Heritage of the People's Europe

HOPE

Heritage of the People's Europe

Grant agreement No. 250549

Deliverable D4.1

T4.1 Service Infrastructure Detailed Design

Aggregator Infrastructure Detailed Design

Version	1.0
Date	31/03/2011
Status	Final
Authors	Alessia Bardi, Sandro La Bruzzo, Franco Zoppi (CNR)
Reviewers	Paolo Manghi (CNR), Sjoerd Siebinga (IISG)

HOPE is co-funded by the European Union through the ICT Policy Support Programme

Revision history

Version	Date	Authors	Modifications
1.0	31/03/2011	Alessia Bardi, Sandro La Bruzzo, Franco Zoppi (CNR)	First release of the deliverable

Table of contents

Figures

1. Introduction

The aim of this deliverable is to explain how the D-NET System software - as developed by DRIVER[1] - was customized to meet the requirements of the HOPE System Architecture [HOPEHLD] for the implementation of the HOPE Aggregator.

The customization activity consists in:

- The evaluation and testing of existing D-NET assets, namely the set of services outcome of the DRIVER project, to realize and adapt those needed to implement a federation of repositories compliant with the requirements.
- The configuration of existing services for the specific needs of the HOPE project.
- The design and implementation of new services.

Examples of the customization activity consists in the personalization of the look and feel of the user interface, the implementation of wrappers and harvesting services tailored to specific operational contexts, the customization of the search functionality as to serve diverse user needs and the adaptation of harvesting and aggregator services to deal with the HOPE data model.

Section 2 introduces the most relevant characteristics of the D-NET architecture components and services, while Section 3 provides a description of the D-NET adaptation to the HOPE scenario focusing on those components and services which have been configured, customized or developed to fulfil the HOPE requirements. Section 4 and Section 5 respectively introduce the Aggregator Implementation and Test Plans to formalize the milestones MS24 and MS25 as defined by the HOPE Description of Work.

Finally, Appendix A is a Glossary of the technical terms used throughout the document, and Appendix B summarizes the workflow for submitting and curating metadata with the HOPE Aggregator.

[1] The DRIVER Infrastructure. http://www.driver-community.eu

2. D-Net Infrastructure

2.1. Preliminary assumptions

The main goal of the D-NET System is to enable an infrastructure which must be scalable and flexible with respect to the number and type of participating repositories, capable of offering different views to different communities of users, extensible with new services, and dynamically configurable in order to maximize Quality-of-Service.

In the following a high-level view of the architecture is presented by:

- Formalizing technical preliminary assumptions.
- Defining the D-NET Information Space: the System supports an Information Space, which hides to end-users the heterogeneity of the aggregated Repository Service Resources.
- Identifying the Services designed and implemented in D-NET.

The following aspects characterize the Architectural Specification of the System:

- The System is based on a Service Oriented Architecture (SOA) implemented by Web Service (WS) Technology[2].
- The architecture employs the use of industry standard interfaces, where available, for communication among the software components. This supports the greatest possible interoperability with third party software and openness to future services.
- The system must allow reusing existing assets[3] as much as possible.

2.2. Service-Oriented Architecture

The service-oriented architecture approach is a way of building distributed systems that deliver application functionality as services to either end-user applications or other services. In particular, Web Services technology offers a natural, interoperable means to expose software functionality as a service to build an SOA. It provides a distributed computing approach for integrating heterogeneous applications over Internet making use of open technologies such as XML[4], SOAP[5], REST[6], UDDI[7], and WSDL[8].

The D-NET System was designed by relying on the Web Service technology. This decision, of course, strongly impacts on the technological solution chosen for HOPE. The main consequence is that the D-NET nodes forming the infrastructure must be equipped with a service container capable to host and support Web services. However, in order to facilitate

[2] W3C Web Services Activity web site. http://www.w3.org/2002/ws/

[3] An asset is intended here as any form of expertise or software which can be provided by the partners and has a potential to earn revenue.

[4] Extensible Markup Language - http://www.w3.org/XML/

[5] Simple Object Access Protocol - http://www.w3.org/TR/soap/

[6] REpresentational State Transfer - http://en.wikipedia.org/wiki/Representational_State_Transfer

[7] OASIS UDDI Specification Technical Committee - http://www.oasis-open.org/committees/uddi-spec

[8] Web Service Definition Language - http://www.w3.org/TR/wsdl

the re-use of partner's expertise and software, the project does not impose a specific Web Service hosting environment implementation, but only the use of WSDL and SOAP/REST as the communication standards.

2.3. D-NET Information Space

The D-NET System aims at constructing Aggregations of Repositories. In practical terms, the System allows existing repositories, matching D-NET guidelines and requirements, to deliver their content to larger communities of end-users through personalized services and interfaces. Such content is heterogeneous, both in the nature, i.e. semantics, and in the structure it conforms to, i.e. the Repository Object Model of reference. For example, repositories may contain any kind of material, raw data, software, pictures, videos, publications, tutorials, multimedia, and may conform to various object models, including OAI-Items (descriptive models), DIDL (complex objects), etc.

The D-NET Repository Infrastructure aims at collecting such heterogeneous content and aggregating it to form a uniform Information Space, which delivers the original data sources through the same interpretation, i.e. object model. The model describes D-NET objects as representations and "place holders" for objects collected from different repositories, by providing a way to uniformly describe their properties of "collected objects". D-NET objects are created at harvesting time by adding to the collected objects some system properties which summarise the description, structure, and content of the object; such summary depends on the Object Model to which the object obeys and is extracted from the object at collection time. D-NET objects' system properties are represented as XML fields compliant to the Descriptive Metadata Format (DDF).

D-NET Compliant Repositories must publish their content through the OAI-PMH protocol and contain toll-free, reachable textual files. D-NET System provides the Services to aggregate, i.e. harvest, clean and enrich OAI-PMH metadata records from several Repositories, and support end-users with uniform search interfaces to the heterogeneous content. In particular, for each repository, D-NET aims at aggregating the OAI-Items it contains, hence all metadata records relative to the OAI-Items therein. Since this does not apply to all of the HOPE repositories, the D-NET System was integrated as described in Section 3.2.1.

2.4. D-NET Services

The D-NET System provides the services needed to activate an open service infrastructure, over which services can offer functionalities by interacting with each other. D-NET uses the infrastructure to provide services to collect heterogeneous content from external repositories, to store and index such content, and to allow users and applications to access such content uniformly.

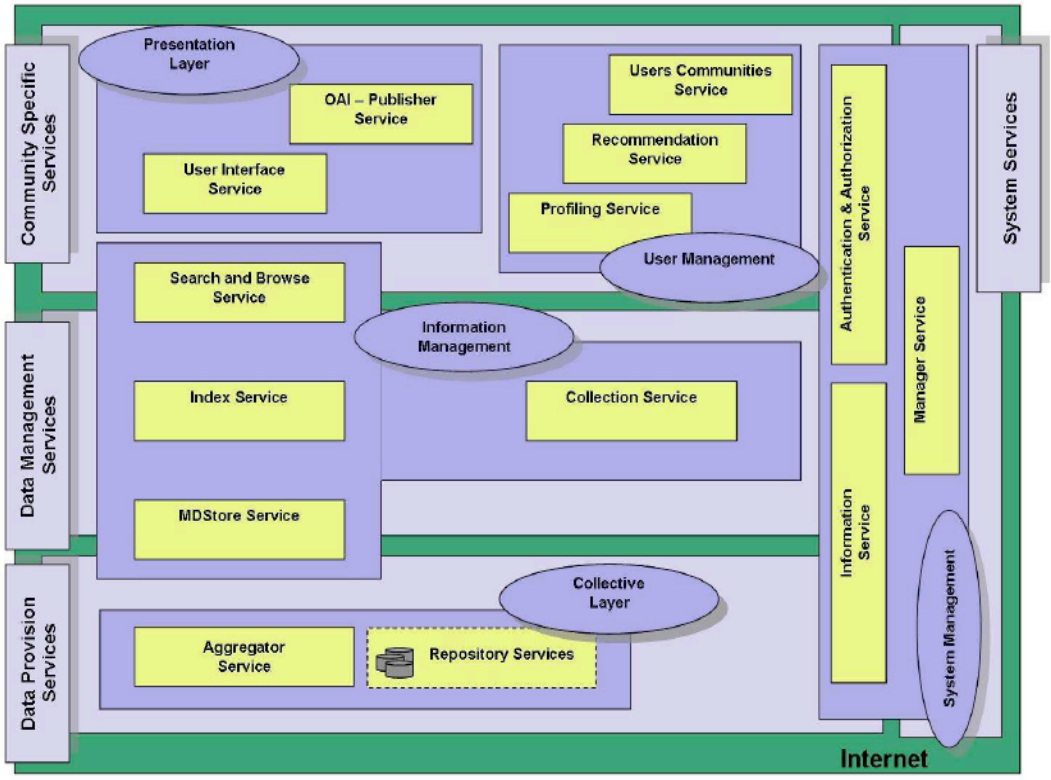

Figure 1 - The D-Net Framework

Figure 1 illustrates the D-NET Services grouped by functional area and by application groups. Groups put together services belonging to separated functional areas, to show when these cooperate to deliver the same application functionalities, here broken down into four main areas:

1. Data Provision Services: Services dealing with data collection and aggregation; Aggregation Services collect content from external repositories to form the D-NET Information Space.
2. Data Management Services: Services offering data storage, indexing, and search functionalities.
    a. D-NET Objects are stored within Metadata Storage Data Structure Resources (MDStores), to be created by MDStore Services.
    b. Metadata records, stored into MDStores, can be indexed by Index Data Structure Resources to be created by Index Services.
    c. Collection Services allow for the definition of Collection Data Structure Resources; Collections are virtual sets of D-NET Objects, specified by means of a predicate query over metadata fields.

d. Search Services accept predicate queries, as well as collections, over metadata fields and return the records matching the predicate; Search Services find such objects by running look-ups over the appropriate index resources and combining their results.

3. Community Specific Services: Services for capturing community-specific user and application needs.

    a. Presentation Layer: includes the Services used by users and applications to interact with D-NET Information Space.

        i. User Interface Services allow users to specify and run queries over the D-NET Information Space; User Interfaces interact with Search Services.

        ii. OAI-Publisher Services allow applications to interact with D-NET Information Space as if it was an OAI compliant repository.

    b. User Management: includes the Services required to ease the interaction with the Information Space and provide advanced functionalities to registered users.

        i. Profiling Services allow users to register to the system in order to exploit advanced functionalities: users can register to Communities and specify their specific topics of interest in order to be recommended for new content and make use of System query profiling functionalities.

        ii. User Community Services allow for the definition of communities as sets of collections to which registered users can be subscribed; communities are created to provide higher views of the Information Space, fragmented into collections.

        iii. Recommendation Services provide registered users with notifications about the addition of interesting objects into the Information Space.

4. System Services: infrastructural Services, thanks to which D-NET test-bed Services can interact and provide functionalities.

    a. Information Services keep track of all services registered to the System and provide subscription and notification mechanisms to the services.

    b. Manager Services are tailored to the types of services that can join the system and are entitled to orchestrate and monitor their behaviour, in order to maximize system quality of service.

    c. Authorization and Authentication Services provide the security mechanisms that guarantee that services are used by authorized users and services.

The service boxes in Figure 1 are not necessarily to be considered as single software entities, each implemented by a unique web service. Each of them represents instead a meta-service, i.e. a set of related real services acting as a whole to supply the functionality associated with the meta-service. A meta-service, during the design phase, is decomposed

into the set of its corresponding and cooperating real-services.

2.5. D-NET System work flow

D-NET Services provide specific functionalities on request from other services or users. The Manager Service orchestrates the available services in order to make them fully functional with respect to the expected functionalities: D-NET users and applications respectively expect to be able to query and harvest an Information Space made of all objects collected from the repository services.

Specifically, repository services register to D-NET Information Service, thereby expressing the will to share their content. The Manager Service reacts automatically to the insertion of a new repository by creating the resources needed to make the repository content available to the system. More specifically, the association between a repository service and the MDStores that will host its content is itself a system resource, called Harvesting Instance. Hence, the Manager Service creates a new harvesting instance, together with the relative MDStores, and assigns the former to an available Aggregator Service, which will independently handle the activity of collecting items from the Repository. Similarly, the manager service needs to ensure that the new MDStores are targeted by at least one Index. To this aim, the service needs to search for available Index Data Structure resources or potentially create new ones.

The D-NET framework enables the combination of services into workflows, to obtain complex and personalised data processing operations. To this aim, the services are designed to exchange XML records through mechanisms offered by D-NET ResultSet Services. *ResultSets are* "*containers*" for transferring list of files between a "provider" service and a "consumer" service. Technically, a ResultSet is an ordered lists of files identified by an End Point Reference (called EPR, the Web Service EPR standard describes the location of a resource on the Internet), which can be accessed by a consumer through paging mechanisms, while being fed by a provider. D-NET services can be designed to accept or return ResultSet EPRs as input parameters or results to invocations, in order to reduce response delays and limit the objects to be transferred at the consumer side to those effectively needed. For example, while the response to a full-text query may consists of thousands of rank-sorted results, the consumer often requires to access tens of them.

2.6. Aggregator Service - Processing of metadata records

The D-NET Aggregator Service (AS) supports all the functionalities in relation to the harvesting of external Repository units and the aggregating process of cleaning and enriching the results. As final point the AS stores the outcome of these steps into D-NET Information Space. We can identify two main component of the Aggregator Service: (i) the Harvester Service, in charge of the harvesting phase, (ii) the Transformation Manager Service, in charge of the aggregation phase. Harvester and Transformation Manager Services are described in major details in the next sub sections.

The Aggregator Service also keeps a local log about each single harvesting-aggregation activity, hence relative to the withdrawal of records from one Repository Service according to

one metadata format, and the subsequent aggregation and storage into the assigned MDStore. Such logs are used to update Harvesting Instance profiles with information relative to the individual MDStores, such as the last date of harvesting, to be used by the Aggregator Service to configure the next incremental harvesting operation.

Finally, the Aggregator Service is also in charge of keeping Repository statistics, to be requested by the Manager Service to update information into the Repository Services available.

The design of the Aggregator Service, as illustrated by the Class Diagram in Figure 2, contemplates the following components:

- The **AS-HIMap** (Harvesting Instance Map) component offers the method to add, remove, and update a *Harvesting Instance entry* (HI-entry) into and from a local store. An HI-entry replicates the information contained into the Harvesting Instance, and contains cleaning-enriching rules, plus statistical information about the harvesting-aggregation process and about Repository content.
- The **AS-HIMan** (Harvesting Instance Manager) component offers methods and a user interface to configure Harvesting Instances parameters.
- The **AS-Engine** component is divided in two subcomponents: **AS-HISche** (harvesting Instance Scheduler) relative to the scheduling of Harvesting Instance, i.e. the activation of the relative harvesting-aggregation processes; and **AS-HIRun** (Harvesting Instance Runner), concerned with the actual execution of such a process.
- The **AS-RIP** (Repository Information Provider) component offers the methods to search for statistics about Repository content.
- The **AS-NH** (Notification Handler) component handles notifications to the local Service.

2.6.1. Harvester Service

The Harvester Service manage Harvesting Instance Resources, each representing the harvesting process of all OAI-Items of a specific Repository Service; where the harvesting of an OAI-Item is here interpreted as the harvesting of all the metadata records of the OAI-Item.

In particular, the Manager Service assigns Harvesting Instances to an Aggregator Service. Before getting started, the activity of harvesting-aggregating relative to a Harvesting Instance requires the intervention of specialized users, namely the Aggregator Managers, who must configure the following aspects, crucial to both harvesting and aggregation phases:

- Scheduling of Harvesting Instance: time interval to wait before next harvesting.
- Type of harvesting: incremental, with regards to the last date of harvesting, or refresh. Incremental harvesting allows the withdrawal of all records published after a certain date, if any; refreshing is the operation of harvesting the whole Repository content, and typically requires the dumping of previously harvested records.

This information, which must be provided for each Harvesting Instance under the Aggregator Service responsibility, is stored within Harvesting Instance profiles.

2.6.2. Transformation Manager Service

Transformation Manager Service manages Transformator Data Structures, each representing the aggregating-transformation processing of harvested records. In particular each Transformator Data Structure is assigned to one input and one output MDStore and deals with the transformation of metadata records from one data model into records of one output data model.

The aggregating activity is started by Aggregator Administrators, who must configure the logic of the transformation and harmonisation phases. The logic of the transformation, called "mapping", is expressed in terms of a rule language offering operations such as:

- Field removal, addition, concatenation and switch.
- Regular expressions.
- Invocation of an algorithm through a Feature Extractor Service.
- Upload of full XSLT transformations. User interfaces support administrators at defining, updating and testing a set of mappings.

A transformation request is thus composed by: the input metadata format, the Endpoint Reference (EPR) of the input ResultSet, the output metadata format, and the reference to the mapping to be applied. If the mapping is not available, the transformation is left pending until the Aggregator Administrators will provide one. The result of a transformation is the EPR of a ResultSet that contains the generated metadata objects.

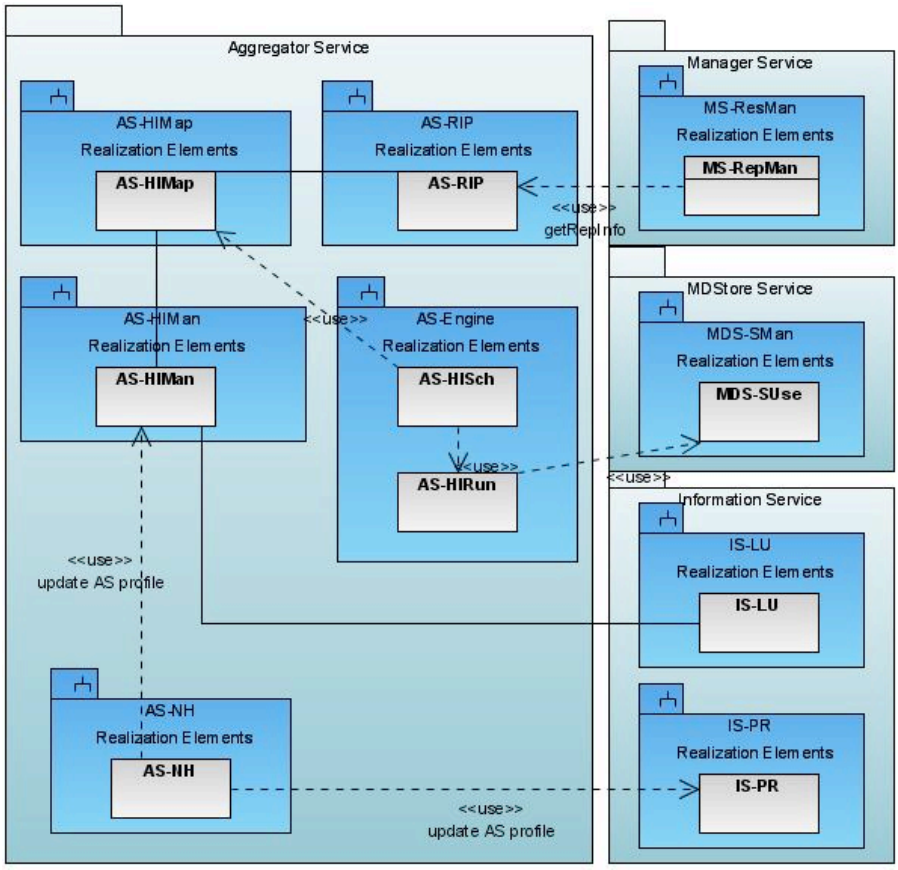

Figure 2 - Aggregator Service

2.7. Information Service

In D-NET Resources can interact directly or indirectly: in the first case, in order to accomplish its computational tasks, a Resource needs to interact with another Resource, of a given Resource Type; in the second case a Resource is notified of an action executed by another Resource.

An Information Service (IS) provides the functionality required by Resources to interact with each other, by offering to Resources mechanisms for:

- Subscribing to specific actions of specific Resources, in order to be notified of the occurrence of such actions.
- Finding the Resources, i.e. their profiles, they need to interact with.

Accordingly, the IS maintains:

- A *Subscription and Notification* table, which associates subscribed Resources to specific *topics*, i.e. specific actions on Resource profiles and Resource Types; the subscription and notification table, as well as the relative managing interfaces, are

built in order to support the OASIS Standards WS Base Notification 1.3.

- An updated version of all Resource profiles, organized into Resource Kinds. Resource profiles must conform to a Resource Type, i.e. to the relative XML Schema, and are logically assigned to the Resource Kind of the Resource Type. Both information, i.e. Resource type and kind of a profile, must be described into the profile itself, hence described in the Resource Type XML schema. System Resource kinds are: *ServiceResources, D-NETPendingServices, HarvestingInstanceResources*, *IndexResources*, *MDStoreResources*, *RecommendationResources*, *ResultSetResources*, *UserResources*, *CommunityResources*, *CollectionResources*.

The IS offers mechanisms to manage Resource Types, hence the addition and removal of pairs Resource Type and relative profile XML Schema. During the addition of a new Resource Type, the relative XML Schema must specify the association of the Resource Type with at least one Resource Kind and the subscription and notification table must be updated with the new topics, relative to the Resource Type.

2.8. Manager Service

The Manager Service (MS) monitors System Quality of Service (QoS) by elaborating the Resource profiles into the Information Service. QoS depends on efficacy and efficiency:

- *Efficacy*: it is measured according to two System aspects:
  - Minimum number of Service Resources available for each Resource Type (including replicas);
  - Number of Data Structure Resources needed for the System to work for each Data Structure Resource Type: such numbers depend on specific System management policies described below.
- *Efficiency*: efficiency depends on Resource workload; the System calculates it from the performance parameters updated by the Resources themselves and reacts based on the *System management policies* described below.

Since the System cannot create Service Resources, the Manager Service may improve efficiency and enforce efficacy, by creating, deleting, or updating Data Structure Resources. Specifically, the Manager Services orchestrates Resources by interacting with the Resource profiles into the Information Service according to the Orchestration protocol described below.

2.8.1. System management policies

D-NET System logic is based on the following policies, which must be enforced in order to provide the required efficacy threshold:

- Efficacy
  - For each Repository Service there must be
    - One Harvesting Instance.
    - One Master MDStore for the harvesting of input metadata records.

        - A number of MDStores, one for each output metadata format.
    - Each D-NET record into an MDStore Resource chosen for export must be targeted by at least one Index Resource.
- Efficiency
    - Distribute the number of Data Structure Resources among the available Service Resources than can handle them. Such policy must be enforced based on thresholds and cost trade-offs.

2.8.2. Orchestration Protocol

The Orchestration Protocol is based on the following two assumptions:

- Service Profiles contain a section dedicated to Manager Service messages, called *blackboard*.
- Services, at registration time, also subscribe to the update of their own profile blackboard (see top picture in Figure 3).
- The Manager Service is subscribed to the update of all Service profiles' blackboards.

Figure 3 illustrates the steps of the Orchestration Protocol. When the Manager Service decides to communicate with a Service it updates the relative profile blackboard in the IS with a message (step A). The relative Service (step B), is notified of the change via the subscription and notification mechanism of the Information Service with a notify message to the Service's Notification handler component. The notify message has the following form:

```
notify (subscrId, UPDATE.serviceId.blackboard,IsId, message)
```

Message parameters contain the blackboard of the Service profile, hence the message to be interpreted.

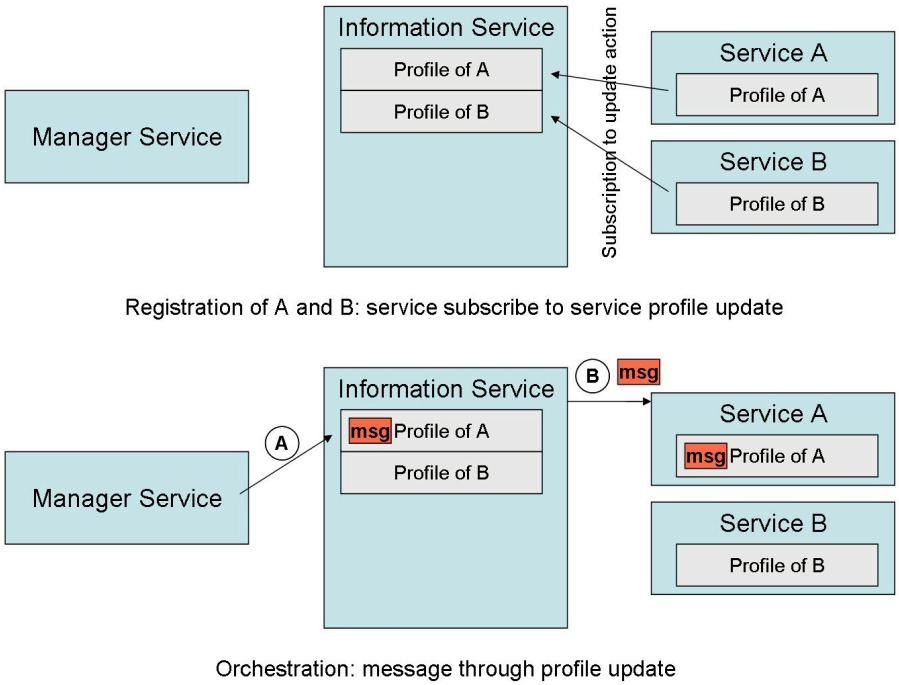

Figure 3 - Manager Service Orchestration protocol

The Manager Service may also need to send an action to a Data Structure Resource, in order to change its behaviour. In this case, the action will be sent to the Service responsible for the action, which will execute it over the target Data Structure.

When Services are delivered an action they must answer to that action with a further profile blackboard update. This will allow the Manager Service to check whether the original action was successful or not, and activate a repairing process if necessary.

Manager Service actions and Service answers are triples of the form `(actionId, subject, parameters)`, where:

- `actionId` is the unique identifier relative to the action: it is released by the Manager Service and used to associate the answers left by the Service to the action that caused them.
- `subject` can be relative to an action, such as CREATE, DELETE, UPDATE, MANAGE, RELEASE or CANCEL ACTION, or relative to an answer, such as ONGOING and DONE. ONGOING is the answer to actions that may take a long time, such as the creation of an Index Resource; at operation termination the Service leaves a message DONE to the action. In those cases where the action is taking too long, the Manager Service may decide to send a CANCEL ACTION, which should fire a recovery operation at the Service side.
- `parameters` may be required in an action to either specify the parameters needed to

create or update a Data Structure Resource and to specify the identifier of the Data Structure to be removed; or in an answer to return the identifier of the Data Structure created with an action.

2.9. Metadata Storage Service

MDStore (Metadata Storage) Services are responsible for the management of MDStore Data Structure Resources. In particular, MDStore Services:

- Create, delete, update MDStore Resources.
- Store metadata records harvested-aggregated from Repository Services into MDstore Resources.
- Deliver metadata records into a MDStore Resource to a requesting Resource, namely an Index Resource or an OAI-Publisher Service.

The design of the MDStore Service, together with its relationships with the other D-NET Services, is shown in Figure 4. The Service is made out of three components:

- The **MDS-SMan** (Store Management) component contains two classes: **MDS-SDef** (Store Definition), which is in charge of creating, deleting and updating (not be confused with content-wise operations, but rather as "configuring") an MDStore Resource; **MDS-SUse** (Store Usage) which implements the functionality required to store and retrieve content into and from an existing MDStore Resource; the component also provides methods to interact with MDStore as if they were OAI-Compliant providers.
- The **MDS-NH** (Notification Handler) component is in charge of alerting the MDS-SMan component of Manager Service actions and of the update of Repository Service profiles, which may involve the modification of the related MDStores.
- The **MDS-ResultSet** component handles the management of ResultSets, returned as results of OAI-metadata record delivery requests. The ResultSet Service is here implemented as a Service component for performance reasons.

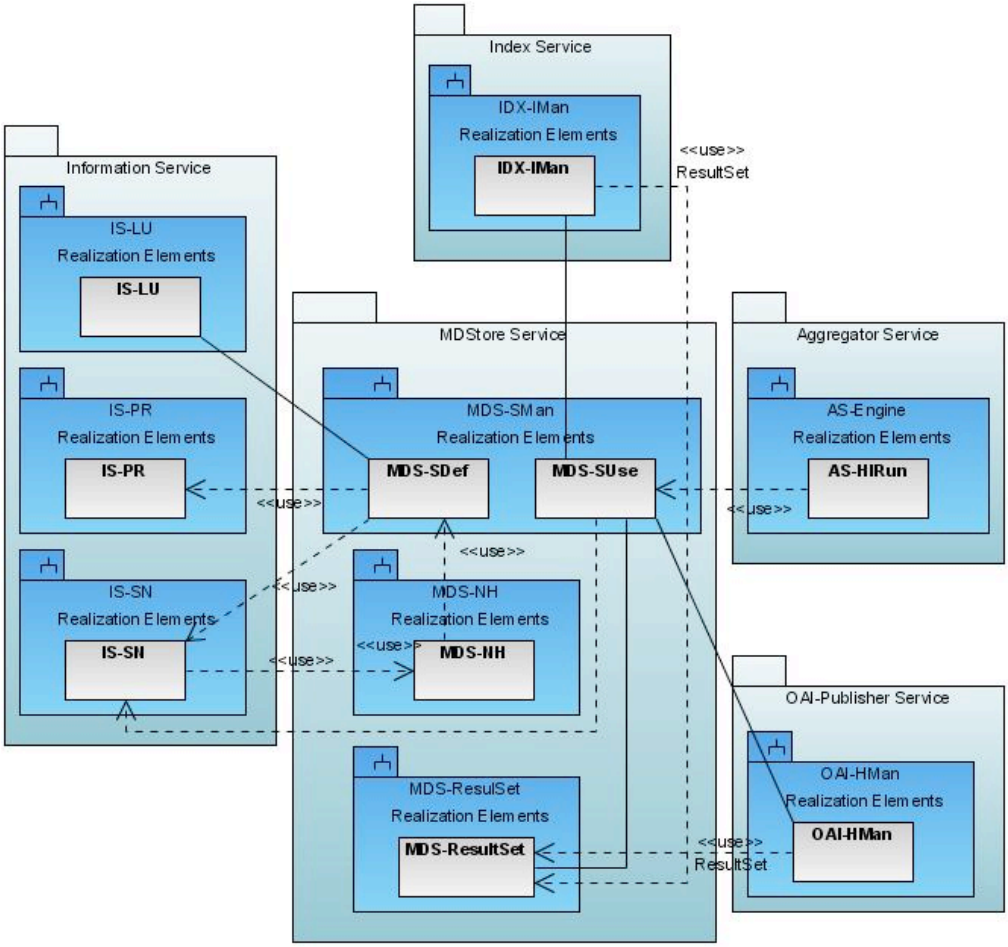

Figure 4 - MDStore Service

CQL query syntax supported by Storage Service

Field Search

The field searching can be executed by typing the field name followed by an equal "=" and then the term which is looking for. Example:

```
query=format=xml
```

There is also the possibility to use 'more' or 'less' operators instead of 'equal'. For example in case of dates the query can be as following:

```
query=lastModificationData>"2008-01-14"
```

```
query=lastModificationData<"2007-01-14"
```

The following fields are available for searching in Lucene Store Data Structure:

- format
- originalName
- mimeType

- originalSize
- lastModificationData
- lastAccessionData
- ownerName
- sourceName

Search with Boolean operators

Boolean operators allow terms to be combined through logic operators. Supported Boolean operators are: AND, OR, NOT. Example:

```
query=format=xml AND format=pdf NOT format=doc
```

Please note that Boolean operators must be all CAPS.

Wildcard Search

Multiple character wildcard searches within single terms (not within phrase queries) are supported. To execute a multiple character wildcard search use the "*" symbol. Example:

```
query=originalName=tes*
```

This will return all records that contain: test, tester, testing, tested.

Range Search

Range queries allow finding documents whose field values range between the lower and upper bound specified by the query. Examples:

```
query=lastModificationData within "2005-01-01 2005-01-05"
query=lastModificationData within (2005-01-01 2005-01-05)
query=lastModificationData within  2005-01-01 2005-01-05
```

2.10. Index Service

The D-NET Index Service (IXS) implements the functionalities in the Index Management area, regarding the management of Index Data Structure Resources, i.e. definition and usage. Index Resources provide fast access to MDStore Resources, through the functionalities typical of information retrieval indices, and are used by Search Service functionalities to improve query performance.

An Index (factory) Service manages a set of Index units capable of indexing metadata records of a given data model, i.e., XML format, and replying full-text CQL queries (Contextual Query Language) over such objects. Consumers can feed units with records, remove records or query the records. The Index Service replies to CQL queries by returning the Endpoint Reference of a ResultSet that contains the result. Moreover, the service supports advanced full-text highlighted search and faceted browsing functionality. The

Service is implemented on Solr[9].

As shown in Figure 5, the Index Service is made out of three components:

- The **IXS-IMan** (Index Management) component contains two classes: **IXS-IDef** (Index Definition) is responsible for the definition of Index Resources; **IXS-IUse** is responsible for the usage of Index Resources, i.e. querying Index Resources as well as feeding them with new metadata records;
- The **IXS-ResultSet** component handles the definition of ResultSets for Index Resource look-up queries. The ResultSet Service is implemented as an IXS component for performance reasons.

9 Solr Apache Lucene Project: http://lucene.apache.org/solr/

- The **IXS-NH** (Notification Handler) component manages the notifications reaching the Service and the single Index Resources.

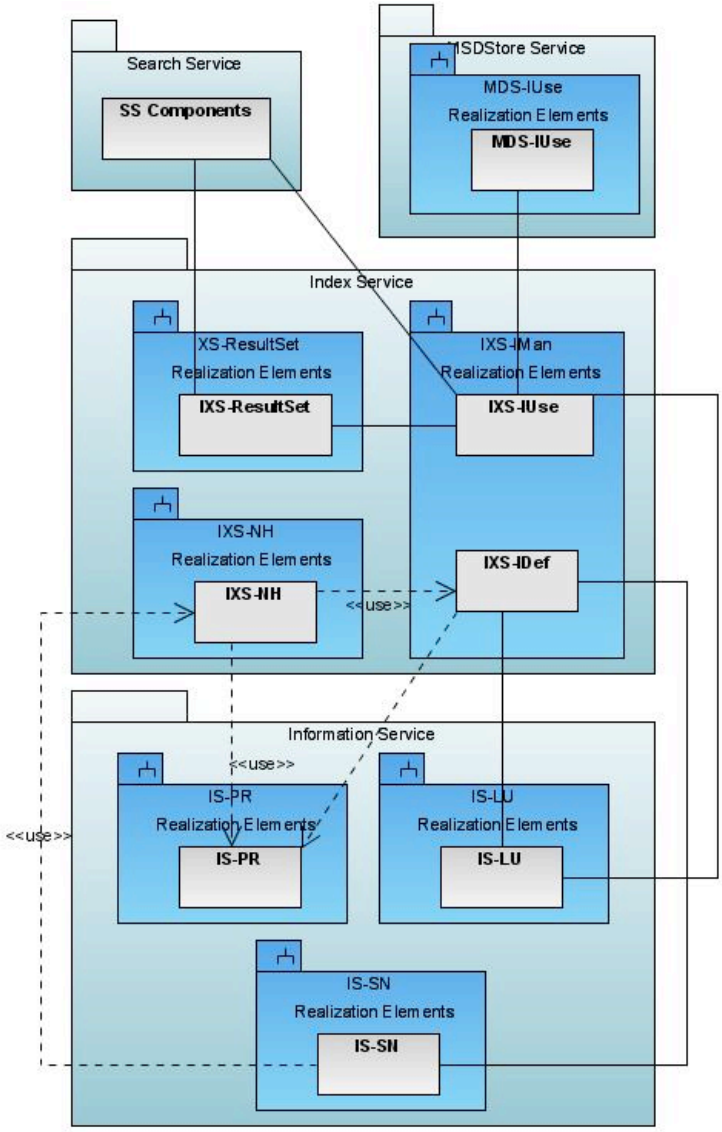

Figure 5 - Index Service

2.11. User Interface Service

The User Interface Service (UIS) serves as the front end of the D-NET system. It handles all incoming end user requests and dispatches them to the appropriate D-NET Service. It manages the browsing and searching (simple and advanced) on both D-NET Information space and on IS entities (communities, collections), the user profiling editing and set up, and all web based recommendation notifications.

Since the layout and the various functionalities provided by the User Interface Service rely on the definition and description of the Descriptive Metadata Format (DDF) used internally by the D-NET Framework (see Chapter **Error! Reference source not found.**), it is important that the UI Service is always up to date on all changes occurring within the D-NET system configuration.

2.12. Metadata Export

2.12.1. OAI-Publisher Service

The OAI-Publisher Service exposes to external applications all functionalities established by the OAI-PMH 2.0 protocol to access metadata objects in the MDStore units. To this aim, the service dynamically discovers and exposes through the ListMetadataFormats verb the list of metadata formats currently available in MDStore units and offers a getRecord operation over all the MDStore units hosting such records.

The Service design comprehends three components:

- The **OAI-PMH** component, which exports all OAI-PMH functions to external applications;
- The **OAI-HMan** (Harvesting Management) component, which implements the OAI-PMH function calls by interacting with D-NET System Services and the local ResultSet component.
- The **OAI-ResultSet** component, which builds results to OAI-PMH calls requiring resumption tokens. Such results include lists of metadata records and lists of OAI Sets.

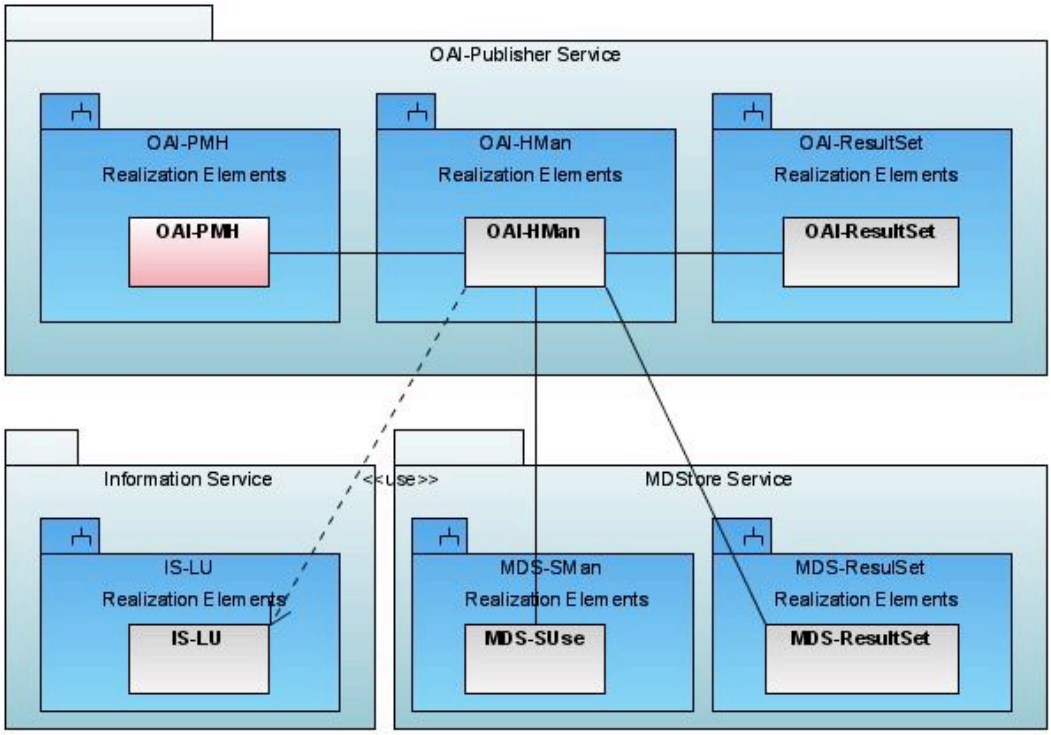

Figure 6 - OAI-Publisher Service

In D-NET the OAI-PMH protocol returns records of all metadata formats that have been stored in at least one Index Data Structure.

2.12.2. Search Service

The D-NET Search Service (SS) provides functionality for evaluating search queries issued by its D-NET end-users. The Search Service complies with the SRW/CQL (Search/Retrieval via URL) interface through which it is possible to run a CQL query over the Index Resources. Queries can be of two types: *document queries* or *aggregate queries.* End-users are typically interested in document queries, where the response is a set of ranked documents - more precisely metadata records - that match the query criteria.

Synopsys for Document Queries:

```
<someCQLquery> [sortBy <fieldName>/sort.(a|de)scending]
```

Examples:

1. `title any Pinocchio`: search for metadata records whose field `title` contains the term “Pinocchio”;
2. `all any fish`: search for metadata records with at least one field containing the term “fish”;
3. `title any Pinocchio date/sort.descending`: search for metadata records

whose field `title` contains the term "Pinocchio" and return them in descending order by date;

4. `title any Pinocchio date/sort.ascending`: the same as query 3. but in ascending order;
5. `subject = "biology" or subject = "chemistry" sortBy date/sort.descending`: search for metadata records whose field subject is equal to "biology" or "chemistry" and return them in descending order by date.

In contrast, aggregate queries do not consider individual documents, but report the number of occurrences of specific field-value pairs. Therefore, the response of an aggregate query is formed in a multi-column table that lists the value for the requested fields in each column and the respective count of occurrences in the last column. Although document queries are the main search tool in the D-NET system, aggregate queries are equally important because they allow for two functionality features:

- Aggregate queries form the basis of the browsing functionality. For example, a user may be prompted with a list of authors and the respective number of publications to select from.
- The Search Service uses aggregate queries in order to query index statistics from an Index Service. Statistical information from indexes is a key feature for efficient query execution.

Synopsis for Aggregate Queries:

`<CQL query>&groupby=<list,of,comma,separated,metadata,fields>`

Examples:

1. `title any Pinocchio&groupby=publisher`: returns, for each different `publisher`, the number of metadata records matching the query;
2. `title any Pinocchio&groupby=publisher,language`: returns, for each different couple <`publisher`, `language`>, the number of metadata records matching the query.

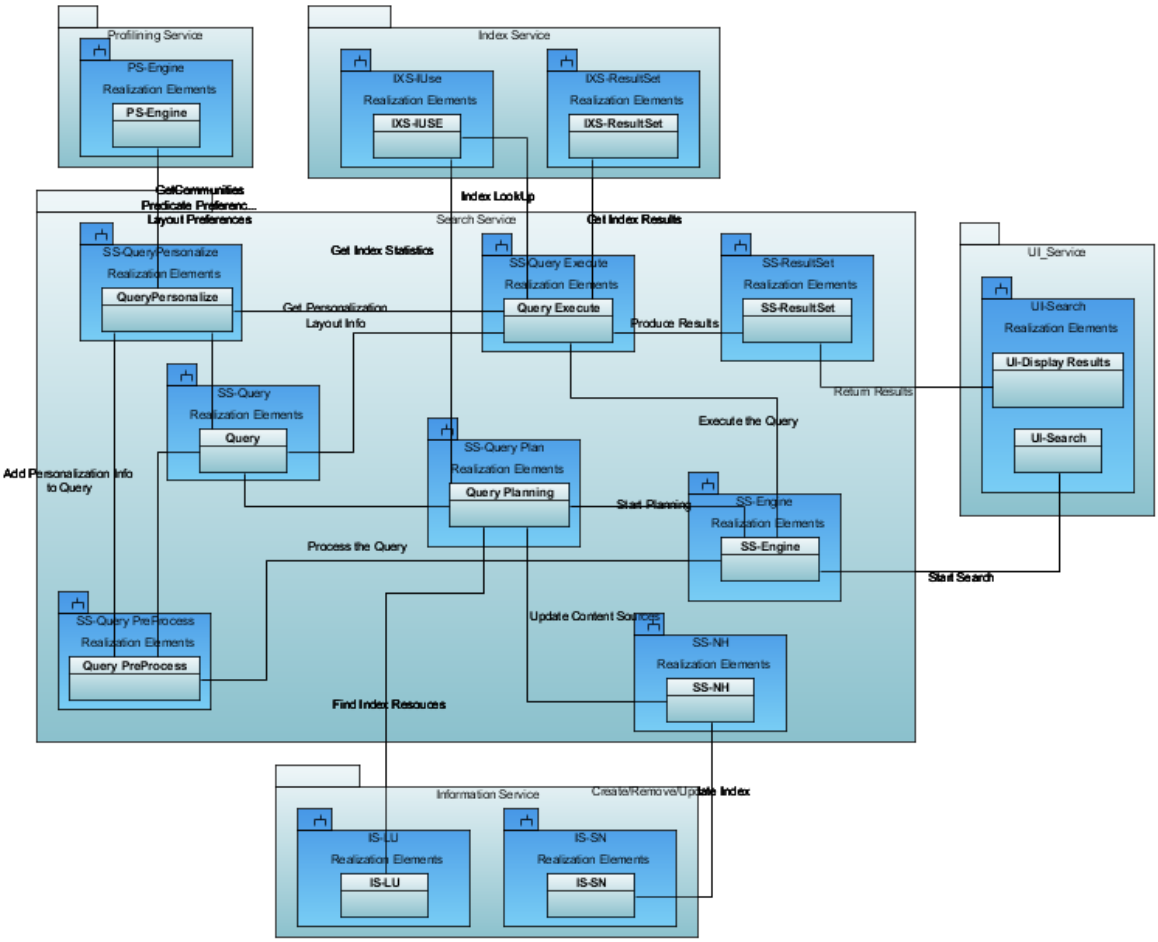

Figure 7 - Overall Search Service Architecture

2.13. Metadata Curation Tools

HOPE curators (often a group of "experts in the field" selected across the content providers) may be willing to perform further semi-automatic cleansing activities. To this aim, the following D-NET Services will be used:

- Authority File Service
- Metadata Editing Service
- Metadata Tagging Service

2.13.1. Authority File Service

The Authority File Service – designed and currently under development – implements

functionalities for "curating" a set of authority files, intended as sets of authoritative metadata records. Note that authority files are typically kept, fed, administrated separately from the core data that depend on them, which has to be kept synchronized to the possible updates occurring to the authority files of reference. In particular, administrators can:

- Create an authority file, by providing the relative metadata data model.
- Create, delete, edit metadata records in it.
- Use algorithms for the identification of candidate pairs of duplicate records.
- Merge a pair of candidate records into one, and "split" metadata records, i.e., obtaining two records from one.

When finished, administrators can "commit" changes and generate a new version of the authority file. Each version is accompanied by a report file, which contains the list of merge or split operations committed in the file and that can be exploited by consumers (D-NET services or external applications, see Metadata Editor Service) to upgrade the data in datasets making use of the authority file according to the latest updates. Consumers can feed an authority file with new metadata records (by sending an EPR of a ResultSet with new metadata records) or access an authority file, which is returned into a ResultSet through an EPR.

2.13.2. Metadata Editing Service

The Metadata Editing Service is a new D-NET Service – designed and currently under development. Its main functionalities deal with enabling a set of authorized users (experts) to edit existing metadata (or add new metadata records) in the chosen metadata format.

2.13.3. Metadata Tagging Service

The Metadata Tagging Service is a new D-NET Service – currently under design. Its main functionalities deal with the association of tags to sets of metadata records in order to create virtual collections of data. Tags should be carefully chosen based on the application context.

3. HOPE Implementation

3.1. General concepts

The HOPE Aggregator Service is realised by means of the D-NET Software Toolkit [D-NET]. The D-NET Toolkit offers a service-oriented framework where developers can build applications by combining a set of D-NET Services.

Furthermore, the framework allows for the addition of new service typologies, in order to introduce new functionality, whenever this is required and without compromising the usability of other components. D-NET provides a data management service kit, whose services implement functionality for the gathering, manipulation and provision of XML records exported by a set of Content Providers. Such services have been realised and added in the years to meet the particular requirements arisen when facing new challenges in different application domains (e.g., DRIVER project, EFG project, OpenAIRE project). Most importantly, they are designed according to two engineering principles:

- *Modularity:* services provide minimal functionality and exchange long lists of information objects through the ResultSet mechanism (by relying on a ResultSet Service instance or by implementing natively the interface functionality of the ResultSet Service) so that they can be composed with others to engage in complex data management workflows.
- *Customizability:* services should support polymorphic functionalities, operating over XML records whose data model, i.e., XML format, matches a generic structural template. For example, the D-NET index service is designed to be customizable to index records of any XML format.

The HOPE Aggregator has three main functional objectives:

- Collecting the data from Content Providers (harvesting, transformation and storage).
- Curating the records (editing, cleaning, enrichment).
- Disseminating the records to third-party systems (pull or push).

In the following sections we describe the D-NET services to be used for the realisation of the three objectives and the implementation of the HOPE Aggregator. We shall see that, in order to satisfy HOPE requirements, in some cases D-NET services will have to be extended and new ones will have to be realised. Figure 8 shows a diagrammatic high-level representation of the services implementing the HOPE Aggregator.

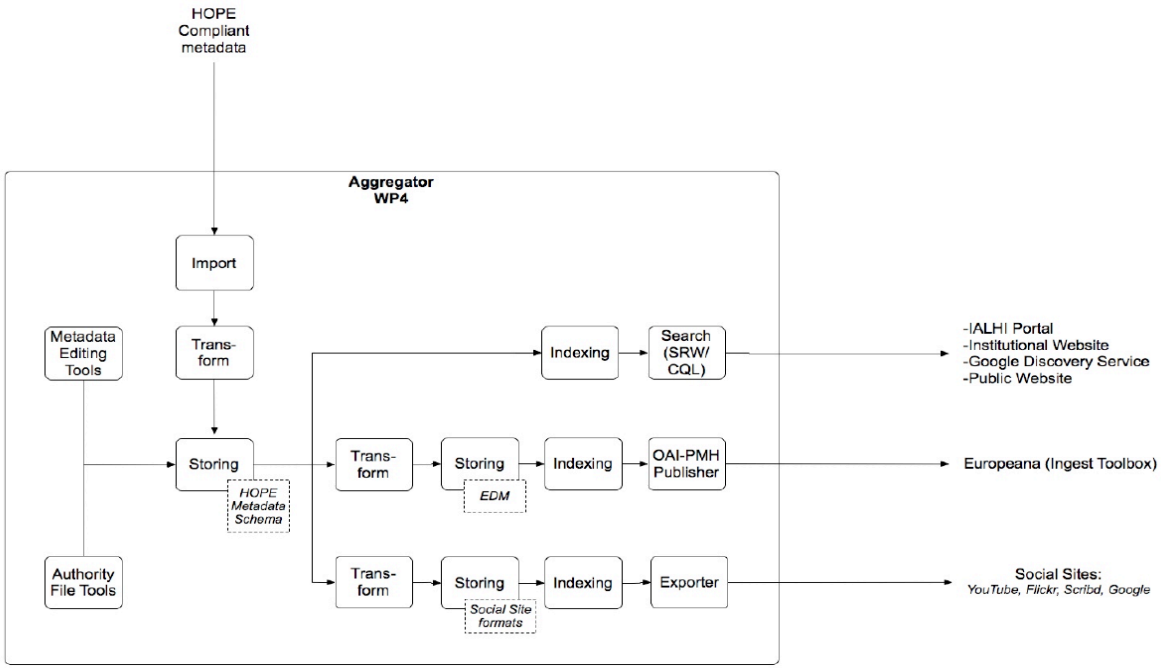

Figure 8 - The HOPE Aggregator Infrastructure

3.2. Aggregator Service - Processing of metadata records: harvesting and transformation into the HOPE schema

The Aggregator has the task of administrating a set of "authorised" Content Providers in order to harvest/ingest their records, transform them into the HOPE metadata format and store them locally.

Each functionality is implemented by an apposite service, following the principle of system modularity. Next subsections describe the services needed in the HOPE project to perform the harvest and transformation of HOPE compliant metadata records.

3.2.1. Harvester Service

The Harvester Service of D-NET currently supports OAI-PMH compliant Repositories. As mentioned in the description of the HOPE application scenario, some of the data sources may not provide an OAI-PMH publisher service. In order to properly manage these cases, the Aggregator Service offers a File System Harvester Service that can import XML files in a given remote/local (with respect to the Aggregator) file system folder.

Content providers will thus export their metadata in the form of XML records through OAI-PMH protocol APIs or through a file export on a remote/local folder.

Figure 9 shows the services composing the harvesting module.

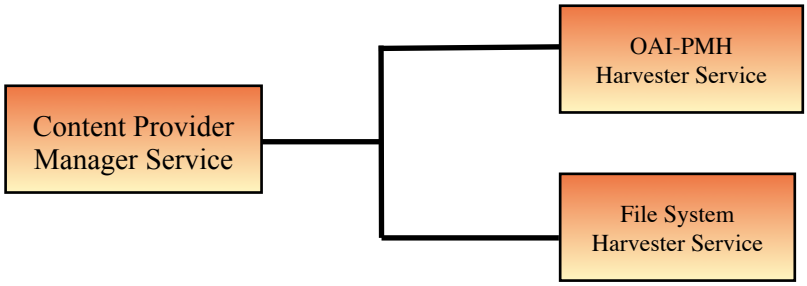

Figure 9 - The Harvesting Module

Content Provider Manager Service: CPs registers to the Aggregator providing some informaton that will end up in a Content Provider's "profile" in the D-NET system. A CP must then register one or more data-sets from which the content they are willing to share in the HOPE Information Space can be gathered from. The information needed to register a data-set is: access typology (OAI-PMH, FTP, SSH), access URI, input metadata format, target domain profile (one of: library, archive, visual, audio/visual, DC-simple) and the collection name.
If the access typology is OAI-PMH, CPs can also specify a particular OAI-Set, telling the Aggregator to harvest content only from that OAI-Set and not from all OAI-PMH repository. t

The service offers user interfaces for the registration (by CP administrators) and the subsequent administration of the CPs and their data-sets (by HOPE Aggregator Administrators, who can fire harvesting, manage transformation mappings, etc.).

- *OAI-PMH Harvester Service:* a Harvester Service can execute the six OAI-PMH protocol verbs, and communicate with a given data source registered to the system. In particular, the verb *ListRecords* fetches from the data source the metadata records of a given metadata format (e.g., oai_dc) that belongs to a particular OAI-PMH set and returns the EPR of a ResultSet that contains them. In case of non well-formed XML records, the OAI-PMH Harvester Service fails and the HOPE Support Team will inform the interested CP about the problem. In case of non valid XML records, no error occurs but it is likely that non valid records will not be processed in subsequent work flows performing XML transformation, since transformation stylesheets are based on the schema declared by the CP.
- *File System Harvester Service*: a File System Harvester Service can import XML files in a given local/remote file system folder. In particular, a "download" call returns an EPR of a ResultSet that contains such files. The File System Harvester Service skips non well-formed XML records (no failure, but a log is kept about the skipped files). In case of non valid XML records, no error occurs but it is likely that non valid records will not be processed in subsequent work flows performing XML transformation, since transformation stylesheets are based on the schema declared by the CP.

- *Digital Resources Supply Service*: a new service created to meet the needs of CPs who are not able to enrich their metadata records with direct references to the described digital resources (i.e., files). The Digital Resources Supply Service allows Content Providers to submit metadata about their digital resources by creating new descriptive metadata records in the HOPE format (the digital resource entity) to supply to the Aggregator.

3.2.2. Transformator Manager Service

The Transformation Manager Services manage Transformator Data Structures (TDS), each representing the aggregating-transformation processing of harvested records. Figure 10 shows the workflow for the metadata transformation sub-system. The workflow is composed of two transformation steps: the first transforms records from the original format into a standard domain profile format; the second transforms records from the standard domain profile format into the HOPE format. Different Transformation Data Structures (TDSs) are used for each workflow step:

- *Domain Profile Transformation:* one TDS is responsible for "mapping" the metadata harvested by one data source into the domain profile defined in the content provider manager service during the registration of the data source. The transformation rules to apply have to be defined by the Content Provider. The records resulting form this first transformation are then stored in the target MDStore associated to the TDS.
- *HOPE Transformation:* one TDS applies a "static mapping" (a set of rules defined once for all by the HOPE consortium) from the records in the domain profile format generated in the previous step to the HOPE metadata format. The resulting records are stored in the target MDStore associated to the TDS.
- *Cleaning*: one TDS is responsible for cleaning the records in the source MDStore. The cleaning process applies on records in the HOPE metadata format. The TDS should apply vocabularies established by the HOPE consortium to clean the data. For this reason, each CP must provide a vocabulary-matching file on which terms used by the CP are mapped into terms of the established vocabularies.
- *Index*: HOPE XML metadata records collected (or obtained by transforming collected records) from content providers are indexed through the Index Service and then ready to be exported.

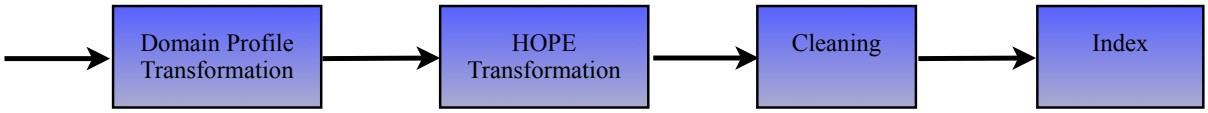

Figure 10 - The Transformation Modules

3.3. Manager Service HOPE customisation

The Manager Service is responsible for the orchestration of the different system components

in order to create an "application" - as HOPE is - on the infrastructure. This is why it has been the most involved by the customization. Main changes can be summarized as follows:

- New parameterisation capabilities, in order to enhance the configurability of the service; examples are: parameterisation of the MDStore format based on domain profiles and number of replicas for Indexes and MDStores.
- Design and implementation of new components for:
  - Management of the five HOPE domain profiles.
  - Processing of a generic XSLT file dynamically chosen.
  - Management of several data sets per Content Provider.
- Design and implementation of new workflows. The behaviour of the system had to be adapted to match the HOPE requirements by defining new workflows, namely:
  - Storage allocation compliant to HOPE, domain metadata format (EAD, DC, MARC21, LIDO, etc.) and EDM requirements.
  - Creation of specific indexes to support the EDM OAI export.
  - Mapping from the domain profile formats to HOPE.
  - Mapping from HOPE to EDM.
  - Export of metadata to social sites.
  - Export of metadata back to CPs.

3.4. Metadata Storage Service HOPE customisation

MDStore services had to be configured to host metadata records of different formats. MDStores are configured to store each XML record as an XML document in a BLOB field of the underlying database, together with its system identifier.

In particular, for each data-set exported by the CP, the following MDStores are needed:

1. One MDStore for the original metadata records in the input format (the local format of the CP).
2. One MDStore for metadata records in the domain profile format chosen by the CP for the corresponding data-set. This MDStore is fed by the Transformator Data Structure performing the first transformation step from local format to domain profile format.
3. One MDStore for metadata records in the HOPE format. This MDStore is fed by the Transformator Data Structure performing the second transformation step from the domain profile format to the Hope format.
4. One MDStore for HOPE cleaned records. This MDStore is fed by the Transformator Data Structure applying the vocabulary matching tables and the rules for the normalization of content.
5. One MDStore for the HOPE edited records. This MDStore is fed by the Metadata

Editor Service and merged with the MDStore of cleaned records when the user of the Metadata Editor commits his changes.

6. One MDStore for the EDM metadata records. This MDStore is fed by the Transformator Data Structure performing the transformation from the HOPE format to the EDM format.

3.5. Index Service HOPE customisation

Index services had to be customized to store and reply to queries over HOPE metadata records and over EDM metadata records. EDM records had to be generated and indexed because the OAI-PMH Publisher Service exports records contained into indices and HOPE needed to export EDM records to Europeana.

Indices had to be optimized to reply as efficiently as possible to the specific queries run by the portal. In other words, indices contain HOPE records relative to descriptive units, digital resources, places, agents, concepts, events and content providers, but enriches such records with extra fields to match the query requirements of the portal. For example, the portal should allow for searching digital resources based on the titles of the descriptive units they are related with. To this aim, the records of digital resource had to be enriched with the titles of such descriptive unit and be indexed accordingly.

A new index service implementation has been designed and is today under development. The service includes advanced functionality for browse by fields, full text index on OCR texts and multilingual search.

3.6. User Interface HOPE customisation

A customization of the "Light User Interface" was implemented to support a simple user interaction with the system during the design and implementation of the HOPE Portal that will be released later on during the project life span.

With such customization – called "Content Checker" – it is possible for CP administrators to check the content transformed in the HOPE format.

A further customization – called "Vocabulary Checker" – allows CP administrators to view records whose content could not be cleaned (for example because of incomplete vocabulary matching tables).

3.7. Metadata Export HOPE customisation

HOPE XML metadata records, which are continuously collected and curated as explained above, will be disseminated through different protocols and possibly different XML formats.

To this aim, several service workflows will be constructed, one for each typology of data export. Whenever the data requires a transformation into another format, the Transformator Service will be involved: this is the case for the Europeana Metadata Schema (EDM) and for the records to be exported to third-party systems, which generally accept records matching

specific import formats. The D-NET services that will be used for metadata exportation are introduced below.

3.7.1. Search Web Service

A Search Web Service offers an SRW/CQL (Search/Retrieval via URL, http://www.loc.gov/standards/sru) interface accepting a CQL query Q and a metadata format, to run Q over the Index units matching that model.

To this aim, queries are routed to the right Index Services; the responses, when more than one Index Service is involved, are then "fused" and pushed into a ResultSet, whose EPR is returned as result.

Note that for performance reasons the Search Web Service memorises a cache of the Index units available, kept up to date by subscribing to the creation and removal of Index units in the system.

3.7.2. OAI-PMH Publisher Service

An OAI-PMH Publisher Service offers OAI-PMH interfaces to third-party applications (i.e., harvesters) willing to access metadata objects in the MDStore units.

To this aim, the service dynamically discovers and exposes through the ListMetadataFormats verb the list of metadata formats currently available in MDStore units and offers a getRecords operation over all the MDStore units hosting such records.

3.7.3. Export Service

A new D-NET Service will be realized, capable of exporting XML records from MDStore units to known social web sites, such as YouTube, Flickr, Google and Scribd. Preferably, the Export Service will not fetch the derivatives from the repository, but will use the appropriate URL for external site to retrieve the derivative. In cases where the target social site is not able to fetch content from a URL (e.g., YouTube), the Export Service must interact with the repository that contains the file in order to download it and store it temporary.

The components of the Export Service are:

- ***ES-search*:** CPs defines which digital objects should be exported to which social site via dissemination profiles and tags. The ES-search component looks up into the Indexes to find the digital resource and the corresponding metadata to export. Metadata is then passed to the ES-dispatcher component.
- ***ES-dispatcher***: this component receives metadata records from the ES-search component and dispatches them to one (or more) of the available ES-publishers (there is one ES-publisher typology for each social site) depending on the dissemination policy.
- ***ES-publisher***: for each target social site, there exist at least one ES-publisher component. Its functionalities are those of:
    - Downloading the file content of the digital resource into a local temporary file if

the target social site does not fetch content from URL.

- Mapping the HOPE metadata of digital resources into the data model of the target social site.
- Publishing the content to the social site by using the apposite APIs.

Downloading and publishing phases can take long execution time and possibly need a large amount of storage space, thus ES-publishers are designed to be easily replicable to different machines.

3.7.4. Metadata Synchronisation Service

Having a continuous feedback loop between the Aggregator and the content providers is one of the value propositions of the HOPE system. Especially, named entity recognition, the addition of harmonized geographical references, data harmonization, and replacing literals with authoritative URLs are important to CPs.

A new D-NET service will realize this feedback loop and make it possible for HOPE and the CPs to become an integral part of the Linked Open Data cloud.

3.8. Metadata Curation HOPE customisation

HOPE curators (often a group of "experts in the field" selected across the content providers) may be willing to perform further semi-automatic cleansing activities.

3.8.1. Authority File Service

The Authority File Service implements functionalities for "curating" a set of authority files, intended as sets of authoritative metadata records. A user interface is available for HOPE curators to see and possibly merge the candidate pairs of duplicate records.

At the time of writing, the HOPE consortium has not yet agreed on which entities are to be considered authoritative in the HOPE Information Space. Theorically, the Authority File Service could be customized to work with the following entities of the HOPE data model: agents, places, events, themes and concepts. Authority files, as well as report files, may be accessed and exploited also by content providers to improve the quality of their data. Authority files will need to make use of PIDs in order to be accessible in an unequivocal fashion across the HOPE system and beyond.

The HOPE Authority File Service will be formalized in [HOPEAFS].

3.8.2. Metadata Editing Service

The Metadata Editing service is configured to work on the HOPE schema. A user interface is available for HOPE curators to edit existing metadata (or add new metadata records) for digital resources, descriptive units, agents, places, concepts, content providers, events.

The HOPE Metadata Editing Service will be formalized in [HOPEMES].

3.8.3. Metadata Tagging Service

The tagging service can be used in the HOPE project to assign:

- *Theme Tags:* groups of HOPE records that refer to a particular context or historical period can be associated to historical themes;
- *Dissemination Tags:* each Content Provider defines dissemination policies for their sets of metadata. A dissemination policy defines the rules that are to be applied for the export of records to social sites, Europeana and the IALHI portal. Dissemination rules can be overridden by tagging HOPE XML records with the opportune dissemination tags, so that Content Providers can define easily fine-grained dissemination rules and exceptions to the rules defined in their Dissemination Profile.

The HOPE Metadata Tagging Service will be formalized in [HOPEMTS].