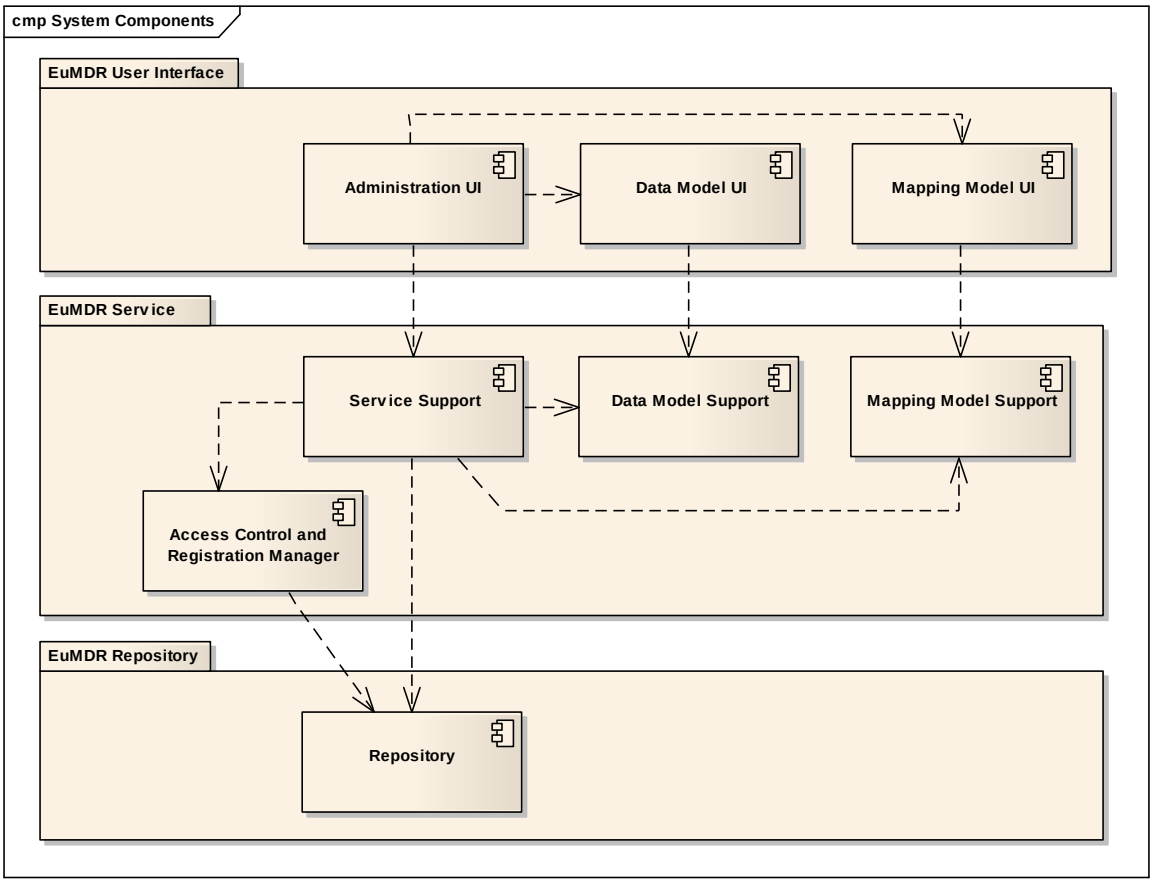


Figure 12 - Overview of the component architecture for EuMDR.

Component Description	
Name	Administration UI
Description	Access restricted user interface for managing the registration process of data and mapping models. It depends on both the Data Model UI and Mapping Model UI for the visualization and editing of respectively data and mapping models.
Owner System	EuMDR User Interface
Component Description	

Name	Data Model UI
Description	Access restricted user interface for managing all the information related to data models. This interface may be customized for particularly complex data model with special functional requirements.
Owner System	EuMDR User Interface

Component Description	
Name	Mapping Model UI
Description	The same as the Data Model UI, but for managing mapping models. Mapping models require different user interfaces than data models, since they manage relations between two data models. Customized mapping interfaces may also be designed for particular complex data models.
Owner System	EuMDR User Interface

Component Description	
Name	Service Support
Description	Responsible for supporting all EuMDR Service' processes and delegating to other components particular tasks.
Owner System	EuMDR Service

Component Description	
Name	Access Control and Registration Manager
Description	Responsible for managing all registration related information and checking access for users.
Owner System	EuMDR Service

Component Description	
Name	Data Model Support
Description	Component responsible for supporting the data model's related functionality: • Translation from particular technological implementations of data models (e.g. XMLSchema, DTD) into an internal EuMDR representation. • Validation of data models for completeness and consistency.
Owner System	EuMDR Service

Component Description	
Name	Mapping Model Support
Description	Component responsible for supporting mapping related functionality: • Interpreting the information contained within a mapping model and compiling it into a specific technological implementation (mapping script). • Validation of mapping models for completeness and consistency. • Decompiling a mapping script into mapping model. • Identifying new mappings from the evaluation of both the source and target data models and/or existing mapping models.
Owner System	EuMDR Service

Component Description	
Name	Repository
Description	Repository responsible for storing: • The information related to users, organizations, data and mapping models; • The artefacts that are imported or produced by the EuMDR (e.g. mapping scripts generated from the mapping models, XMLschemas, DTDs).
Owner System	EuMDR Repository

5.3. Service Interfaces

The following tables describe the Service Interfaces designed for interaction between the EuMDR system components and also for external integration with the REPOX system.

Service Interface Description	
Name	Discovery and Access Interface
Description	Interface for discovery and access of model information. The information is filtered according to each user access restrictions.
Operations	• Search for a particular data model given a set of search parameters. • Search for a mapping model between a source and target data models. One of the source or target data models may be absent. • Access both data and mapping model information.

Service Interface Description	
Name	Administration Interface
Description	Interface for managing organizations, registered users and administered models.
Operations	• Manage Organizations • Manage Registered Users • Manage Administered Models

Service Interface Description	
Name	Import Interface
Description	Provides access to import functionality.
Operations	• Import a new data model by interpretation of a schema. • Import a new data model by interpretation of data (submission of a batch of data). • Import a new mapping model (e.g. a XSLT script or some mapping from another MDR).

Service Interface Description	
Name	Mapping Interface
Description	Provides access to mapping functionality.
Operations	• Get mapping script for a specific source and target data model (or a particular mapping model). A particular technology and script version may be chosen for the mapping script. • Get information (version, last update, system and software requirements, limitations, etc.) about a specific mapping script. • Get information about the latest mapping version.

5.4. Data Model

A data model describes how the information is represented and organized in the system. Figure 13 shows an overview of the data model of the EuMDR system, followed by a brief description of the main entities.

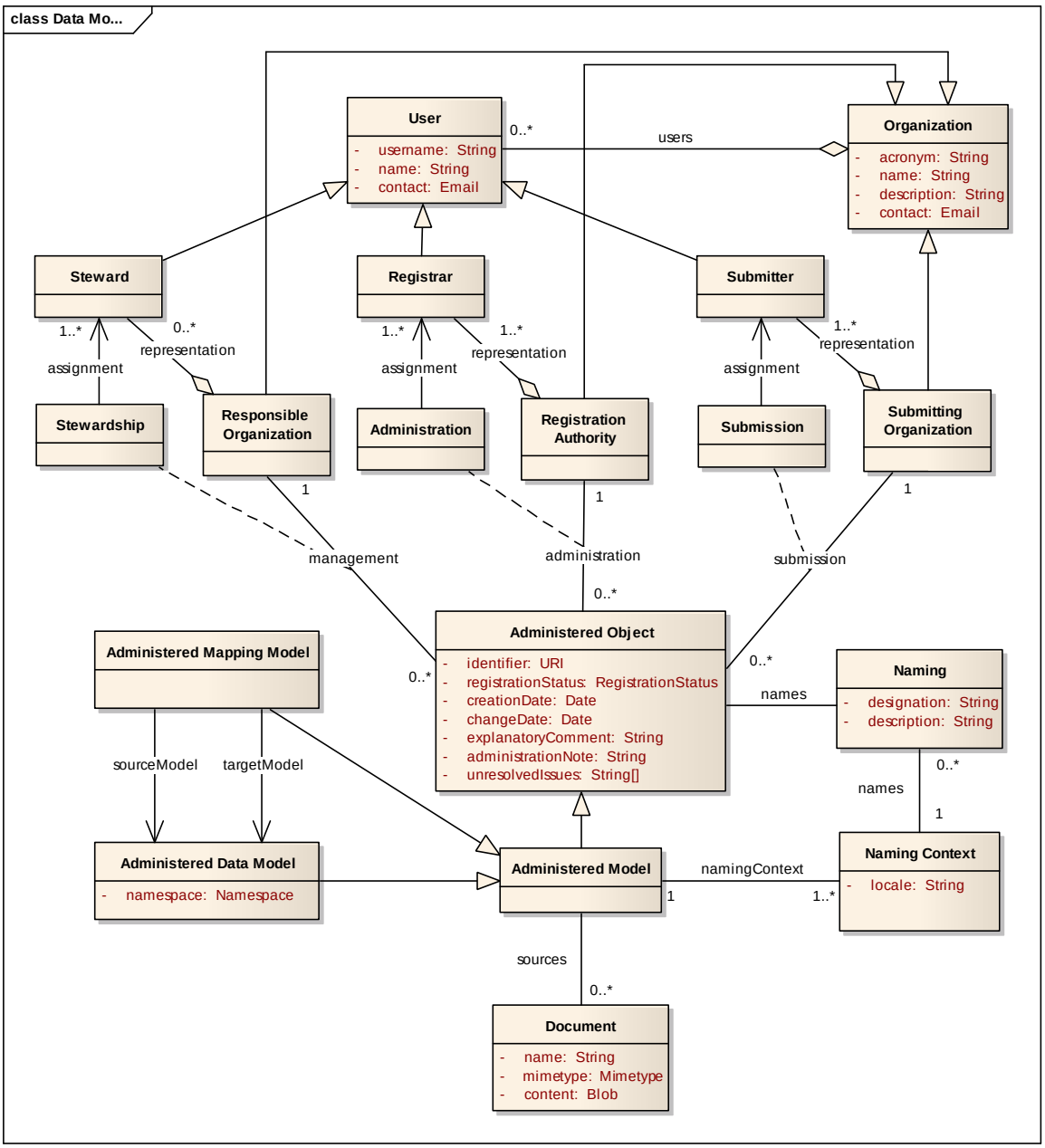


Figure 13 - Overview of the EuMDR data model.

5.4.1. Administered Object

An Administered Object is an abstract class that defines an object that is subject of registration in the MDR. It defines the current context of registration of the object.

5.4.2. Administered Model

An Administered Model is an abstract class that defines a model that is subject of registration. A model can be composed of other objects (elements) that are also subject of administration and that compose the model.

5.4.3. Administered Data and Mapping Models

An Administered Data Model and Administered Mapping Model are two concrete classes that represent respectively a Data Model and Mapping Model that are being registered in the system.

5.4.4. Naming Context

A Naming Context is a context where an Administered Model is described in a specific natural language. The Naming Context is thus composed of all the Naming definitions and descriptions defined for each element that compose the model. The Naming class is a unit of description for a particular Registration Object.

5.4.5. Document

A Document class represents an artifact (e.g. a DTD or a XSD schema file) that is associated to a particular Administered Model. It may be used to represent a particular piece of information that originated the registration of the model.

5.4.6. User

The User class represents a User of the system. This class can be refined in 3 other classes representing the 3 roles (Registrar, Steward and Submitter).

5.4.7. Organization

The Organization class represents the organizations which the user belongs. Like the user, the organization class can also be refined in 3 other classes representing the types of organizations defined in the system (Registration Authority, Responsible Organization and Submitting Organization). Each type of organization plays a different role in the registration of an Administered Object (represented by the association classes Administration, Submission and Stewardship).

6. Implementation

This section describes the software implementation of the EuMDR service. It starts by a brief introduction of the implementation guidelines followed during development; continues by detailing the libraries implemented as support for the design components; and finishes with a brief description of the external libraries being used.

6.1. Implementation Guidelines

The implementation of the component libraries that make up the system tries to follow the software development guidelines[4] and supporting technologies[5] defined by the Europeana Labs for contributing projects.

In this sense, the Java Programming language was chosen for development the software components that make up the EuMDR service. All code was developed and compiled according to the Java version 1.6. Also, the Apache Maven was chosen as build technology for the system libraries.

The software code is stored in the contributing projects subversion located at Europeana Labs[6].

6.2. Implemented Libraries

This section describes the system libraries implemented as support for the design components defined in Section 5.2. Each library description identifies its Artifact ID in Maven and contains a reference to its POM configuration file where project building and dependency information is defined.

Almost all design components were implemented except for the “Access Control and Registration Manager” component which is already supported by the JBoss jBPM Identity package.

Library Description	
Component Name	Administration UI
Technical Specification	Developed using GWT technology.
Artifact ID	mdr-console
POM	https://europeanalabs.eu/svn/contrib/mdr/mdr-console

[4] http://europeanalabs.eu/wiki/DevelopmentGuide
[5] http://europeanalabs.eu/wiki/DevelopmentSupportingTechnologies
[6] https://europeanalabs.eu/svn/contrib

Library Description	
Component Name	Data Model UI
Technical Specification	Developed using GWT technology.
Artifact ID	mdr-schema-ui
POM	https://europeanalabs.eu/svn/contrib/mdr/mdr-schema-ui

Library Description	
Component Name	Mapping Model UI
Technical Specification	The interface design is inspired by current state of the art tools for mapping between data models (e.g. Biztalk Mapper[7], Altova MapForce[8]). It is implemented in Java using the Swing widget toolkit, so to allow it to be deployed as a standalone tool or as a web component embedded in an Applet. The decision to use Swing was made because it offers a good trade-off between flexibility and performance, and is supported by all java supported platforms. Figure 14, shows an image extracted from the first prototype developed for the mapping UI component.
Artifact ID	mdr-mapping-ui
POM	https://europeanalabs.eu/svn/contrib/mdr/mdr-mapping-ui

[7] http://msdn.microsoft.com/en-us/library/aa547076(BTS.70).aspx
[8] www.altova.com/mapforce.htm

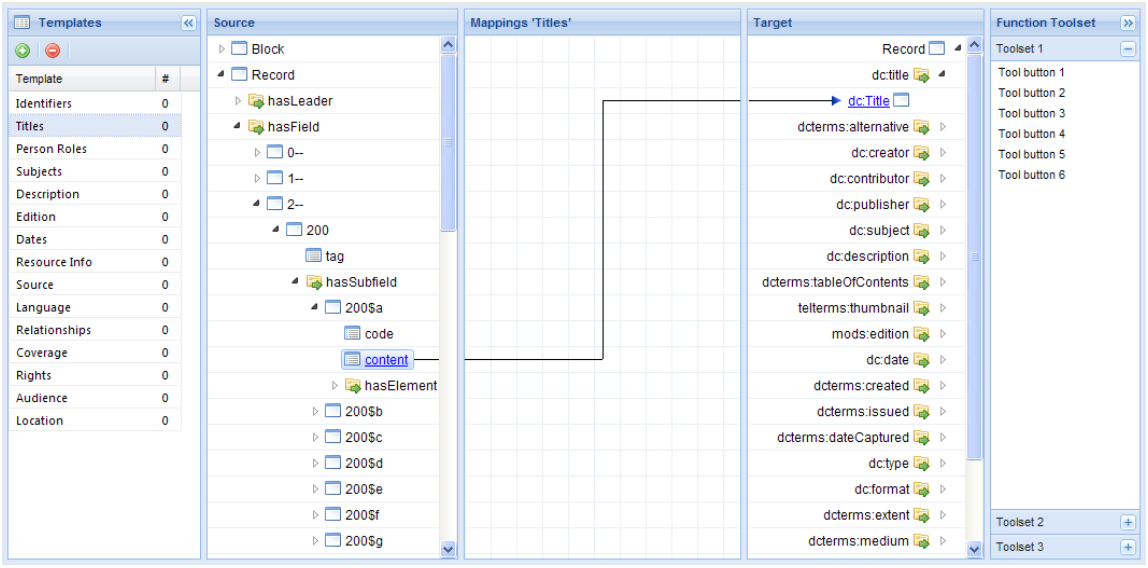


Figure 14 – Overview of the first prototype for the mapping user interface.

Library Description	
Name	EuMDR Service
Technical Specification	This component is implemented using the jBPM[9] and jPDL[10] workflow execution system which can run under JBoss or Tomcat[11] application servers. The service interfaces interacts with the workflow execution system using simple REST interfaces. Other types of service interfaces can be developed if required.
Artifact ID	mdr-service
POM	https://europeanalabs.eu/svn/contrib/mdr/mdr-service

Component Description	
Name	Access Control and Registration Manager
Description	Responsible for managing all registration related information and checking access for users.

[9] http://www.jboss.org/jbpm
[10] http://docs.jboss.com/jbpm/v3/userguide/
[11] http://www.ucosoft.com/deploy-jbpm-322-to-tomcat-and-mysql.html

Technical Specification	This component will be developed in Java and will possibly integrate with the Europeana CRM system to access information related to user accounts and roles, and to login into the system.
Owner System	EuMDR Service

Library Description	
Name	Data Model Support
Technical Specification	Uses 3 groups of libraries to support DTD, XMLSchema and RelaxNG schema languages.
Artifact ID	mdr-schema
POM	https://europeanalabs.eu/svn/contrib/mdr/mdr-schema

Library Description	
Name	Mapping Model Support
Technical Specification	Developed using the Java programming language.
Artifact ID	mdr-mapping
POM	https://europeanalabs.eu/svn/contrib/mdr/mdr-mapping

Library Description	
Name	Repository
Technical Specification	Implemented using the Hibernate Java persistence framework. A possible alternative implementation can be to store in a RDF compliant database, preferably working over a relational database.
Artifact ID	mdr-model
POM	https://europeanalabs.eu/svn/contrib/mdr/mdr-model

6.3. External Libraries

The system makes use of several external libraries. This section describes the most relevant libraries in use by the current implementation and the reason for their choice. For a more detailed description of the external dependencies, see the POM associated to each library that composes the system.

6.3.1. Spring Framework

The server side implementation of both the “Administration UI” and “Service Support” is wired together using the core Spring Framework dependency injection, with the dependencies managed in a single XML configuration file. This file can be found in the WEB-INF directory with the name of the Maven Artifact ID followed by the suffix “.config.xml”.

6.3.2. JBoss jBPM

The system uses the JBoss jBPM as execution environment to implement the main processes running at the EuMDR Service component. The JBoss jBPM also uses Hibernate for persistency. The usage of a workflow execution system allows for a simple and flexible way of adjusting the workflow to the Europeana’s needs. This also allows for a better parameterization of the requirements for each registration status level. The version 4.4 was chosen for the current system implementation.

The jBPM configuration file can be found in the root of the system library for the “Service Support” component with the name “jbpm.cfg.xml”.

6.3.3. Hibernate

The Hibernate Java persistence framework was chosen as the persistency layer for storing Java Objects. The version 3.6.0 Final was chosen for the current system implementation.

The Hibernate configuration is done through the JBoss jBPM library in the “jbpm.hibernate.cfg”, which besides including the jBPM Hibernate configuration, it also includes the system persistent data model found at “mdr.hbm.xml”.

6.3.4. GWT

The Google Web Toolkit was chosen as the development toolkit for building the Administration interfaces of the system. This toolkit eases the process of designing and creating complex web-based user interfaces by offering a wide range of UI components which can be found natively or through various extension libraries. One of these libraries that were used for implementation is the “Ext GWT”[12]. This is a highly known open source JavaScript library before the GWT and was recently re-implemented using the GWT development toolkit.

The GWT configuration can be found in the “mdr-console.gwt.xml” file which imports other two implemented GWT modules: “mdr-schema-ui.gwt.xml” and “mdr-mapping-ui.gwt.xml”.

6.3.5. JAX-RS

The Java API for RESTful Web Services was chosen to implement the service interfaces provided by the “Service Support” component. This library is used as a Java programming language API that supports in creating web services according to the Representational State Transfer (REST) architectural style. This type of service interfaces was chosen since it offers a very lightweight and easy way of publishing the services to external systems.

The JBoss RESTEasy library was chosen as the implementation support for the JAX-RS, since it highly known and was already in use by the jBPM v4.4 distribution package.

[12] http://www.sencha.com/products/gwt/

7. Conclusions

This document describes the specification, design and implementation of the Europeana Metadata Registry. From a list of functional and non-functional requirements, use cases were identified and the components and structure of the system were derived. All of these were described in detail.

It is our belief that the developed and delivered system is well implemented and satisfies the corresponding requirements. Next steps should now consist in its evaluation by Europeana, from which we expect to have to revise small details (for which the IST will be available to assist until the terminus of the project).

Appendix -1: Acronyms

BPMN	Business Process Modelling Notation
DTD	Document Type Definition
EuMDR	The European Metadata Registry
GWT	Google Web Toolkit
IST	Instituto Superior Técnico
MARC	Machine-Readable Catalog
MDR	Metadata Registry
RA	Registration Authority
RO	Responsible Organization
SO	Submitting Organization
UNIMARC	Universal Machine-Readable Catalog
UML	Unified Modelling Language
UW	University of Vienna (Universität Wien)
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformations