DELIVERABLE

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

Deliverable 4.2 – EuDML global system functional specification and design

Revision: 1.0

Authors:

Aleksander Nowiński (ICM)

Wojtek Sylwestrzak (ICM)

Krzysztof Wojciechowski (ICM)

José Borbinha (IST)

Project co-funded by the European Commission within the ICT Policy Support Programme	
Last changed: 06/09/2011	
Dissemination level:	Public (PU)

Revision History

Revision	Date	Author	Organisation	Description
0.01	30.06.2011	Aleksander Nowiński	ICM	First draft framework
0.09	15.07.2011	Krzysztof Wojciechowski	ICM	Second draft framework
0.11	25.07.2011	Krzysztof Wojciechowski	ICM	Enhancers description added
0.19	29.07.2011	Krzysztof Wojciechowski	ICM	REPOX description added
0.20	02.08.2011	Aleksander Nowiński	ICM	Extended section on data, identifiers and data handling.
0.23	04.08.2011	Michał Politowski	ICM	Revision and editing
0.24	19.08.2011	Wojtek Sylwestrzak	ICM	Copyedit
0.25	19.08.2011	Lígia Rodrigues	IST	Revision
0.26	19.08.2011	Jose Borbinha	IST	Revision
0.27	20.08.2011	Wojtek Sylwestrzak	ICM	Copyedit, alignment
0.28	21.08.2011	Wojtek Sylwestrzak	ICM	Restructuring
0.29	22.08.2011	José Borbinha	IST	Style edit
0.30	22.08.2011	Aleksander Nowiński	ICM	Changed first figure, edit
0.31	30.08.2011	José Borbinha	IST	Corrected "typos" and missing images from REPOX
0.40	01.09.2011	José Borbinha	IST	Split of previous version (core document plus appendix)
0.41	01.09.2011	Wojtek Sylwestrzak	ICM	Overall copyedit
0.5	02.09.2011	Aleksander Nowiński	ICM	Nomenclature and copyedit.
1.0	06.09.2011	Wojtek Sylwestrzak José Borbinha	ICM IST	Bumped up version following internal review, typos

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 version numbers.

This document is complemented by the "Deliverable 4.2 – EuDML global system functional specification and design – Appendix".

Other major contributors of this document or its appendix are Zuzana Nevěřilová, Gilberto Pedrosa, Marek Horst, Jacek Plebanek, Michał Politowski, Wojtek Hury, Tomek Rosiek, Radim Řehůřek, Łukasz Bolikowski, Radim Hatlapatka.

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This deliverable contains original unpublished work except where clearly indicated otherwise.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.

Table of Contents

1. Introduction

This document describes the overall architecture and the key components of the EuDML software system. Together with its complementary appendix, it is intended as the general reference document for development purposes. Therefore it is expected that the document is going to be continuously updated to reflect the evolution of the system.

This is a concise document, focused on the main concepts of the EuDML architecture. More detailed descriptions of individual services and other components are provided in an the companion document "Deliverable 4.2 – EuDML global system functional specification and design - Appendix".

1.1. Design concepts and general architecture

The EuDML system implements a Service Oriented Architecture. The functionality offered to the user is backed by custom services, which encapsulate internal data persistence and access. The services are defined by their interfaces, and the implementation is by design hidden from the user, so that it can be easily replaced with an alternative one.

The presentation layer (web user interface) has been explicitly separated from the service layer. The user interface is run in a separate application container, while all the services are deployed in another container, called here the backend container. Other (non-user) system interfaces, so called external system interfaces, which offer EuDML contents as a service, or are used to for content harvesting, are also separated. This default deployment scenario is expected to be used throughout the system's lifespan. The separation of the presentation layer and the service layer ensures that user interface may use only service interfaces and never interferes with the internal implementation.

An EuDML specific concept in the architectural design is the introduction of the so called *middle layer services*. The middle layer services may not exactly fit the service definition - formally they may run on the client side. These are software components responsible for applying business logic to a set of base services and to provide efficient and unified method of operating on basic services. This layer is also responsible for application metadata format specific logic, as the basic services usually do not rely on custom metadata format. A most important example of this service type is EuDML store, which applies metadata format logic onto underlying storage services.

1.2. EuDML general architecture

The key components of the EuDML system are the Backend, the User Interface and the External Services Interface (Figure 1).

1.2.1. Backend

The *Backend* is a collection of services providing all the internal functionality of the system. This includes both basic repository services and more complex services, including some oriented towards the user interaction. The basic repository functionality covers metadata storage, content storage (including full-text files), searching and relation browsing. It also includes technical aspects of the data processing, like indexing documents.

Other services in the backend are responsible for user interaction (user registry, annotation service), metadata enhancements, as well as other internal aspects of the system.

The backend is accessed only by web user interface and by external services interface components – it is never accessed directly by the remote clients. Therefore to avoid unnecessary complication on the communication level, decision has been made, that security will be applied on the upper level – in the web interface module and in the external service layer.

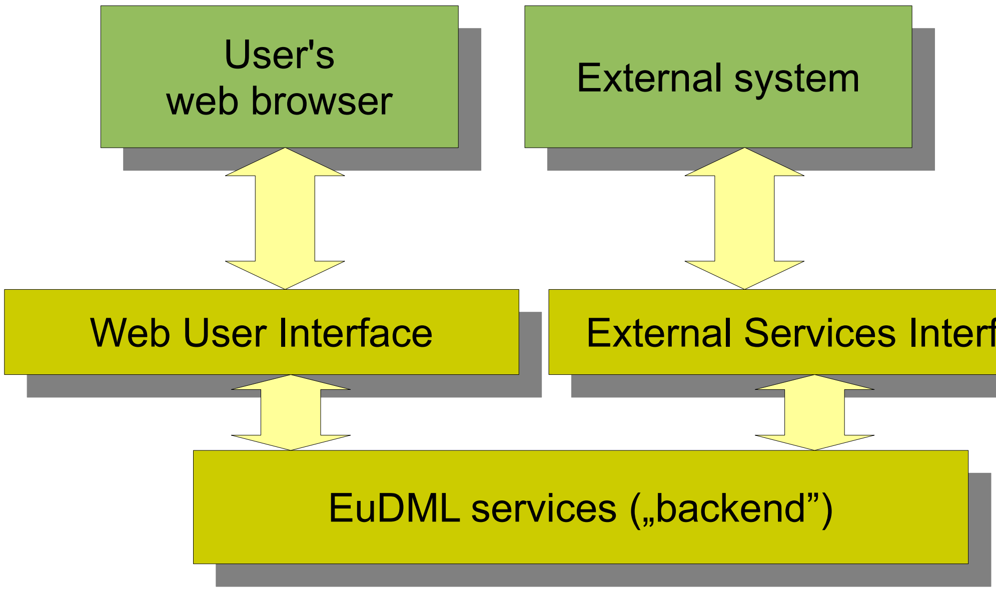


Figure 1: High-level system architecture view

1.2.2. User interface

The *User Interface* component is a web application, which is responsible for communicating with users using web browsers. This component uses backend services to read/write the data, and exposes the metadata repository contents in a user friendly way. In this layer security is applied ensuring that user may perform only authorized action.

1.2.3. External Services Interface

The *External Services Interface* is carried out by the REPOX system, which manages the harvesting processes of the metadata and full-text provided by the data providers, together with the processes of data transformation and normalization. It also comprises an OAI-PMH server for sharing metadata with external parties.

Also in this component security is enforced, so remote clients are properly authorized while performing actions.

1.3. EuDML as an XML record based system

The EuDML is designed as a record-oriented system. The system manages a collection of records, where each record represents a single information entity uniquely identified by an EuDML identifier. An information entity will usually be a publication (such as an article, or a book). Each record is a collection of XML metadata files and related content files.

This approach is the opposite of the typical relational-database driven approach, where records are decomposed into rows in multiple tables.

The benefits of the EuDML approach include the flexible metadata format support and easy adaptation to new requirements or extensions of the existing metadata formats. It is also beneficial in a case of complex or rich metadata, when full relational decomposition would require a number of tables and rows to map a single publication.

1.3.1.Relations and indexes separated from the data

As the data is stored in the form of isolated records, the relations between the records and the browsing indexes must be kept separate in dedicated services. This is a necessary trade-off for benefits of XML record driven architecture. Two core components of the system are used to maintain this task in EuDML: the general purpose Search Service and the Browse Service.

1.3.2.Workflow processing

The fact that the services are separated from the records themselves implies that XML source records must be processed to extract the necessary information for the indexes and the relations services. To this end a workflow processing engine is used. The processing engine is capable of iterative application of the specific workflow to a collection of the records in the system. As a result, the relations and search index entries are generated, but the same technology is used to perform metadata enhancement and clustering.

1.4. Communication layer

As a base communication layer for the system HTTP Invoke remote invocation mechanism has been chosen. To provide efficient and lightweight configuration of the communication layer Spring Remoting framework is used. System does not make assumption about mechanism used to invoke remote services, and HTTP invoke may be replaced by any other mechanism provided by Spring Remoting easily (RMI, WebServices etc.).

2. Overview of the technology used

All the existing technologies to be used in EuDML have been carefully selected not only to perform accordingly but also to be both open-source and open-licensed. Eventually, the following set of technologies has been chosen to support the EuDML solutions.

- *Java* – the system is Java based, developed and tested with SUN/Oracle Java v. 1.6
- *PostgreSQL* is used as a relational database engine
- *JDBC* and *Spring JDBC* are mostly used in low level services to access database storage
- *Hibernate ORM* - used by some services for persistence layer
- *Webapp* - core components are packaged as standard Java webapps, using WAR packaging
    - *Tomcat webapp container* is used for operational run
    - *Jetty webapp container* - used extensively in development process
    - *Apache http server* - used as a proxy in complex deployments
- *Spring framework* (v2.5) - used as a core framework of the project
    - *IOC container* - used to define deployment and software component orchestration
    - *Spring remoting* as core communication layer between components (configured to use HTTP invoke as transport layer)
    - *Spring MVC* is a core technology of the web UI
    - *Spring security* - manages UI security and login process
    - *Spring integration framework* is used as a base for workflow processing service
- *Solr search engine* - YADDA search engine is based on Solr
- *Apache Lucene* - used by YADDA similarity service
- *YADDA framework* - core services are either plain YADDA services, or are developed to fit into YADDA framework
    - YADDA services (storage/search/browse/process/user db)
    - YADDA service architecture & remoting
- *REPOX Service* (harvesting data from content providers)
    - *Google Web Toolkit* – Java software development framework for user interfaces
    - *Xalan-Java* – an XSLT processor for transforming XML documents
- *Apache Maven* – software project management and comprehension tool

3. EuDML services and components

The EuDML system follows the Service Oriented Architecture paradigm, and consists of a number of inter-communicating services. The basic set of EuDML services is introduced in this section. In the future the set can and is intended to be extended to support new or enhanced functionalities of the system.

For detailed description of individual services consult the Appendix to this document.

Figure 2 depicts the general dependencies between the services.

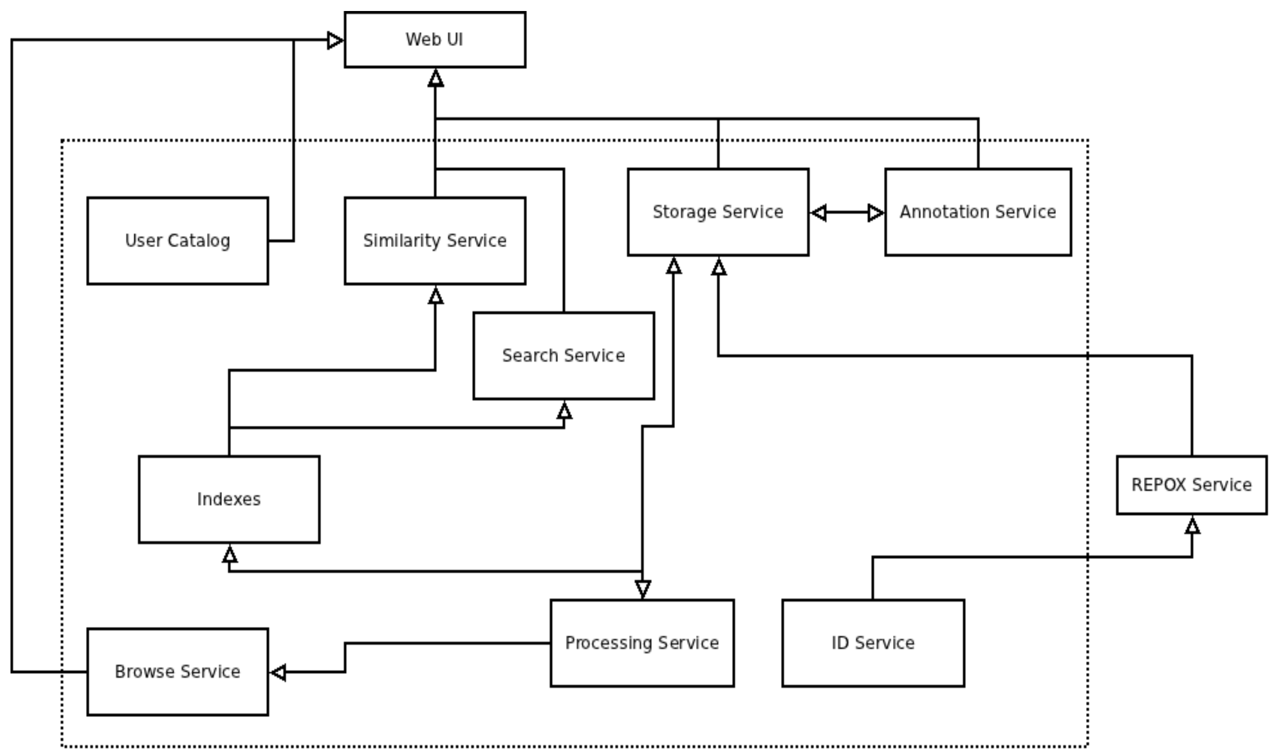


Figure 2: EuDML services interdependencies diagram.

3.1. Workflow Processing Service

As the system is generally record-oriented and XML-based, there is a need to perform iterative processing of the records within the system. This processing typically is:

- indexing data for a search service
- building specific relations in other services (e.g. browse service)
- performing metadata enhancement and clustering

In EuDML project YADDA Processing Service is used as a workflow service.

Processing Service orchestrates and applies the EuDML workflows to the data. It operates on following basic concepts: The workflow defines a number of nodes and the data flow. The process is operation of applying selected workflow to a number of messages. The message is an atomic processing unit, which is passed into a workflow. The Workflow nodes are operations applied to messages, and flow defines order of the node execution. During processing messages are passed between nodes according to the workflow definition. Workflow is defined by channels linking nodes and routers, which may alter the flow (conditional selection of channel, starting parallel execution on different channels etc.) Typically, a message is a publication record, and nodes are single operations - like building index document or extracting text from PDF file. A message may be transformed by nodes or consumed (e.g. stored into persistent storage).

Figure 3 depicts the general processing concept for an example workflow. It starts with an iteration over a store. Then messages are then passed through *processing nodes* and possibly *routers*. The input and output of each node can be of different type but in any given flow the input of a subsequent node must match the preceding node's output. The final messages are stored by *writers* in some kind of stores.

Workflows are defined statically in Spring configuration files.

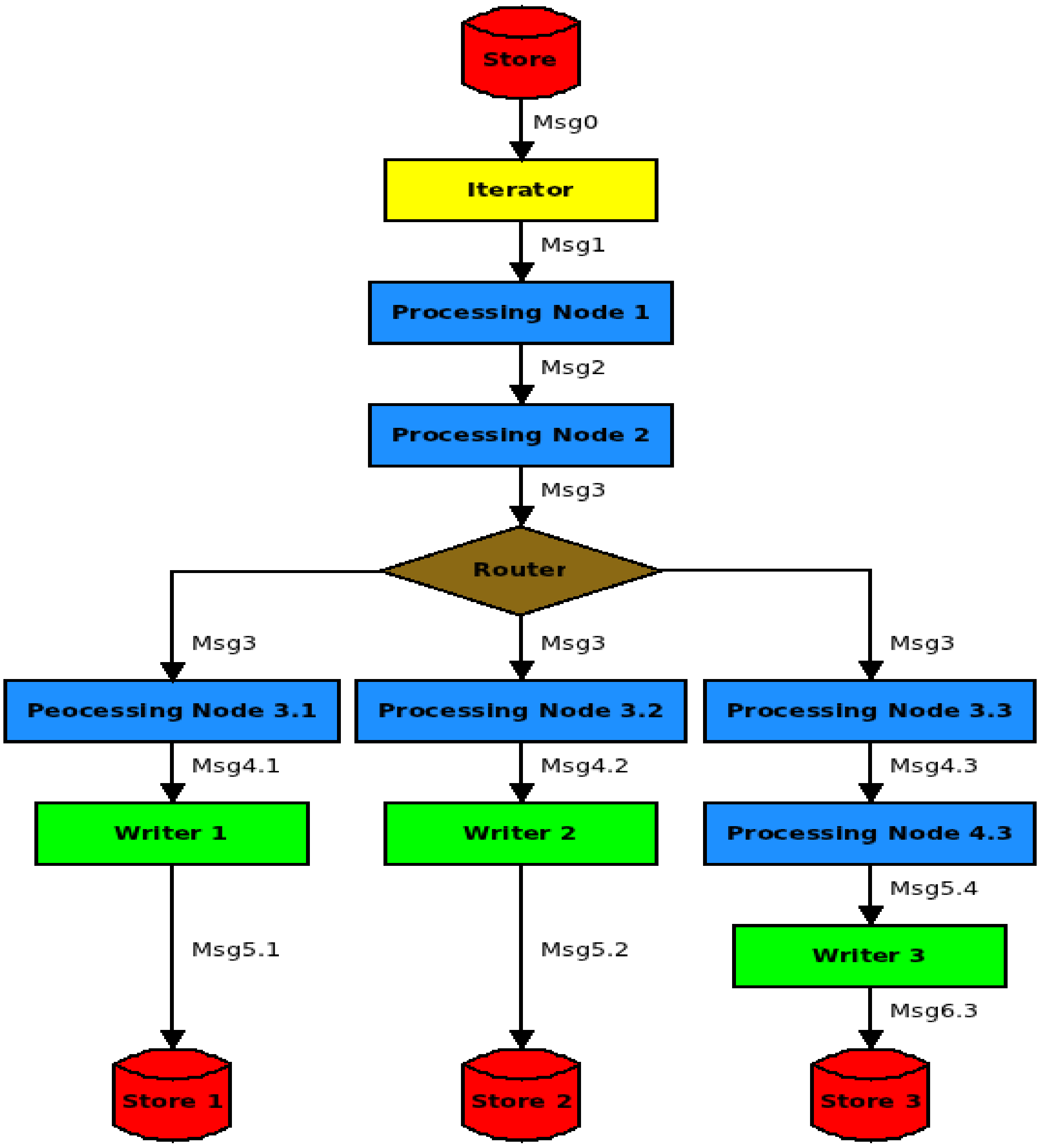


Figure 3: General processing workflow

3.2. Content Aggregation Service

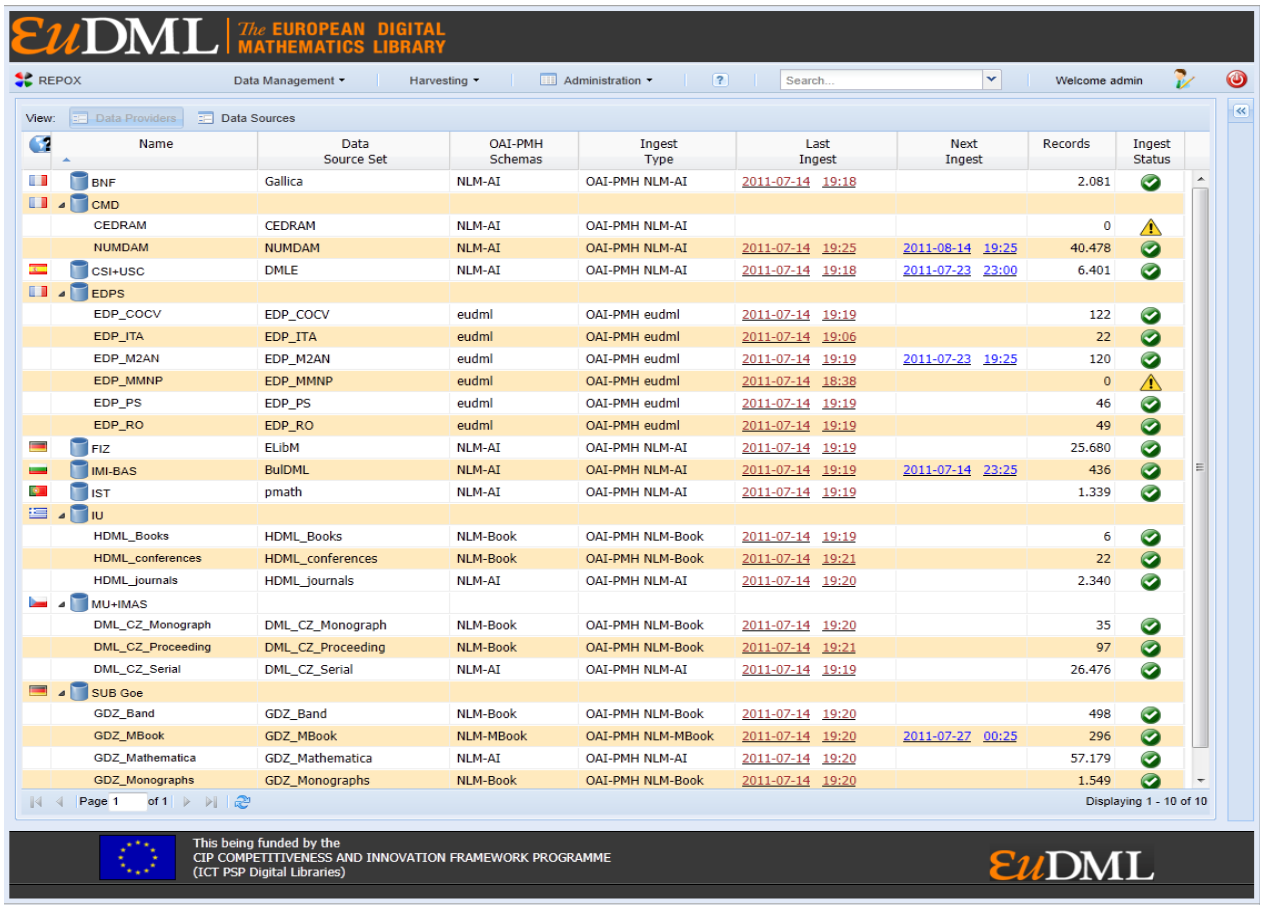
The Content Aggregation Service uses REPOX technology to manage the harvesting processes of the metadata and full-text provided by data providers, as well as the processes of data transformation and normalization.

In brief, REPOX provides the following main functions:

- Registration of data providers, their collection descriptions (a single data provider can make more than one collection available to EuDML), and the configurations for the harvesting of the related metadata and full-texts.
- The automatic and manual harvesting of the metadata by OAI-PMH (according to configurations and options provided by the data providers and the decisions of the administrators of the central service) or by HTTP or FTP. It is provided Support for multiple metadata formats (OAI-DC is assumed by default, but any other format is also possible). The harvesting of the full-text is expected to be done through HTTP or FTP.
- The delivery of metadata to external partners (by this way, records that had been enhanced by the EuDML specialized services can be given back to their original providers).
- Monitoring of the quality of service of the OAI-PMH servers, including statistics.

The REPOX service has its own graphic user interface (example in Figure 4) to be used by humans.

REPOX also provides to the outside a set of REST services for data providers, data sources and records.



Name	Data Source Set	OAI-PMH Schemas	Ingest Type	Last Ingest	Next Ingest	Records	Ingest Status
BNF	Gallica	NLM-AI	OAI-PMH NLM-AI	2011-07-14 19:18		2.081	✓
CMD							
CEDRAM	CEDRAM	NLM-AI	OAI-PMH NLM-AI			0	⚠
NUMDAM	NUMDAM	NLM-AI	OAI-PMH NLM-AI	2011-07-14 19:25	2011-08-14 19:25	40.478	✓
CSI+USC	DMLE	NLM-AI	OAI-PMH NLM-AI	2011-07-14 19:18	2011-07-23 23:00	6.401	✓
EDPS							
EDP_COCV	EDP_COCV	eudml	OAI-PMH eudml	2011-07-14 19:19		122	✓
EDP_ITA	EDP_ITA	eudml	OAI-PMH eudml	2011-07-14 19:06		22	✓
EDP_M2AN	EDP_M2AN	eudml	OAI-PMH eudml	2011-07-14 19:19	2011-07-23 19:25	120	✓
EDP_MMNP	EDP_MMNP	eudml	OAI-PMH eudml	2011-07-14 18:38		0	⚠
EDP_PS	EDP_PS	eudml	OAI-PMH eudml	2011-07-14 19:19		46	✓
EDP_RO	EDP_RO	eudml	OAI-PMH eudml	2011-07-14 19:19		49	✓
FIZ	ELibM	NLM-AI	OAI-PMH NLM-AI	2011-07-14 19:19		25.680	✓
IMI-BAS	BulDML	NLM-AI	OAI-PMH NLM-AI	2011-07-14 19:19	2011-07-14 23:25	436	✓
IST	pmath	NLM-AI	OAI-PMH NLM-AI	2011-07-14 19:19		1.339	✓
IU							
HDML_Books	HDML_Books	NLM-Book	OAI-PMH NLM-Book	2011-07-14 19:19		6	✓
HDML_conferences	HDML_conferences	NLM-Book	OAI-PMH NLM-Book	2011-07-14 19:21		22	✓
HDML_journals	HDML_journals	NLM-AI	OAI-PMH NLM-AI	2011-07-14 19:20		2.340	✓
MU+IMAS							
DML_CZ_Monograph	DML_CZ_Monograph	NLM-Book	OAI-PMH NLM-Book	2011-07-14 19:20		35	✓
DML_CZ_Proceeding	DML_CZ_Proceeding	NLM-Book	OAI-PMH NLM-Book	2011-07-14 19:21		97	✓
DML_CZ_Serial	DML_CZ_Serial	NLM-AI	OAI-PMH NLM-AI	2011-07-14 19:19		26.476	✓
SUB Goe							
GDZ_Band	GDZ_Band	NLM-Book	OAI-PMH NLM-Book	2011-07-14 19:20		498	✓
GDZ_MBook	GDZ_MBook	NLM-MBook	OAI-PMH NLM-MBook	2011-07-14 19:20	2011-07-27 00:25	296	✓
GDZ_Mathematica	GDZ_Mathematica	NLM-AI	OAI-PMH NLM-AI	2011-07-14 19:20		57.179	✓
GDZ_Monographs	GDZ_Monographs	NLM-Book	OAI-PMH NLM-Book	2011-07-14 19:20		1.549	✓

Page 1 of 1

Displaying 1 - 10 of 10

This being funded by the
CIP COMPETITIVENESS AND INNOVATION FRAMEWORK PROGRAMME
(ICT PSP Digital Libraries)

EuDML

Figure 4: REPOX Graphical User Interface

3.3. Storage Service

The EuDML Storage Service (EuDML store) is the central part of the system. It is a middle-layer service, which encapsulates logic of the record format and layout and implements it using lower layer services (currently YADDA MD-Storage and YADDA Archive). Its main role is to store and furnish both metadata and content (both full-texts and plain-texts, typically being PDF and text files) of the information entities maintained by the system. The internal structure of the service assumes that each entity consists of a number of parts, and both metadata, and contents can be stored in many concurrent formats. For example, almost all items in EuDML will have their corresponding metadata in Zentralblatt Math. So the Storage Service keeps its ZBMath metadata alongside with the metadata harvested from the content provider's OAI-PMH server and converted to NLM format by the Content Aggregation Service.

The main reason to store the items' contents (payloads) is to have them ready available for subsequent iterative mass-processing, e.g. text mining. The second reason is to keep backup copies for redundancy and possible future long term preservation purposes. The EuDML system is currently not intended to serve publication contents (full-texts) from its stored copies but instead link to the original source at the data provider's site.

In a number of cases it may be useful to keep full-texts in different formats. The EuDML system can at the same time keep parts of a publication in PDF, and other (or actually the same) parts in plain-text or XML structures extracted by specific tools. If a data provider supplies EuDML with e.g. LaTeX source files, they may be stored as well in order to avoid the need to extract the structure and details from the rendered files. Finally, keeping the original and processed full-texts allows EuDML maintainers and developers to gradually improve processing tools in order to obtain better quality of the derived data.

Other EuDML services communicate with the Storage Service in order to obtain data to be enhanced, presented (displayed) or to store data after being enhanced. At a lower level, persistence of the data is assured by the Storage Service by a PostgreSQL database.

3.4. User Directory Service

The User Directory Service is dedicated to store information about users, groups and relations between them. This service is also used for storing additional information about users e.g. email address, biography etc. It can also authenticate user using pluggable token/credential mechanism.

The stored information concerning users includes:

- user domain (the User Registry implementation supports multiple domains)
- user identifiers (from different namespaces)
- user attributes
- user flags
- user credentials (used in user authentication process)
- groups to which user belongs

The stored information concerning groups is:

- group domain
- group name
- group roles
- whether the group has a parent group (from which it inherits roles)

In EuDML there is a custom middle layer service, EuDML UserDB, which uses User Directory to authenticate users. This EuDML UserDB provides user/password authentication implementation over the generic authentication mechanism. It may also be used to keep other information provided by the users e.g. their email address, biography etc.

3.5. Search and Browse Services

There are two aims that the Search Service serves. Firstly, it is used as a document indexer – it stores document's metadata (title, authors, abstract...) and full-texts in the index. Secondly, as a document searcher – it provides API to search the index for documents corresponding to a specified query (using provided Java query model).

The Search Service is based on Apache Solr (Figure 5). It serves as a simplified outer layer (completely hiding the Solr API and configuration) which provides:

- simple Java API for document indexing and searching
- simple XML (and Java) index metadata configuration (index fields declaration)
- fully automatic Solr instance configuration and initialization

It uses many of the Solr advanced capabilities and extends them through the mechanism of plug-ins. Its main features include:

- advanced Solr search functions, such as: Faceted Search, Filtering, Dynamic Fields
- Solr features with improved or customized processing logic, such as: Sorting, Highlighting, Wildcard Search
- new features, designed to improve search in such cases as: authors name search, full-text search
- special plug-in for mathematical formulae (in MathML format) indexing and searching

The Browse Service is a supporting service, responsible for dealing with relations between objects. This service allows to define relations, index them for fast access, and query the data in these relations. This service is used to provide effective browsing over data objects within the storage. It also allows efficient querying of aggregated data, for example fetching the count of objects fulfilling specific criteria, by maintaining lazily materialized aggregated data views. The Browse Service supports numeric, boolean and timestamp data as well as case-sensitive and insensitive text strings and string arrays.

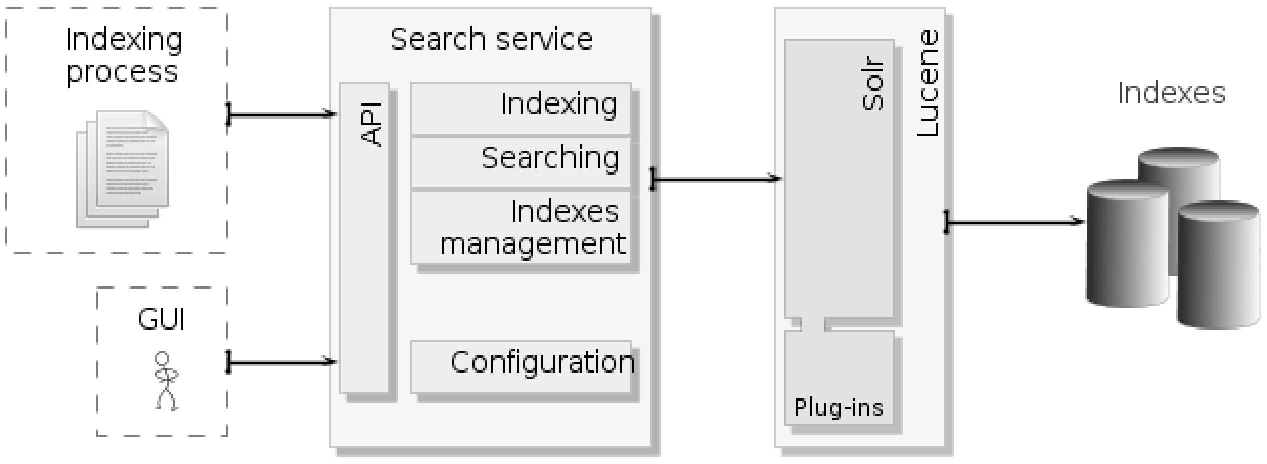


Figure 5: Search Service Implementation

3.6. Similarity Service

The Similarity Service detects similarities between text documents. A given document, either already stored in the service or specified in the request (in the latter case, document's text must be contained in the query), is used as a basis for a similarity query to find other similar documents stored earlier in the service. What exactly 'similar' means, depends on the specific implementation.

EuDML has two implementations of the service which share a common API:

- YADDA similarity (Lucene based)
- Gensim (developed at Masaryk University team)

A single target implementation has not been chosen.

3.7. Annotation Service

The Annotation Service is dedicated to store pieces of information related to a particular target item having a URI. In EuDML every identified information entity can be annotated. It allows the users to express their observations concerning any given item and store them in order to be shared with other, possibly future, users interested in it.

The annotation bodies are stored in YADDA MD-Storage and annotation relations (ownership, target, state, visibility, language...) in Sesame RDF storage. Serialization is realized via XStream.

3.8. ID Service

The ID Service is responsible for managing assigned identifiers (in the form of URI) and ensuring that all the identifiers are unique and assigned only once. The service manages and keeps track of the identifiers assigned to the objects.

It is a functional requirement that all information entities in the EuDML system have persistent, unique, long-term identifiers. It is assumed that if a publication has been visible to the users under a certain id for even a short period of time, it may have been linked from external pages or otherwise referred to and thus this id cannot disappear or point to another resource in the future. As the EuDML data space is not a static collection but a living system which grows all the time with new data being constantly imported, those identifiers must be assigned with care. Imports often are performed multiple times with the same data, and it has to be assumed that a new version of an existing object may appear in source repository. The same information entity can be also imported independently from multiple sources. To handle these cases the ID management service has been developed.

The service operates basing on the assumption that each document has a number of identifiers of various types. Those may be well recognized, unique identifiers (like DOI, Zentralblatt) or repository specific ids. The ID service is called whenever a document needs assigning identifier (it is imported into EuDML system). At this moment, the service is provided with a complete list of all known document identifiers, in form of pair (type, value). The service checks if at least one of the identifiers is already known to the system. If so, all identifiers are added as equal to the known one and already assigned EuDML identifier is returned. If none of the identifiers is known yet to the service, then a new identifier is assigned, and the provided identifiers are stored as equal to newly created identifier.

Service data is persisted within a database and is a key to keep proper identifiers management. In the case of re-importing all the data into EuDML (e.g. to a new instance of the system) it is sufficient to migrate this service database to ensure that during the import process all the publications will have the same identifiers, as they had in original system.

Id assigning policy is done according to the EuDML identifiers assigning rules. The identifiers are started from 1000 and then each new id is generated as a simple increment of the latest identifier assign. This protects the 'golden identifiers' (short ones) for special purposes, and uses namespace efficiently, generating identifiers as short, as possible.

3.9. User Interface

EuDML User Interface is a typical Java based web application, developed with use of widely acknowledged Open Source tools including JSP, Spring Framework, Tiles and Apache Commons tools. In addition to these standard tools, the EuDML UI makes use of the YaddaWeb2 framework developed by ICM, which allows to easily implement repository frontends. The EuDML UI is accessing the EuDML Repository through HTTPInvoke remoting protocol. The following services of the Repository are used:

- EuDML store - which serves metadata and contents
- Browse Service - which serves lists of journals
- Search Service - which is responsible for performing searches
- User Registry Service - which stores user accounts and is used to authenticate users
- Annotation Service - which stores user created content
- Person Directory - which is responsible for serving data about contributors

Authorization and Authentication in EuDML UI is based on the Spring Security Framework. The framework is customized in order to use EuDML specific authentication data source (User Registry Service). Spring Security gives off-the-shelf support for features like OpenId integration and authentication through persistent cookie. Since all infrastructure of EuDML maintained by the participants of EuDML project, there is no need to implement service-to-service security between backend and User Interface. Because of that, security enforcement is performed only in the EuDML UI application and on the external service interfaces.

3.9.1. YaddaWeb2 framework

YW2 (YaddaWeb2, formerly YaddaWebLite) is a set of components allowing to easily implement common repository frontend functionalities:

1. Searching for publications
2. Page lists retrieved from different sources
3. Displaying details of particular publications in convenient way
4. Filter and enrich text entities from the publication metadata to be displayed on the web page (highlight search texts, convert domain specific formatting into HTML)
5. Consistently handle system errors, access violations and trials to access non-existent resources.
6. Display AJAX tree with hierarchy of publications

YW2 is a Java framework intended to be used along with Spring and Spring MVC frameworks. It is distributed as a set of JAR packages which may be included in a particular Java Web application. Having included YW2 in an application, it is possible to make use of concrete classes or components to implement particular functionalities. In most cases it will be necessary to implement some domain-specific environment of particular components like datasources, metadata deserialization components, view templates etc. Since all components belonging to the YWL are loosely coupled, it is possible to use only some of them. A typical YaddaWebLite based application may look like in Figure 6.

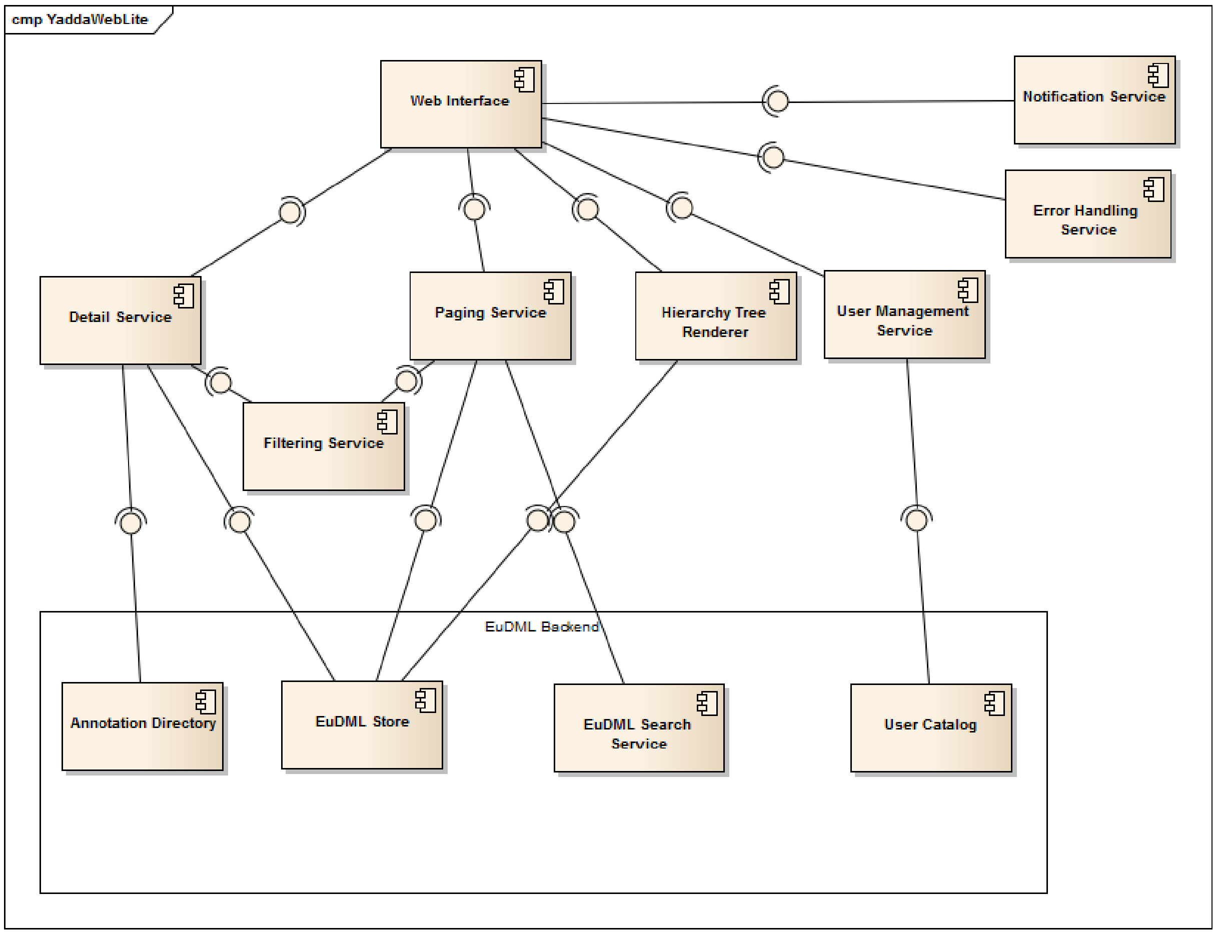


Figure 6: YaddaWebLite with EuDML Backend Services

3.10.External Services Interface

The External Services Interface is intended to provide data and service-level communication between EuDML and third party applications or services. Initially, the metadata can be exposed in a controlled way through an OAI-PMH server. Other interfaces may be researched, developed and implemented in the future (namely REST services).

4. Metadata and data handling

For a digital library, proper handling of metadata and data is, obviously, a crucial point of its design. EuDML handles object's metadata, content (associated files) and plain-text. Each object's metadata are imported from content providers, converted, aggregated, stored and maintained locally together with the related plain-text. The other object's content files (e.g. PDF) will be downloaded from their providers and stored within EuDML for subsequent processing, but the system is not intended to be serving them to the users, as long as they are available in from the original provider's site. It may be possible to serve the content from the EuDML archive, though, if the original site is inaccessible.

4.1. EuDML Identifier (eID) format

One of the goals of the EuDML is to offer standard, long living identifiers and URLs for the objects (publications), in order to offer a unified way of online referencing mathematical works. These identifiers and URLs must be short enough to be usable and must share a common format and assigning strategy.

In EuDML IDs are URNs with custom namespace: **eudml**. This way we offer reasonable standard format of the identifier, which is the custom string will be composed of two parts:

- object type
- object identifier (an string in format dependent on the object type)

We have decided to incorporate object type into identifiers. With this, we would like to preserve document identifier space and not to waste it on other, less important objects, like annotations.

EuDML identifier has the following format:

```
urn:eudml:<type_id>:<object_id>
```

Specific objects have special rules for the proper id format, as described in the table below:

Object type	type_id	Identifier format	example
Publication	doc	For objects with content ("leafs") a numeric string as assigned by ID service	`urn:eudml:doc:12345`
Annotation	an	If an annotation refers to another object: <random uuid>@<referred object id short form> and if not: <random uuid>	`urn:eudml:an:63ce7b10-01f0-11e0-a976-0800200c9a66@doc:12345` `urn:eudml:an:63ce7b10-01f0-11e0-a976-0800200c9a66`
Classification	classification	manually assigned code	`urn:eudml:classification:msc`
Classification category	category	<classification id>-<code>	`urn:eudml:category:msc-33C15`
User	User	Random UUID based	`urn:eudml:user:63ce7b10-01f0-11e0-a976-0800201c9a66`
Bookshelf	bookshelf	Random UUID based	`urn:eudml:bookshelf:63ce7b10-01f0-11e0-a976-0800201c9a66`

Note that only document and annotation type id are abbreviated. We have decided that this is optimal, as in fact only those two types of objects have typically external address, and those identifiers are expected to be passed outside of the system, so short notation is an advantage for external system users.

4.2. Short form

Short eID form is an URN without namespace (**urn:eudml:**) prefix so, only has type string and proper identifier. The short form will be used inside the system and in URLs when appropriate.

4.3. URNs and URLs

As the UI will offer stable addressing, a persistent mapping (URN->URL) will be defined as the URL of the document in EuDML system. This mapping will be permanent, so a document may be cited as its URL

4.4. Document identifiers assigning

As the same publication may appear in multiple providers collection and may be imported multiple times to the system, we must track identifiers carefully. The namespace of the identifiers has no limit, but users expect that the same publication has the same EuDML identifier regardless of the import source and time.

To keep track of the identifiers we have chosen id assigning strategy based on other document identifiers.

Each imported document has one or more identifiers known during the import time, including provider's internal identifier. Each of those identifiers is described as a pair of text strings (identifier type, identifier value), which are assumed to be unique. Other than providers identifiers are cross-system identifiers like DOI, Zentralblatt or AMS citation. Whenever a document is imported into a system, an ID service is consulted and all known document identified are passed as a request. If for at least one of the identifiers EuDML ID is known, then this EuDML ID is returned, and all identifiers passed to the request are assumed to be equal to this identifier. Else, if no identifier is known to the ID Service, then a new identifier is assigned as a first free identifier (i.e. next in sequence), and all passed identifiers are stored as equal to this one. The content of the ID Service database is complete information on all assigned ID, and is sufficient to ensure, that if all the contents are imported again, then the same identifiers will be assigned to the same publications.

4.5. Stored metadata

As the record may appear in multiple source forms, within the system record is stored in multiple formats, including all source forms. This is important, as we know that it is not always clear, that we can choose best version of the record, and therefore merging of the metadata during conversion to NLM format may be desired. Even if we have only one provider for the record, we still often have a Zentralblatt record version, often containing extra metadata not available on the original source.

To unify this, within the store all source metadata are stored, and during import an EuDML metadata (in NLM format) is generated according to specific conversion rules. Those rules are pluggable and may be adopted according to the particular situation. This generated NLM is the so called 'base NLM' record, which contains the best possible metadata converted from all source providers and merged into a single record. This format does support storing full-text within NLM.

4.6. Used formats

Metadata are kept in EuDML in XML based formats. Every format has its own XML Schema. All provider source format records are stored within EuDML.

Main operation format is NLM adapted to special requirements of the service. It is described in the full details in Deliverable 3.2. Except of NLM, also Zentralblatt MATH internal format is used to deal with their data. This format is the base for all operations and displaying the data to the user through the UI.

For storing mathematical expression, MathML is primarily used. As an alternative to MathML, the service keeps mathematical expression also as LaTeX code (as a special tag inside NLM files) because this format (though not XML-based) is widely used by mathematicians and people using mathematics in their work.

For full-text items, the service uses mainly PDFs files (both OCR-ed and digitally born) as this is de-facto standard in digital libraries. Technically there is no problem to keep also other formats (e.g. DjVu, TIFF).

4.7. Handling contents

Content (metadata, plain-text, and full-text) is placed in Storage Service by REPOX Service in available source format. During this process, REPOX does conversion from content provider's specific formats to NLM format used in EuDML. In this step, persistent identifiers are generated by ID Service. The converted NLM file contains no plain-text inside and, if plain-text is present, it is stored separately. This is done for future processing efficiency, as NLM are used constantly, while plain-text files are used only once (during indexing process) and may have significant size.

In the next step, metadata are enhanced with ZBMath metadata by lookup either by ZbMath identifier or, if identifier is not known, using author, title etc. in the query. If ZBMath metadata is present, it is stored as another source metadata and NLM is generated once again, containing data from all sources, according to used merging procedure.

Later, if no plain-text is provided, extraction from PDFs is performed (it can be OCR or another techniques in case of digital-born PDFs). In the next stage, plain text is enhanced with MathML, which data are used intensively by whole service. The latter step can be combined with PDFs extraction; it depends on the data that are available.

The two latest steps may be applied once again, if there is hope to obtain better results. Typical example is when work will not be present in ZBMath, but after few months it will be present, so it will be downloaded.

After finishing this stage, the system contains a full set of required data and the internal processing starts: the Search Service indexes plain-text and metadata and other relations stored in the system are filled in with data. In this step, also similarity index may be generated/updated.

In the next step, matching references process tries to identify references in the documents with items. This is finally used to make links between references and objects.

As data comes from many independent sources there might exist duplicates in the final collection that forms EuDML. The last step is to find and merge duplicates. This is done at the end of handling contents because information gathered during the previous steps can influence this action.

Also detection of the duplicates will occur over time, even after data is presented to the public. If this late duplicate detection occurs (possibly supported with expert knowledge), then old references will redirect to one, unified record, so references to all duplicates will be supported.

4.8. Metadata enhancement

Metadata enhancement tasks shall be implemented by a number of partners, using a number of different technologies. Therefore, it is important to design a flexible enhancement architecture.

Each enhancer is implemented as a set of workflows that are to be sequentially run in the backend by a resource manager. Each process consists of a number of the so-called *processing nodes*. The first node in the workflow (*source node*) typically iterates over contents of a repository: fetches items one-by-one and feeds them to the next node (or nodes) in the workflow. The terminal node (or nodes) in the workflow (*writer node*) typically stores incoming items in a repository.

Processing nodes are written in Java as implementations of certain interfaces (such as IProcessingNode<I,O>), and workflows are defined using Spring framework configuration files (XML format).

Figure 7 Shows an example of an enhancer implemented using two workflows. Note that workflows are allowed to reuse actions.

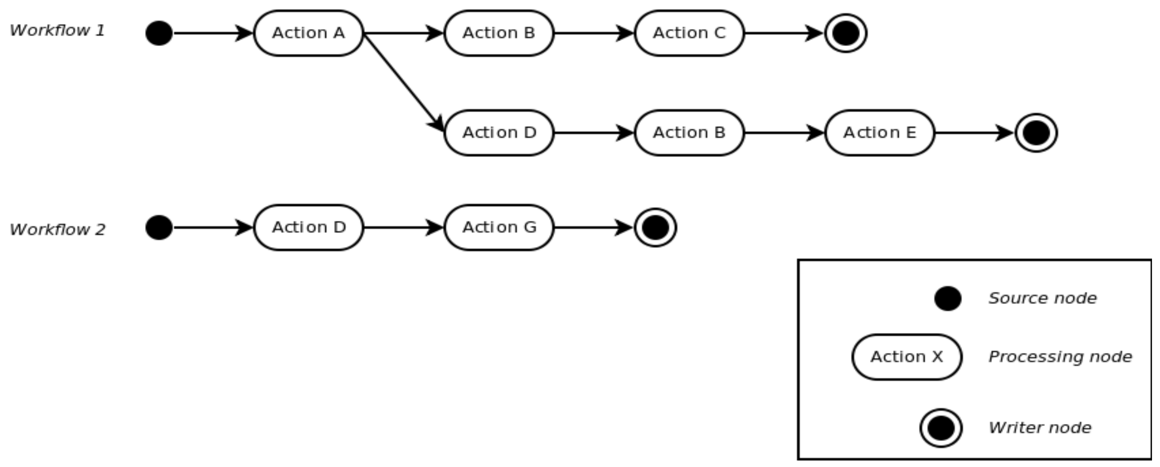


Figure 7: EuDML metadata enhancement workflow.

In this moment the expected available enhancers are:

Enhancer Name	Description
Bibliographic reference analysis	Extracts, parses and matches bibliographic references
PDF type checker	Checks if PDF is digitally- or retro- born.
PDF MathML extractor	Converts digitally-born PDF documents into plain-text enriched with math formulas expressed in MathML
NLM TeX to TeX + MML	Converts TeX formulas tags from NLM metadata into both MathML and TeX NLM tags
TeX to NLM	Converts TeX formulas into MathML and TeX tags
PDF to Text via OCR	Extracts text from PDF files containing scanned documents
Math. metadata lookup	Lookup additional metadata e.g. in Zentralblatt Math
PdfJbIm	(Re)compress PDF documents containing scanned bitmaps
PDF size optimization	Optimizes size of PDF documents without causing any loss of information