This is a pre-print of the following article: Paolo Manghi Michele Artini Claudio Atzori Alessia Bardi Andrea Mannocci Sandro La Bruzzo Leonardo Candela Donatella Castelli Pasquale Pagano , (2014),"The D-NET software toolkit", Program: electronic library and information systems, Vol. 48 Iss 4 pp. 322 - 354
http://dx.doi.org/10.1108/PROG-08-2013-0045

Please, cite this document by citing to the official article.

The D-NET Software Toolkit: A Framework for the Realization, Maintenance, and Operation of Aggregative Infrastructures

Paolo Manghi

Michele Artini, Claudio Atzori, Alessia Bardi,
Andrea Mannocci, Sandro La Bruzzo
Leonardo Candela, Donatella Castelli, Pasquale Pagano

Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
Via Moruzzi 1, Pisa
name.surname@isti.cnr.it

Abstract In the last decade, as a consequence of the multidisciplinary and data-driven character of modern science, researchers grew a strong demand for collecting, integrating and combining information objects from multiple r research-oriented data sources. The objective is to improve the way research results are shared, discovered, and re-used across community boundaries. Interoperability issues of data and technologies, scarcity of computational resources, and highly evolving requirements typically represent an obstacle to practitioners constructing the "aggregative infrastructures" capable of performing such an integration. This paper presents the architectural principles and the services of the D-NET Software Toolkit, a framework where developers find the tools for constructing and operating aggregative infrastructures in a cost-effective way as instances of service-oriented data infrastructures. In D-NET developers can select from a variety of data management services, can configure them to handle data according to given data models, and can combine them into autonomic workflows to obtain personalized aggregative infrastructures.

1. Introduction

Research Repositories (RRs) are systems devised to support scientists at depositing, preserving, accessing and re-using research material and outcome. Traditionally, RRs include digital archives and publication repositories, which assist the research life-cycle by offering such functionalities for multimedia objects, e.g. digitized material, and scientific literature, e.g. articles (Candela et al., 2013b). In the last decade, the advent of data-driven science has introduced datasets, intended as scientific raw and secondary data, in the scholarly communication chain. A new generation of RRs, namely dataset repositories, and relative technologies have been developed for this purpose, e.g. CKAN (http://ckan.org/), DRYAD (White et al., 2008), PANGAEA (Diepenbroek et al. 2002). Such RRs contain datasets accurately described with metadata in order to enable citation, discovery and re-use (Callaghan et al., 2012).

The multidisciplinary character of science and the strong requirements for immediate access to new research results called for systems that facilitate researchers at discovering and accessing content available from RRs, be these in their community or across distinct communities. Such novel systems, hereafter *aggregative infrastructures*, are capable of collecting *information objects* (i.e. metadata and the files they describe) from heterogeneous RRs in order to form uniform, cleaned, and enriched cross-RR information spaces. These information spaces facilitate access to the objects at their original sites, by enabling the realization of advanced services over the aggregated content; e.g. search, browse, statistics, recommendations. In this work we shall focus on three categories of aggregative infrastructures, for publication (or dataset) *repositories*, *cultural heritage*, and *scholarly communication*. Publication (or dataset) repository infrastructures collect and process bibliographic (or scientific) metadata from a pool of repositories to support improved and uniform cross-search and access to research literature (or datasets). In the context of cultural heritage infrastructures metadata and audio/video material from several digital archives are collected and processed to enable data normalization, cross-discovery, preservation, or export towards third-party services. Finally, scholarly communication infrastructures are built on the experience of repository infrastructures and follow the demands of modern science. Scholarly communication infrastructures give life to environments where publications and datasets can be collected from the relative data sources to be interlinked and enriched with further contextual information (e.g. funding, institutions, research initiatives). The aim is not only to provide to the researchers a richer, searchable, navigable and reusable information space about research, but also to offer tools for measuring research impact with respect to funding, research initiatives, etc.

Although distinct in the nature of the information objects they handle, aggregative infrastructures have common functional and architectural patterns. Examples of the functional patterns are those involving data curation: managing multiple metadata formats, maintaining mappings between them, or de-duplicating, annotating, validating, and enriching metadata. Examples of architectural patterns are scalability (e.g. support access to large sets of objects), robustness (e.g. preserve objects from being lost), and administration of a set of data sources, hence dealing with tasks such as data workflow definition and scheduling. Despite the similarities, however, software solutions to aggregative infrastructures are often tailored to one application domain and can hardly be re-used into scenarios where distinct requirements apply. As a consequence, the realization of an aggregative infrastructure is generally a from-scratch operation, hardly sustainable in terms of design, development and maintenance of software (Manghi et al., 2010c).

This paper presents the D-NET Software Toolkit. D-NET proposes a service-oriented framework specifically designed to support developers at constructing custom aggregative infrastructures in a cost-effective way. D-NET offers *data management services* capable of providing access to different kinds of external data sources, storing and processing information objects of any data models, converting them into common formats, and exposing information objects to third-party applications through a number of standard access APIs. D-NET services are obtained by encapsulating advanced and state-of-the-art open-source products for data storage, indexing, and processing – such as PostgreSQL (The PostgreSQL Global Development Group, 2010), MongoDB (MongoDB, 2012), Apache Solr (Kuć, 2013), and Apache HBase (The Apache Software Foundation, 2013) – in order to serve broad application needs. Most importantly, D-NET offers *infrastructure enabling services* that facilitate the construction of domain-specific aggregative infrastructures by selecting and configuring the needed services and easily combining them to form autonomic data processing workflows. The combination of out-of-the box data management services and tools for assembling them into workflows makes the toolkit an appealing starting platform for developers having to face the realization of aggregative infrastructures. D-NET was briefly introduced in (Iatropoulou et al., 2010; Manghi et al., 2010b) to describe its initial engineering and its use in the realization of repository infrastructures. After several years of improvement and adaptations to satisfy different real-case applications, this paper aims at providing the first in-depth description of D-NET and its today largely enriched data management service kit.

Outline of the paper The paper is organized as follows: Section 2 presents the high-level architecture of aggregative infrastructures, resulted from rationalizing the issues that practitioners in the digital library field have to face when designing and developing such systems. Section 3 presents the general-concepts of the D-NET software toolkit. Section 4 describes D-NET's data management services and how they can be used to realize customized, robust and scalable aggregative infrastructures. Section 5 reports on real-case aggregative infrastructures powered by D-NET. Section 6 comments on the cost-effectiveness of the D-NET approach, illustrating pros and cons. Finally, Section 7 concludes the paper.

2. Aggregative infrastructures

In the context of Research Repositories, an information object is a digital object characterized by: (*i*) a unique identifier (in its scope of creation), (*ii*) a metadata object, and (*iii*) the described resource(s), e.g. file, web resource, real-world resource. Aggregative infrastructures are information systems whose software components address the functionalities for collecting metadata objects from a set of RRs (e.g. repositories, archives, libraries, databases) and processing them with the purpose of forming homogeneous information spaces. For the purpose of metadata enrichment and object delivery, aggregative infrastructures may also collect and process the digital objects described by such metadata. Aggregative infrastructures offer advanced services to scientists and communities (and their applications, e.g. e-Science infrastructures) who may not have the resources (e.g. funds, computational power, the storage space, or the technical competence) to individually build such aggregations. In the following we shall describe the general features of aggregative infrastructures, present categories of such systems, and report on solutions to support their realization.

2.1 Features

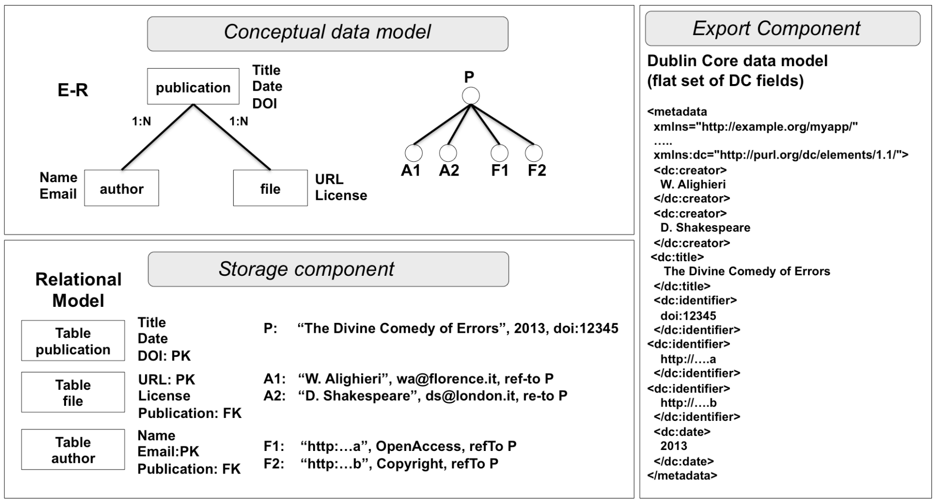
Content Typically, aggregative infrastructure components deal with information spaces of metadata objects and, possibly, relative digital objects:

Metadata objects Metadata objects are descriptive information relative to a digital object or to a "real-world entity" (e.g. a book in shelf, a painting in a museum). Typically, if needed, the link between metadata objects and their digital objects is encoded by a metadata property value (e.g. a URL). *Metadata data models* describe the structure of metadata objects and how these objects of different types are linked with each other. Such models can be can be conceptually represented as graphs of entities, whose edges define entity relationships. The relative objects can be digitally encoded according to different physical models so as to serve different usages, such as efficient access, computation, and export.

Figure 1 illustrates a conceptual model and two physical storage models, respectively intended for access to and exports of metadata objects relative to publications, authors and publication files. Such objects are stored as records of relational tables into a storage software component, but also as Dublin Core XML metadata records of an export software component. The relational representation mirrors the entities of the conceptual model, while the XML representation wraps up publication objects with the related author names and the URLs of the publication files. Export formats of metadata are generally expressed using XML. As a consequence, metadata data models typically come with a corresponding XML schema (also called XML format) to be used for export/import. Often mappings from such formats onto standard XML schemas are provided so as to leverage interoperability, e.g. Dublin Core

(Dekkers and Weibel, 2003), MARC (Furrie et al., 2003), EAD (Pitti, 1999), MPEG21 (Walle and Koenen, 2006). Lately, also the RDF format for LinkedData (Linked Data community; Bizer et al., 2009) is being regarded as a valuable and standard format/protocol to export metadata objects.

An *information space* is a collection of metadata objects conforming to the same conceptual data model. Such a collection can be materialized according to different physical models (e.g. relational database, graph store, XML native DBs), depending on the functionality to be supported (e.g. statistics, navigation, mining, export). Typically, an information space conforming to a conceptual model can be *mirrored* according to different physical models, i.e. the same object collection is kept copied or synchronized according to different backends, in order to enable different kinds of processing over the data. Orthogonally, information space can be *staged* into other information spaces, conforming to distinct conceptual data models.



```
<metadata
  xmlns="http://example.org/myapp/"
  …..
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <dc:creator>
    W. Alighieri
  </dc:creator>
  <dc:creator>
    D. Shakespeare
  </dc:creator>
 <dc:title>
     The Divine Comedy of Errors
  </dc:title>
  <dc:identifier>
    doi:12345
  </dc:identifier>
<dc:identifier>
    http://….a
  </dc:identifier>
<dc:identifier>
    http://….b
  </dc:identifier>
  <dc:date>
    2013
  </dc:date>
</metadata>
```

Figure 1 - Metadata data models and physical representations

Digital object Digital objects are files (e.g. video files, text files, image files) typically accompanied by technical metadata descriptions, such as the file format (e.g. mime), and application-oriented parameters (e.g. color features for images, size of file, version of creating application). In the following, we shall consider digital objects and metadata as logically de-coupled, hence described by different information object data models, although in principle they may be stored together as one digital object. Depending on their intended usage, digital objects can be stored in various ways: from plain file systems, to scalable storage systems, e.g. MongoDB, complex policy management systems, e.g. iRODS (Rajasekar et al., 2010), and streaming servers.

Functionality Aggregative infrastructures are constituted by software components whose functionalities and workflows largely vary depending on application scenarios. In general components can be classified in the following functional areas, depicted in Figure 2:

Storage Components in this area manage storage and indexing of digital objects and/or metadata objects. For example a full-text index component which keeps an information space of Dublin Core metadata objects; or a storage component preserving large information spaces of AVI video digital objects together with MPEG21 metadata objects.

Mediation Components in this area are capable of administering a set of "external" data sources in order to enable information object exchange. The exchange may be based on standard digital library APIs and formats, such as the Protocol for Metadata Harvesting (OAI-PMH), Object Exchange Protocol (OAI-ORE) (Carl Lagoze and Herbert Van de Sompel, 2007), and the more traditional ODBC, FTP, WSDL/SOAP, SRW, and REST. Components in this area may range from "registries" of data sources, which keep a list of data source descriptions (e.g. access interfaces, typology, etc.), and "mediators", which implement the actual data exchange. Examples may be a registry for administrating a set of remote JDBC-compliant library catalogues; and a search mediation component, capable of sending SQL queries to the remote catalogues and returning their fused results.

Provision Components in this area allow third-party applications to access information spaces by means of standard APIs and formats. Examples may be OAIPMH publisher or OpenSearch components abstracting over the index components in the storage area.

Manipulation Components in this area offer functionality for information space enrichment, mirroring, staging, statistics, and validation. Enrichment may take place thanks to components for inference and digital object transcoding: transcoding DOCX documents into text files, mining text files for subject classification, histograms from image digital objects, etc. Mirroring and staging may be achieved via components for the transformation of metadata objects (e.g., mapping EAD metadata objects onto MARC21 metadata objects). Validation evaluates the quality of information spaces, e.g. quality of metadata records from a repository with respect to some given data quality rules.

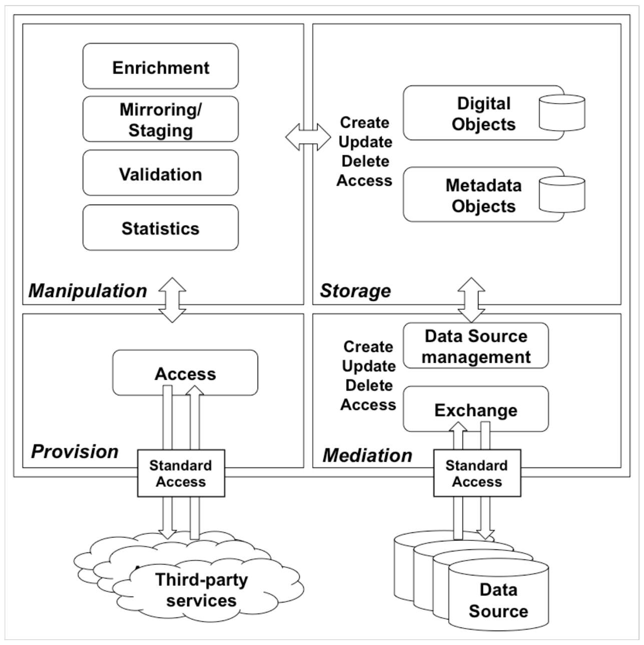


Figure 2 - Aggregative infrastructures functional architecture

Actors The life-cycle of aggregative infrastructures consists of three main activities, namely *realization*, *maintenance* and *operation*, whose responsibility is of four main actors: *designers*, *developers*, *system administrators*, *data managers*/*curators*.

Realization: design of a system matching the initial requirements of data aggregation and involves two actors, namely designers and developers.

Maintenance: standard software support, e.g. upgrade of software versions, and software refinement, that is the adaptation of the software to new evolving functional requirements; both designers and developers are involved.

Operation: the activity of operating a running infrastructure, which concerns (*i*) hardware/application administration and (*ii*) information space management. The first task is responsibility of system administrators who are in charge of monitoring and fixing hardware features and software deployment to match Quality of Service requirements, e.g. robustness, availability, scalability, etc. The second task is assigned to data managers/curators who must make sure information objects are properly collected from data sources, transformed, enriched, validated to form mirrors or stages of information spaces according to the given requirements.

2.2 Real-case scenarios

In this section, we shall report on three typologies of aggregative infrastructures to showcase their main features and differences: *repository infrastructures*, *scholarly communication infrastructures*, and *cultural heritage infrastructures*. The aim is to show the variety of functional components involved in their realization, the underlying implementation complexity, and to highlight the interoperability issues that arise when combining components into workflows for information space delivery. Examples of components are exemplified in Table 1.

Table 1 - Aggregative infrastructure components: samples.

Functional areas	Functional sub-area	Components	Repository Infrastructures	Cultural Heritage infrastructures	Scholarly Communication infrastructures
Mediation	**Importing**	*XML OAI-PMH harvesting*	X	X	X
		HTTP harvesting		X	X
		XML FTP harvesting	X	X	X
	Exporting	*uploading files to remote data sources*		X	
	Data source management	*registration, profiling, scheduling*	X	X	X
Manipulation	**Mirroring and Staging**	*XML XSLT mappings: one-to-one mapping*	X	X	X
		XML scripts: one-to-many/many-to-one mappings		X	X
		file transcoding	X	X	X
	Enrichment	*full-text from PDFs, DOCX, etc.*	X	X	X
		key-words (concepts) from full-text	X	X	X
		histograms from images		X	
		relationships between metadata objects		X	X
		de-duplication/merge of metadata objects	X	X	X
	Validation	*validation of metadata from data sources*	X	X	X
	Statistics	*statistics: graphs, pies, tables*	X	X	X
Storage	**Digital objects**	*file store*	X	X	X
		long-term preservation		X	X
		video and images		X	
	Metadata objects	*XML store*	X	X	X
		full-text index	X	X	X
		graph-like data models		X	X
Provision	**Standard APIs**	*OAI protocols*	X	X	X
		Optimized search over multiple full-text indices	X	X	X
		LinkedData	X	X	X
		Video streaming		X	

Repository Infrastructures Institutional Repositories (Lynch, 2003) are RRs adopted by research institutions to archive, preserve and provide access for research publications (e.g. article, books, technical reports) and relative bibliographic metadata records. Known repository technology platforms are Fedora (Lagoze et al., 2005), DSpace (Tansley et al., 2003), ePrints (Millington and Nixon, 2007), and Greenstone (Witten et al., 2001). In the Digital Library realm, publication repository infrastructures are a common category of aggregative infrastructures, whose aim is to collect metadata records from a set of OAI-PMH compliant data sources, aggregate such records to form uniform and searchable information spaces, and offer web portals to query over such information spaces (Jackson et al., 2008). Known examples of such systems are OCLC-OAIster (http://www.oaister.org), BASE: Bielefeld

Academic Search Engine (http://www.base-search.net), NARCIS, (Digital Academic Repositories, http://www.narcis.nl), DART-Europe (http: //www.dart-europe.eu/), CORE (http://core-project.kmi.open.ac.uk) and many others.

Figure 3 shows an example of repository infrastructure's workflow for metadata collection and transformation: metadata records (e.g. Dublin Core records) are collected from a data source to be transformed onto metadata objects of a target uniform information space. Other workflows may be present in the aggregative infrastructure, e.g. for inferring content and enriching the information space. The variables in this scenario, which have an impact on the kind of components to be developed, are typically: the number and variability of the repositories to be harvested, the required frequency of data collection from the repositories, the common data model of the information space (e.g. an XML metadata format), the metadata mappings, the information extraction operation to be performed (e.g. computation intensive), the level of robustness (e.g. replicas of aggregated metadata), availability (e.g. number of indices, number of networks involved), and scalability (e.g. number of accesses). It is evident how, depending on the peculiarities of the application domain, repository infrastructures may substantially differ from each other and rely on components integrating different software for data management. As shown in Table 1, these infrastructures may need components for (i) the collection, storage and indexing of information objects, (ii) the management of a set of data sources and relative workflows, (iii) the transformation of XML records with one-to-one mappings (e.g. REPOX (Reis et al., 2009) and MINT (Kollia et al., 2012)), and (iv) the export of the objects via several standard protocols for bulk or selective access (e.g. OAI-PMH, OAI-ORE, SRW, LinkedData (Linked Data community; Bizer et al., 2009)).

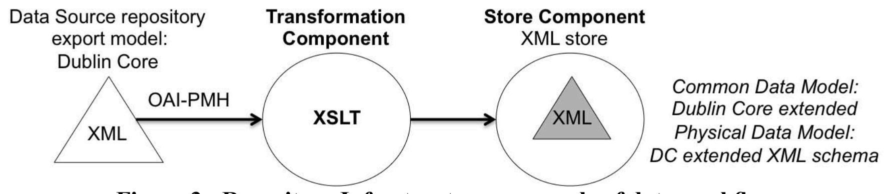


Figure 3 - Repository Infrastructures: example of data workflow

Dataset Repositories are RRs adopted by research institutions (e.g. DRYAD (White et al., 2008)) to archive, preserve and provide access to research data information objects, namely *datasets*, such as time-series, images, genes and proteins, worksheets. The nature of datasets and relative metadata is strictly dependent on the scientific discipline. Dataset repository infrastructures are also frequent and respond to the demand of making accessible heterogeneous and independent scientific data sources. Known examples are the CLARIN infrastructure for language resources (CLA), the ESPAS infrastructure for geospatial information (Hapgood et al., 2012), or DataCite for dataset repositories associated to the DataCite initiative (www.datacite.org).

Cultural Heritage Infrastructures Cultural Heritage (CH) is certainly one of the most active community in the realization of aggregative infrastructures (Loebbecke and Thaller, 2011; Bardi et al., 2012b; Hunter and Gerber, 2010). *Digital archives* are a category of RRs dedicated to the storage of (potentially large) digital objects and metadata objects of cultural heritage material, including texts, videos, audios, and images. The increased availability of cultural heritage digital content raised a natural need to realize infrastructures for the integration, preservation and delivery of such content to wider research, academic, and public communities. Examples are the infrastructures supported by the European Infrastructure for Cultural Heritage (Aloia et al., 2011) (http://www.europeana.eu), the European Film Gateway project (Artini et al., 2013) and the Heritage of People's Europe (Bardi et al., 2012b).

The realization of aggregative infrastructures for CH can be particularly complex when compared to other disciplines, due to the high degree of heterogeneity and multidisciplinary flavor brought in by cultural heritage communities (Papatheodorou, 2012). Typically, digital archives export objects conforming to graph-like metadata data models (e.g. MARCXML for libraries, EAD for archives, EN 15907 for audio/video, LIDO for visual objects), generally packaged as XML records. Similarly, the aggregative infrastructure information spaces contain metadata objects conforming to graph-like data models. Common functional requirements are: making CH material available via full-text search, enabling the navigation of relationships between objects, and providing statistics based on such relationships. Examples of workflow in a CH aggregative infrastructure are shown in Figure 4. The first peculiarity of such aggregative infrastructures is that the transformation of data source XML records into metadata objects in the information space is typically not a one-to-one mapping. The transformation requires components to "unpackage" the incoming XML metadata records in order to create information objects compliant to the common data model of the target information space. Secondly, an XML physical representation of the information space may not be the optimal choice, for example to enable navigation and statistics, and more complex storage and provision components may be required (e.g. relational databases, triples stores such as Virtuoso, Sesame, graph stores such as Neo4J). Finally, aggregative infrastructures for CH tend to include components to handle (possibly large) digital objects to enable, for example, their long-term preservation (e.g. OPM provenance model (Moreau et al., 2011), OAIS reference model (Consultative Committee for Space Data Systems, 2002)), their transcoding into different formats (e.g. thumbnails extraction), or their dispatch to third-party services, e.g. YouTube, Flickr, remote FTP sites, data sources. Table 1 lists a collection of typical CH aggregative infrastructure components.

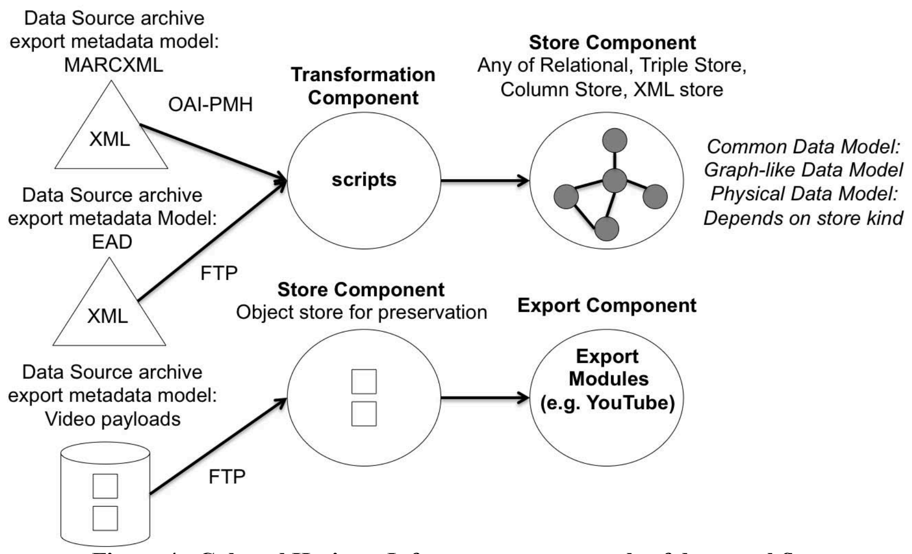

Data Source archive export metadata model: MARCXML
XML
OAI-PMH
Data Source archive export metadata Model: EAD
XML
FTP
Data Source archive export metadata model: Video payloads
FTP
Transformation Component
scripts
Store Component
Any of Relational, Triple Store, Column Store, XML store
Common Data Model: Graph-like Data Model
Physical Data Model: Depends on store kind
Store Component
Object store for preservation
Export Component
Export Modules (e.g. YouTube)

Figure 4 - Cultural Heritage Infrastructures: example of data workflow

Scholarly Communication Infrastructures Although publication and dataset repositories were conceived to serve complementary and non-interoperable tasks of the research process, scientific communication requirements in data-driven science are today forcing them to interoperate (Castelli et al., 2013). Interlinking datasets with literature would greatly improve data availability, discoverability, interpretability and re-usability (Boulton et al., 2012).

CRIS systems (Current Research Information Systems) are a category of RRs devised to store metadata about the "research life-cycle", such as funding schemes, projects and participant organizations and people, research tools and devices, patents. Their function is not only to keep track of grants and stakeholders, but also to associate publications and datasets with contextual information on research activities and actors. CRISs play an important role for institutions willing to measure and assess the quality of their researchers and the impact of research funding or initiatives in terms of productivity, e.g. publications, datasets, patents.

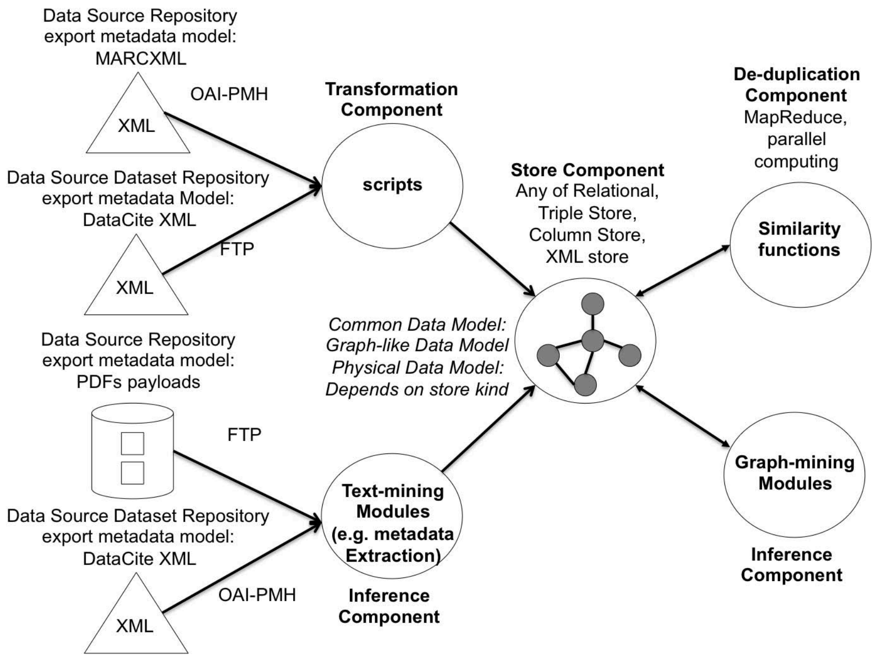

Data Source Repository export metadata model: MARCXML
XML
OAI-PMH
Data Source Dataset Repository export metadata Model: DataCite XML
XML
FTP
Data Source Repository export metadata model: PDFs payloads
FTP
Data Source Dataset Repository export metadata model: DataCite XML
XML
OAI-PMH
Transformation Component
scripts
Text-mining Modules (e.g. metadata Extraction)
Inference Component
Store Component
Any of Relational, Triple Store, Column Store, XML store
Common Data Model: Graph-like Data Model
Physical Data Model: Depends on store kind
De-duplication Component
MapReduce, parallel computing
Similarity functions
Graph-mining Modules
Inference Component

Figure 5 - Scholarly Communication Infrastructures: example of data workflow

Scholarly Communication Infrastructures aggregate content from dataset repositories, publication repositories, and CRISs systems to play the role of a meta-CRIS system, operating cross-institution, cross-region and cross-discipline. Examples are the European OpenAIRE infrastructure (Manghi et al., 2012a) and the Swedish ScienceNet infrastructure (Johansson and Ottosson, 2012). These aggregative infrastructures generally include components to handle transformation, storage and indexing of graph-like metadata models, and introduce inference and de-duplication/merging components in order to: (i) identify new relationships between natively disconnected objects, and (ii) disambiguate metadata objects collected from independent data sources but representing the same real-world entity (e.g. bibliographic records collected from different repositories but describing the same publication). The former components are necessary in order to deliver added value and improve discovery of material by end-users, funding agencies, etc. The latter components are crucial in order for the aggregative infrastructures to expose coherent and precise statistics, fundamental to generate credible measurements and assessments. Besides, such systems tend to grow arbitrarily large information spaces (e.g. hundreds of millions of interlinked objects), which become hardly sustainable in terms of size or computation using open source RDBMs, such as PostgreSQL or MySQL. Using such tools, replication, scalability, availability, volatility, processing of very large data collections become critical challenges for system administrators and developers. For such reasons, such aggregative infrastructures consider the adoption of scalable open source “column stores” such as HBASE (Khetrapal and Ganesh, 2006) and Cassandra (Lakshman and Malik, 2010). Moreover, such stores are integrated with the Hadoop Map Reduce framework, facilitating parallel computing over very large collections, for example to run efficient de-duplication algorithms. Figure 5 illustrates a typical scholarly communication infrastructure’s workflow.

2.3 Related Work

As shown in the previous section, given initial data management requirements, designers and developers realize their aggregative infrastructures by customizing existing software products and pipelining them into workflows. The interoperability issues that may arise are solved by coding software “wrappers” and “connectors”. Typically, resulting solutions are developed “from scratch” to optimally serve a specific application scenario and in general do not target general-purposeness and reuse in different contexts. Such an approach turns out to be not sustainable in the majority of cases, both in terms of maintenance and evolution of the software (Manghi et al., 2010c).

As a reaction, research in the e-Infrastructure field started investigations on software systems designed to support the construction of aggregative infrastructures (Candela et al., 2013a; Manghi et al., 2010a). Typically, such systems implement modules supporting general-purpose functional patterns for data collection, processing, storage and provision in order to allow developers to build aggregative infrastructures by re-using, customizing, and pipelining functionalities into workflows to meet the specific community needs. Each solution has pros and cons and may be more apt to one application scenario or another. For example, the gCube software system (Simeoni et al., 2009) provides a service-oriented solution that is particularly apt for application scenarios requiring parallelization of algorithms for big data analytics, customization of big data staging workflows, dynamic service deployment, and serving computational resources on demand, i.e. hybrid cloud services. The SYNAT system proposed in (Mazurek et al., 2013) (Rosiek et al., 2013) and the CORE system described in (Knoth and Zdrahal, 2012) are instead designed to facilitate the construction of publication repository aggregative infrastructures. Both products (*i*) enable the customization of metadata aggregation and mining processes, and (*ii*) support the realization of advanced access services. A further interesting example is that of ARIADNE (Duval et al, 2011), an infrastructure for the aggregation of learning objects. The underlying software has been designed to enable the construction of learning object infrastructures, but also to support the integration of several of such infrastructures. This is achieved by (*i*) supporting open standards for metadata description, harvesting, query, and ingestion, and (*ii*) enabling the synchronization of registries of learning object repositories from different infrastructures.

While in the context of aggregative infrastructures the gCube system approach would be an overkill, SYNAT, CORE, and ARIADNE services would certainly offer fully-fledged solutions for the realization of repository and learning object infrastructures. On the other hand, being focused on classes of problems, the last three solutions would miss many of the functionalities required to enable the construction of more complex aggregative infrastructures (e.g. support for graph-like metadata models, de-duplication in graph-like information spaces, information space tagging components, etc.).

The D-NET Software Toolkit started from an approach that is similar to SYNAT's, CORE's, and ARIADNE's but then evolved to satisfy the requirements of broader kinds of aggregative infrastructures such as those of cultural heritage applications and modern scholarly communication scenarios.

3. D-NET Software Toolkit

The *D-NET Software Toolkit* is open source (Apache license), fully developed in Java, based on the Web Service framework (http://www.w3.org/2002/ws), and available for download at http://www.d-net.research-infrastructures.eu. Its first software release was designed and developed within the DRIVER and DRIVER-II EC projects (2006-2008) (Feijen et al., 2007). The scenario motivating its realization was that of constructing the European repository infrastructure for Open Access repositories (DRI). The infrastructure had to harvest (tens of) millions of Dublin Core metadata records from hundreds of OAI-PMH repository data sources, harmonize the structure and values of such records to form a uniform information space. A D-NET data infrastructure is a run-time distributed environment, inspired by Service-Oriented Architecture paradigms (Lomow and Newcomer, 2005; MacKenzie et al., 2006), where administrators can dynamically construct, refine, and monitor service-oriented applications for information space construction. A running D-NET infrastructure consists of two main service layers: the *application layer*, which consists of running D-NET data management services, needed to compose the applications of the aggregative infrastructure; and the infrastructure core, called *enabling layer* (Candela et al., 2007), whose services manage and combine services in the application layer to shape up the target applications. The D-NET data management kit, to be presented in the Section 4, provides services that offer out-of-the-box and customizable data management functionalities by encapsulating a variety of open source products. The rest of the section will introduce the enabling layer services and the infrastructural features they deliver.

3.1 The Enabling Layer

Services in the D-NET enabling layer offer *resource discovery & re-use* functionalities, as inspired by overlay networks in peer-to-peer systems, and *autonomic computing* functionality, as inspired by orchestration mechanisms in service-oriented computing. The layer comprises four services: Information Service, Manager Service, Result Set Service, and Chron Job Service.

Information Service (IS) D-NET *resources* are intended as infrastructure entities whose purpose is to be discovered and re-used by services, which are themselves system resources. The IS offers functionality for registration and discovery of system resources. Resource registration consists in submitting a *resource profile* to the IS. A profile is an XML file containing information about the resource, such as an identifier, geographical location, resource typology, properties that describe the current status. Resource discovery consists in submitting a query to the IS, looking up for a resource based on its profile properties. Services, whose status may change over time (e.g. amount of disk space available for an index service), must keep their status properties up-to-date to enable effective resource discovery. The set of resource typologies managed by the IS is not static and can be modified at run-time to integrate new kinds of functionality services into the system. In any moment, the pool of resource profiles in the IS represents the up-to-date "map" of resources available to the system.

For example, a Vocabulary Management Service can offer UIs and APIs for the creation, update, and deletion of vocabularies of terms. The Vocabulary Management Service can register vocabularies as resources in the IS, and keep them up-to-date overtime, so that other services (e.g. a metadata Transformator Service) can discover and synchronize with them. Another important example of resources is that of data sources (e.g. publication repositories, dataset repositories). Managers of such data sources, whose maintenance lays out of the infrastructure, are responsible for the registration of data sources. Their profile may include a description of the data source (e.g. name, web site, location) and technical properties (e.g. access protocols, access parameters). Once registered, the data source can be discovered by services that are able to access their content. As we shall see below, other typologies of D-NET resources may be workflows (i.e. repeatable sequences of actions involving several services), workflow configurations, XML metadata formats, XML mappings, etc.

The IS implements subscription and notification mechanisms according to the WS-BaseNotification specification (Graham et al., 2006). Services can subscribe and be notified to events regarding system resources, for example: registration of a resource of a given kind, de-registration of one given resource, update of resource status information in the profile (thus implicitly to the events which caused such changes to the status), etc.

The current implementation of the IS consists of several Java modules on top of an eXist-db (http://exist.sourceforge.net), an open source native XML database. Resource profiles are represented as XML files matching the XML schemas associated to their resource typology. The adoption of an XML native DB softens the need for service developers to know the exact structure of the profiles when searching for resources with given properties (i.e. XQuery), and allows for a straightforward implementation of the WS-BaseNotification topics, whose formalism based on labeled trees and path queries can be directly mapped onto XML and XQuery.

Manager Service (MS) The service addresses service orchestration and monitoring, hence "autonomic behavior". One or more MSs can be configured by developers to autonomously execute *workflows*. D-NET workflows are resources describing sequences of steps, where each step may consists of business logic (i.e. Java code), remote service invocations, workflow forks (i.e. parallel sub-workflows), and workflow conjunctions (confluence of parallel workflows). Typically, service invocations are preceded by a look-up into the IS, which discovers the "best" service of the needed kind and available to execute the call. Workflows can be fired manually or as a consequence of the notification of a resource-related event from the IS or because of time-events, i.e. chron jobs set by a Chron Job Service (Section 3.1). Workflows are commonly used to automatically schedule data collection from data sources, population of information spaces, and synchronization of information space mirrors or information space staging.

Workflows can implement long-term transactions by exploiting subscription and notification of events in the IS. When a time-consuming step is to be fired (e.g. indexing a large set of metadata objects), the invocation is accompanied by a subscription request to the event "conclusion of the step". The MS waits for the relative notification before moving on to the next step. Workflows can also be used as monitoring threads, checking for consistency and availability of resources or consistency and Quality of Service of the aggregative infrastructure. For example, aggregative infrastructure policies may require that a given collection of information objects be replicated K times. A monitoring workflow may, at given time intervals, check that this is really the case and possibly take corrective actions, e.g. creating a missing replica. When corrective actions are not possible, warning emails can be sent to administrators.

The MS implementation is based on the workflow engine of Sarasvati project (http://code.google.com/p/sarasvati). The MS user interface offers a graphical overview of the ongoing workflows and allows administrators to interact with such workflows, for example to manually re-execute them or to redefine their configuration parameters. Administrators can also consult the history of workflow executions, which keep track of successful and failed workflow steps, as well as of values of given input output parameters for such steps (e.g. number of metadata records). Workflows are treated as infrastructure resources, hence can be shared by different instances of the Manager Service, and are preserved in the IS.

ResultSet Service The service manages ResultSets, namely "buckets" for asynchronously transferring list of objects between a provider service and a consumer service. A ResultSet is an ordered list of files, which can be accessed via paging mechanisms, and is identified by an End Point Reference (EPR) – the Web Service EPR standard describes the location of a web resource on the Internet (Gudgin et al., 2006).

D-NET services can be designed to accept or return ResultSet EPRs as input parameters or results to invocations. Such an approach implements a uniform data (file) exchange layer across different D-NET services, thereby enabling transparent exchange of information objects between the heterogeneous technologies encapsulated by the services.

ResultSets are implemented based on the SOAP protocol for data exchange (Gudgin et al., 2007). As such, the transmission of data (envelope serialization and materialization) may cause delays or overload the channel in the case of large files. To cope with these issues, D-NET services may also expose REST interfaces and provide responses in alternative formats, e.g. JSON. For example this may be the case for index services in the need to communicate with end-user portals.

Chron Job Service The service handles a set of scheduled tasks, called "chron jobs". Each job defines a pair *time interval* and *workflow*. When the time interval expires, the workflow is fired. Time intervals can be repeated infinitely or a given number of times. An example of chron job usage is in the context of repository infrastructures, to fire the workflows necessary to harvest metadata objects from repository data sources. Such actions are generally not executed manually, especially in the presence of hundreds of repositories, and need to be repeated infinitely at given time intervals. As we shall see, DataSource Manager Services exploit Chron Job Service to provide aggregative infrastructure administrators with control panels for harvesting schedules.

3.2 Infrastructural properties

D-NET enabling services enable the construction of aggregative infrastructures with the following features:

Economy of scale Services constituting one aggregative infrastructure may be hosted over servers maintained at different institution sites; such an economy of scale approach softens the administration costs for individual institutions sustaining the infrastructure.

Robustness Service replicas, i.e. clones of functionality and content, can be kept at different sites. This strategy, in combination with dynamic discovery of resources, makes the system more robust to network failures and system crashes (availability of service) as well as to concurrent accesses (scalability by workload distribution).

Autonomicity Manager Services can autonomously orchestrate and monitor the status of services in the aggregative infrastructure.

Elasticity Thanks to dynamic discovery, services can join or leave the infrastructure anytime without administrators having to re-configure application workflows. For example, to make the infrastructure more robust to data loss, administrators should deploy and register a new storage service to the Information Service and increase the parameter "number of storage replicas" in the replica-monitoring workflow of the Manager Service.

4. D-NET Data Management Services

New D-NET service typologies can be added to the framework, generally without requiring changes to existing services, and their instances can be combined into workflows with instances of existing services. To become D-NET services, services must implement the service participation and data exchange policies established by the D-NET environment, which include (i) common data-exchange APIs, to enable functionality-pipeline, (ii) subscription and notification interfaces, to enable distributed transactions with long-term and asynchronous operations, and (iii) service profile registration and update to the Information Service. In order to facilitate the "D-NET-ization" of products/software, the service template required to implement D-NET encapsulation is available to developers.

Today, the D-NET framework provides a *data management service kit*, whose services implement typical aggregative infrastructure components. They have been added in the years to meet the peculiar requirements arisen when facing new challenges in different application domains. Such services, in order to address aggregative infrastructure requirements of arbitrary user communities, are designed according to two engineering principles: modularity and customizability.

Modularity Services provide "functionality in isolation", that is functionality "factored out as much as possible", in order to maximize re-use in different workflows. In addition, they support “functionality pipelining”, that is services in the Data Management Kit agree on how information objects are represented and exchanged between them. Services exchange XML serializations of metadata and digital objects, called *object representations* (ORs). ORs are XML records with *header* and *body* sections, the former describing technical and provenance properties of the object and the latter containing the encoding of the object. Data models are treated as D-NET resources, whose Information Service profile contains a name, a unique identifier, and a link to the XML schema. The header section contains: (i) the identifier assigned to the object in the infrastructure, (ii) the original identifier of the object, if the object was collected from a data source registered to the system, (iii) provenance information about the object (e.g. the identifier of the data source resource from which it was collected, a reference to the D-NET workflow which generated it), and (iv) the identifier of the *data model resource* if the object is metadata. The body section contains the XML encoding of the information object, be it a metadata or a digital object. Metadata objects are represented by an XML encoding whose structure is described the data model resource indicated in the header. Digital objects are represented by an XML with three elements: the URL of the object, if it was collected or still is out of the infrastructure boundaries, the D-NET unique identifier of the object, if it is stored in a D-NET Store Service, and the D-NET unique identifier of a metadata object, if the object is associated to a metadata record.

Functionality in isolation and pipelining facilitate the engineering of custom data management workflows. For example, a repository infrastructure may need to harvest metadata objects, store them and then index them, while another infrastructure may instead need to harvest and then index them straightaway. Such different workflows can be supported in the same system only if harvesting, storage and indexing functionality are factored out and implemented as independent services.

Customizability Services managing metadata objects should be customizable at run-time to operate according to a given data model. This feature makes service instances dynamically programmable (e.g. by Manager Services) and promotes their re-use to serve different goals.. For example, we shall see that instances of D-NET services providing storage and indexing of metadata objects are dynamically configurable to handle collections of metadata records of several data models, based on requests from other services. To make this possible, some of the D-NET data storage services obey to the so-called *factory pattern* design, inspired by the WSRF resource framework (Banks, 2006). A factory service instance can be used across workflows in the need of the same logic (e.g. indexing), but with different purposes (e.g. distinct metadata data models, separate collections of metadata objects), in order to exploit at best local resources (e.g. disk space, CPU). To this aim, one factory service instance implements functionality for: (i) managing a set of *units*, i.e.

containers of objects of a given data model (e.g. Index units) and (ii) managing the objects of one or more units (e.g. feeding and querying an Index unit). Units are themselves resources registered with special profiles to the Information Service, in order for services to discover them for usage (e.g. discovery of the Index units supporting a given data model). Bulk deposit/feeding of objects into a given unit is performed by sending a ResultSet EPR with the objects to the factory service, which will asynchronously pull the objects from the ResultSet to store them into the unit.

The following sections describe the core D-NET data management services available in the latest D-NET release, following the architectural functional areas in Figure 6.

4.1 Storage

Services in this area provide storage of information objects by wrapping known open source technologies, such as full-text indexes, relational databases, document stores, etc.. In particular, in the case of metadata objects D-NET services support storage of ROs abstracting over the underlying physical models (e.g. flat records of a full-text index, interrelated relational tables). To this aim services offer a programmatic solution to the problem of storing an input ResultSet of metadata ROs (XML records), conforming to a given data model (XML schema), onto the local physical model. Two solutions are generally supported: a high-level mechanism, based on so-called *data model layouts*, and a low-level mechanism based on software plugins. For example, the D-NET Index service is based on the Apache Solr index (Kuć, 2013), whose physical model consists of a flat list of index fields, each with configuration flags and parameters such as searchable field, browsable fields, stemming options, returned in query results, etc. The service exposes factory-pattern APIs for the creation of Index units that conform to an input metadata data model resource profile and to a given layout for such model. A data model layout for Index Services is a D-NET resource whose profile in the Information Service contains the name of the layout, the data model of reference, i.e. the identifier of the relative profile, and the names of fields that will appear in the index. For each index field the layout specifies (*i*) the relative indexing configuration parameters and (*ii*) the Xpath that obtains the value to be indexed from the metadata records conforming to the target data model. The same data model may be associated to several layouts, in order to allow the definition of several indexing views of the same metadata record structures. Layouts can be edited by administrators at run-time and support Index Services with all the information needed to generate an index unit for a given data model according to given domain requirements. Further D-NET services, such as MDStore Services and OAI-PMH publisher service, adopt flat list of fields as physical models and can therefore exploit the layout mechanism. Other services, for example Database Services and ColumnStore

Services, cannot instead count on such run-time programmatic approach, since the underlying physical models are not bound to a static structure and vary from application to application (e.g. different relational DB structure). In these cases, mappings are implemented by software service plugins and identified by a unique name to be invoked when feeding an input ResultSet to the service.

Index and Browse Service An Index (factory) Service manages a set of Index units capable of indexing metadata objects and replying full-text CQL queries over such objects (Library of Congress). Consuming services can create units according to a given data model layout, feed them with metadata objects, remove objects, and perform queries over one or more units sharing the same layout. Figure 7 shows a consumer running a query Q over Index 1 and 2, which contain objects of the same layout. The Index Service returns the EPR of a ResultSet that contains the metadata objects in the two units that match Q, ordered by ranking. The service is implemented on the Apache Solr index (Kuć, 2013) and, as such, offers also browse by faceted search (drill-in) functionality. The service also exposes Solr REST APIs.

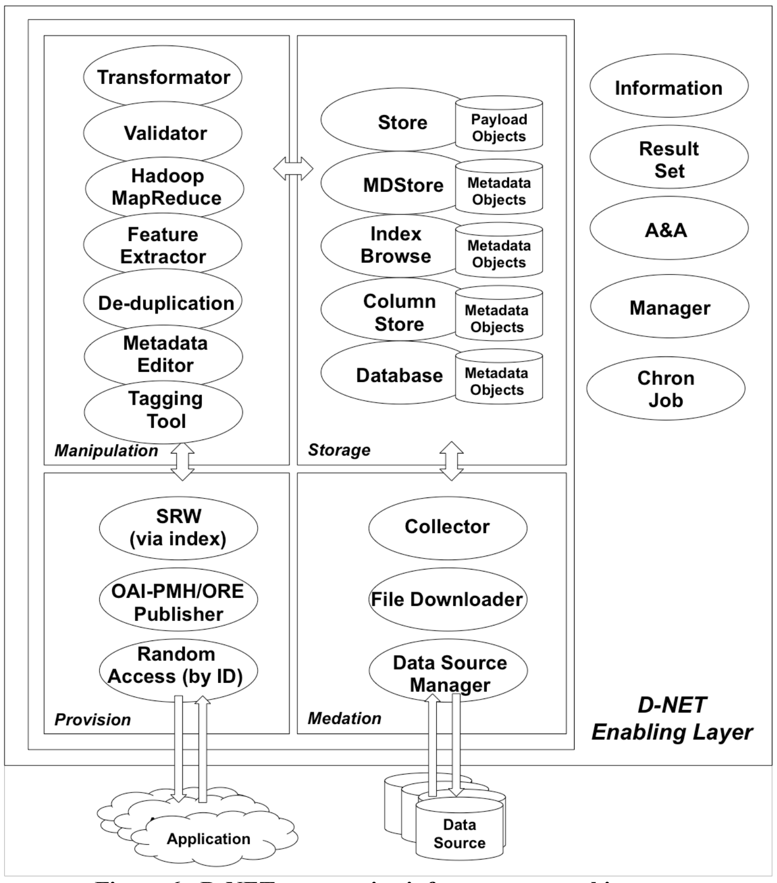


Figure 6 - D-NET aggregative infrastructure architecture

Two different implementations of the service are available, both encapsulated under the same D-NET factory service API. The first implementation adopts Apache Solr in a stand-alone mode while the second in "sharded" mode (i.e. distributed horizontal partitioning). The latter solution is recommended in two scenarios: (*i*) the number of metadata objects grows beyond the usability thresholds of a stand-alone installation – depending on the number of concurrent accesses, RAM and I/O speed – or (*ii*) the refresh rate of the index is very high and the number or size of metadata objects entails indexing time that exceed the application requirements.

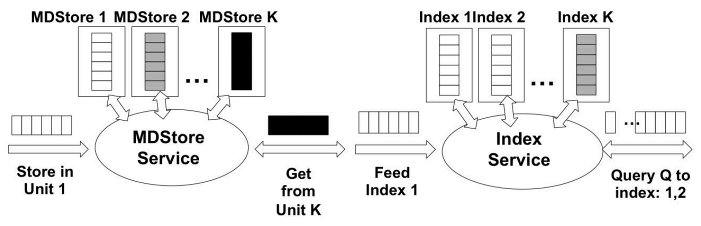


Figure 7 - MDStore Service and Index Service

Store Service The Store (factory) Service manages a set of Store units capable of storing digital objects of a given data model. The Store Service offers to consumers upload functionality that accepts a Store unit and a ResultSet of digital object ORs, and batch downloads files from the list of object URLs. Consumers can retrieve the content of Store units in bulk (e.g. all objects, by date of creation) or fetch specific objects by identifier or by download URL. The Service is implemented as an abstraction over the document-oriented storage MongoDB (Chodorow and Dirolf, 2010), in order to exploit its high-scalability and replica management features, but also to take advantage of out-of-the-box support with the Hadoop Map-Reduce framework.

MDStore Service An MDStore (factory) Service manages a set of MDStore units capable of storing metadata objects of a given metadata data model. Consumers can create and delete units, and add, remove, update, and fetch the metadata objects in a given unit. Figure 7 shows one consumer requesting to store metadata objects from a ResultSet into MDstore 1 and another consumer requesting to access the records stored in MDStore K, which are returned through a ResultSet EPR. As the Store Service, the service is implemented on top of MongoDB.

Database Service The Database (factory) Service offers functionalities for managing a set of Database units. Each unit is an independent database instance, containing relational tables. Consumers can create, delete and manage new Database units, by handling tables and records within. A consumer, be it a D-NET Service or a third-party application, may opt for an interaction through D-NET APIs or a standard socket JDBC connection. The service is developed on top of Postgres DBMS v9.

Column Store Service The service implements interfaces exposing methods to feed metadata objects to an HBase cluster (The Apache Software Foundation, 2013; Khetrapal and Ganesh, 2006) and to fetch objects from the cluster. In order to feed objects to the service, consumers must provide an input ResultSet and a reference to the service plugin required to map the objects into rows of the HBase cluster according to the opted physical representation of the data model. Plugins must be "injected" in the service by developers and are strictly application dependent. Their internal logic is implemented by MapReduce jobs, directly executable over the cluster and for the specific purpose of feeding. The same holds for the action of fetching objects, where extraction plugins are required to return a ResultSet of metadata objects that fits the application needs. As shown in Figure 8, the service hosts several feeding and fetching jobs, which can be invoked by consuming services to accomplish their application tasks. The nodes of the cluster have to be set up by the system administrator at service deployment time. In the current implementation the cluster is ruled by Cloudera (Monash, 2009).

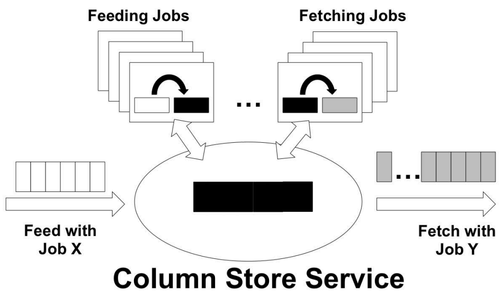


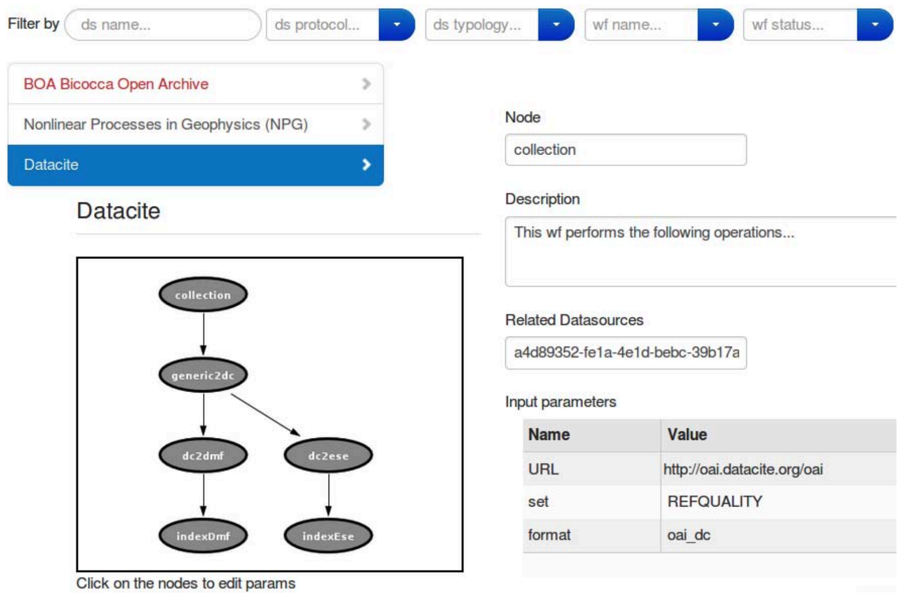
Figure 8 - Column Store Service

4.2 Mediation

Services in the mediation area are capable of fetching data from external data sources and import them into the aggregative infrastructure as information objects conforming to a given data model resource. In order to be discovered and accessed, data sources must be registered to the Information Service with a profile that contains their descriptive properties (e.g. location, name, institution), technical properties (e.g. data source manager, size, availability), and typology (which can vary depending on the application domain, e.g. publication repositories, dataset repositories, aggregators, databases, etc.). The profile can specify on or more *access point interfaces* (APIs), that is different ways to access the content of the data source. For example, a publication repository may provide an OAI-PMH interface as well as an FTP interface to provide bulk-access to metadata and files of publications, respectively. Data source registration occurs manually via the Data Source Manager service user-interface, and can be performed by data source managers or infrastructure administrators, depending on the aggregative infrastructure requirements.

Data Source Manager Data Source Manager Services are instances of the Manager Service tailored to provide user interfaces for the registration and administration of data sources in the aggregative infrastructure.

Administration tasks include the organization and scheduling of data collection and processing workflows.. Figure 9 illustrates the admin user interface used to execute and monitor the execution of a workflow for the data source Datacite. The workflow collects records from the OAI-PMH API of DataCite, transforms them into Dublin Core (DC) and then splits into two branches to be executed in parallel. Each branch transforms the DC records into another format (DMF and ESE) and eventually populates a dedicated Index unit. From the same interface, the data manager can modify the mappings to be applied in the two transformation paths, check the history of past executions of the workflows, and also set an automated scheduling of the workflows, thanks to an interaction with the Chron Job Service.



BOA Bicocca Open Archive

Nonlinear Processes in Geophysics (NPG)

Datacite

Datacite

Node: collection

Description: This wf performs the following operations...

Related Datasources: a4d89352-fe1a-4e1d-bebc-39b17a

Input parameters

Name	Value
URL	http://oai.datacite.org/oai
set	REFQUALITY
format	oai_dc

Figure 9 - Data Source Manager Service: user interface

Collector Service The Collector Service accepts as input a data source and an API for the data source and returns a ResultSet of metadata ORs. The service embeds modules capable of handling the collection of metadata objects via different access protocols. Currently, the service supports OAI-PMH and FTP, but can be extended with further modules. In addition, it can host custom HTTP access modules. Each module, which is identified by a unique name, a URL, a list of parameters and an output metadata data model resource, is implemented by code capable of performing the HTTP calls properly using the parameters and converting the result onto a ResultSet that contains their OR serialization. The conversion logic is hard-coded in the modules, which have to be provided by developers to serve the need of data managers on a case-by-case fashion. Data source resources can, in their profile, specify an HTTP access point by indicating the name of the module to be used.

FTP Service An FTP Service can download digital objects from a given FTP API of a data source registered to the system and store them into a Store unit. To this aim, the service collects the list of files from the given FTP entry point, generates a ResultSet of digital object ROs and sends it to the Store Service as input to the given Store unit.

4.3 Provision

Services in the data provision area interface external applications, e.g. end-user portals, third-party services, with objects in the storage area. Beyond bare random access, D-NET supports the following APIs.

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. The service expects an input ResultSet of metadata ROs, a data model resource and a data model layout. The layout specifies: (*i*) which OAI-PMH sets should be exported from the input ResultSet, based on pairs: set name, Boolean predicate (Artini et al., 2008); (*ii*) which metadata fields of such collections can be used to specify such queries. The service is implemented on MongoDB.

OAI-ORE Publisher Service The OAI-ORE Publisher Service offers programmable OAI-ORE interfaces over relational database records in a Database unit resource. The Service can be configured via user interfaces by data managers (see demo at http://demo.oaizer.research-infrastructures.eu/) to export records of one table as ORE Aggregated Resources, in RDF or XML format (La Bruzzo et al., 2013). Configurations can also specify how records linked to the ones in the table can be included in the transitive closure of the aggregated resources.

4.4 Manipulation

Services in the manipulation area designed to perform information space enrichment, validation, mirroring and staging.

Feature Extractor Service A Feature Extractor Service generates a ResultSet of OR objects by applying a given extraction algorithm to an input ResultSet of OR objects. Examples are: extracting histograms from image digital objects; extracting full-text or keywords from PDF digital objects; converting digital objects from one format to another (e.g. DOC to PDF). Algorithms can be plugged-in as software modules, which are invoked by their unique name.

Transformator Service A Transformator Service addresses the general problem of transforming metadata objects of one metadata data model into objects of one output metadata data model (Haslhofer and Klas, 2010). A call to the service expects an input metadata data model, an input ResultSet of metadata ORs, an output metadata data model, and the mapping to be applied. User interfaces allow data managers to change or modify the mapping by selecting from a list of XSLTs (to be uploaded) or using a mapping rule script editor (Jochen Shirrwagen, University of Bielefeld). Scripts consist of rules of the form: (i) field removal, addition, concatenation and switch, (ii) regular expressions, (iii) invocation of an algorithm through a Feature Extractor Service. User interfaces also support data managers at testing a set of mappings, in the style of products such as Repox (Reis et al., 2009) and MINT (Kollia et al., 2012). XSLTs and rule scripts encode one-to-one mappings, where each incoming metadata object is converted into a corresponding metadata object. In the case of one-to-many and many-to-one conversions, where the expressiveness of XSLT is not enough, data managers select from a list of (Groovy) software modules.

Hadoop MapReduce Service The service offers support for the execution of MapReduce jobs over tables in the Column Store Services, by invoking the name of the job and passing over the required parameters. Such jobs reside locally to the service and are implemented to address application-specific requirements. The association of a job to a given table is hardcoded, hence the consistency of their execution and any possible side effect is responsibility of the data manager.

De-duplication Service The De-duplication Service is an abstraction over the Hadoop MapReduce Service. The service embeds a number of MapReduce jobs for the identification of sets of duplicates in a given Column Store Service table. For each job, data managers can configure a clustering function (Map function) implementing blocking techniques (Reduce function) and a similarity function to identify sets of equivalent records to be merged (Manghi et al., 2012c). Its output is a list of sets of equivalent rows returned via a ResultSet, also made available in dedicated rows of the Column Store table. The interpretation and usage of the output list depends on the application context.

Validator Service A Validator Service is used by data managers (and by data source managers) to verify the quality of a ResultSet of OR metadata objects with respect to a set of validation rules. Sets of rules are defined by data managers as instantiations of rule templates of the form:

if ranking_function(record) *below* threshold *remove* record *else return rank*

Given an input ResultSet and a set of rules, the service returns the ResultSet of the objects that passed the validation (i.e. records ranked over a given minimal threshold of quality), the ResultSet of records that did not pass the validation, and a report of the overall evaluation of the input ResultSet (made available via mail or an URL). Validator Services are often employed in repository infrastructures, for example to ensure a minimal level of data quality of the harvested metadata objects, or to exclude data sources whose overall quality lays beyond a given threshold. The service was realized by Nikon Gasparis, Manos Karvounis, and Antonis Lempesis (University of Athens).

Tagging Tool Service The Record Tagging Service allows data managers to tag a set of records in an Index unit. The service can be configured to include user-defined tag schemes and relative tag. The user interface provides data curators with a virtual environment where they can (i) search and browse to identify the sets of objects they believe should be tagged or untagged, (ii) perform the tagging and untagging actions, and (iii) preview the effects of these actions before making the changes visible to the end-users – see demo at http://demo.tagtick.research-infrastructures.eu (login/password dnet/dnet).

Metadata Editor Service The Metadata Editor Service allows data managers to add, edit and delete metadata records once they have been aggregated and stored into an Index unit. Edit actions include change of property values as well as creating relationships between records in the information space. The service acts as a "record patcher", meaning that changes are persisted independently from the edited records and are applied to the last available version of the records (e.g. last harvested and transformed) before these are streamed to the next step in a workflow. Data managers can be assigned to specific metadata collections (by data source or by MDStore) so as to limit their area of action. In most scenarios, owners of the collections (e.g. data source managers) are enrolled as infrastructure data managers so that they are in charge of updating their records in the infrastructure.

5. Constructing Aggregative infrastructures using D-NET

A D-NET infrastructure is a running environment, enabled by one running instance of the enabling services. Data management services can dynamically register or unregister from the infrastructure environment and be used as components of the intended aggregative infrastructure. D-NET designers and developers construct customized aggregative infrastructures by (i) selecting the services they need from the data management kit, (ii) configuring them to match the data model they require, (iii) deploying and registering the services to the infrastructure enabling layer, and (iv) configuring data workflows using Manager Services. Besides, they might exceptionally realize new services to complement missing functionality. However, some deployments of D-NET infrastructures have been packaged to be distributed as ready-to-install aggregative infrastructures. On request, packages for repository infrastructures, CH infrastructures, and scholarly communication infrastructures are available.

Repository infrastructures D-NET is today the software platform of several repository infrastructures. The CEON national repository aggregator of Poland (ICM), La Referencia national repository aggregator of Argentina (MINCYT), and Recolecta the national repository aggregator of Spain (FECYT) are infrastructures based on installations of the same D-NET package. The package addresses common requirements of this category of infrastructures: (i) aggregation of an arbitrary and dynamic number of repositories, (ii) flexibility of data processing workflows to facilitate their extension with further metadata transformations, (iii) ensuring scalability, robustness, and availability of service on low-cost servers, as typically available at institution sites, and (iv) use of open source products. Specifically, the installation builds an information space with two mirrors: MDStore units, where data source metadata objects are collected and transformed, and Index units, where these objects are made available to portals.. Moreover, Validation Services are included, to rule out low quality metadata or relative repositories, and OAI-PMH Publisher Services, to export the aggregated information space to third-party consumers.

Recently, D-NET has also been used in the context of the ESPAS project (Hapgood et al., 2012), where scientific geo-spatial datasets are being collected and integrated from several scientific data sources.

Cultural Heritage infrastructures In the Cultural Heritage D-NET is currently powering two aggregative infrastructures, respectively for the European Film Gateway EC project (Artini et al., 2013) and for the Heritage of People's Europe EC Project (HOPE). A further aggregative infrastructure focusing on epigraphy material is being realized for the EAGLE project (www.eagle-project.eu).

In the EFG project, the infrastructure provides a single access point to 59 collections from 36 filmography archives and across 21 European countries, for a total of 640,000 digital objects such as movies, documentaries, posters, censorships, etc. The relative metadata objects are collected as metadata records from the archives and mapped onto EFG metadata objects in the information space. EFG objects conform to the EFG Data Model (Savino et al., 2009), a graph-like metadata data model whose information space has two mirrors: MDStore units (XML physical representation), i.e. each object is represented by one XML file with elements encoding relationships to other objects) and Index units. The infrastructure includes Metadata Editor Services and a testing Index unit, intended to manually check the quality of the information space prior sending content to the public Index unit. Due to the graph-like data model of the XMLs harvested from the data sources – e.g.

Cinematographic Works Standards EN15907 (European Committee for Standardization, 2010) – the aggregative infrastructure includes Transformation Services capable of applying many-to-many mappings from original metadata objects to EFG metadata objects. Finally, since the EFG information space has to be delivered to Europeana, the space is staged into another information space conforming to the Europeana Data Model and made available via OAI-PMH Publisher Service. Workflows keep data sources and information spaces synchronized, exception made for the two EFG information space mirrors, which require human validation.

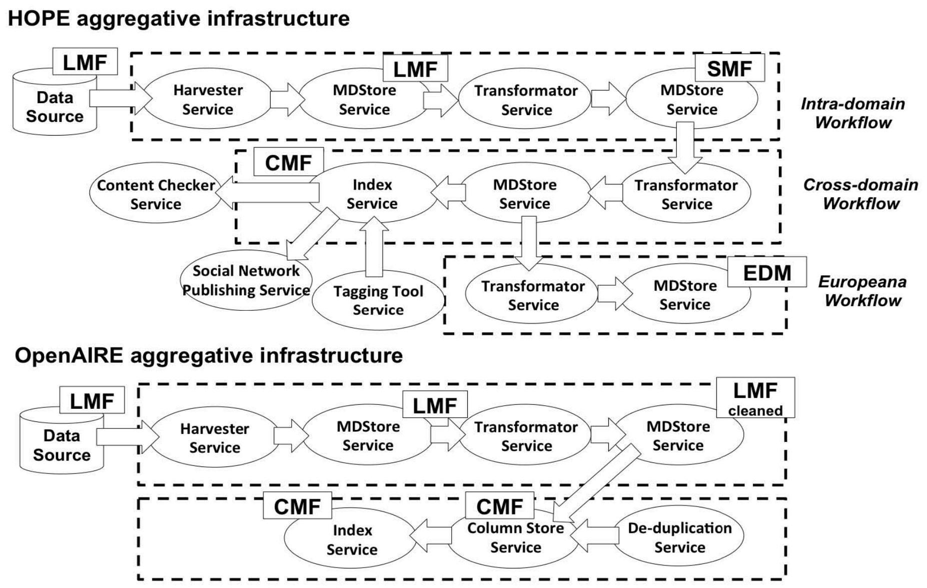


Figure 10 - The HOPE and OpenAIRE aggregative infrastructures

In the HOPE project, the infrastructure today collects more than 2 millions metadata objects, describing around 1,600,000 digital objects, collected from 15 content providers in the field of social and labor history from 18th to 21st century. Similarly to EFG, the HOPE common data model is physically represented with XML files kept in two information space mirrors: MDStore and Index units. The infrastructure stages the HOPE information space into an EDM information space for Europeana. On the other hand, due to the higher heterogeneity degree, the aggregative infrastructure adopts different workflows (Bardi et al., 2012b). HOPE data sources may belong to the domain of libraries, archives, visual, and audio/video. The HOPE common data model needs to capture important concepts of such domains and is therefore designed as a combination of standard metadata formats in such domains: MARC for libraries (Library of Congress, 2005), EAD for archives (Library of Congress, 2002), EN 15907 for audio video, LIDO (ICOM International Committee for

Documentation, 2010) for visual. The infrastructure defines the two-phase transformation workflow illustrated in Figure 10. Given a data source of one of such domains, the original metadata records (compliant to the local metadata format - LMF) are collected and transformed into the corresponding standard XML format (SMF), to be eventually transformed into the HOPE common metadata format (CMF). The infrastructure includes the Tagging Tool Service, to tag metadata objects according to given history ontologies, and Social Network Publishing Services (developed for HOPE), to automatically publish digital objects in the original data sources onto web social tools, such as YouTube and Flickr.

Scholarly Communication infrastructures The goal of the OpenAIRE project (Open Access Infrastructure for Research in Europe) is to realize and maintain the European Scholarly Communication Infrastructure (Manghi et al., 2012a). To this aim, the project deployed a D-NET aggregative infrastructure capable of collecting and interlinking content from publication repositories, dataset repositories, and CRIS systems. The infrastructure must deliver statistics services to measure the impact of research with respect to Open Access mandates (Suber, 2004 – 2012) and funding over research. The infrastructure information space implements the OpenAIRE data model (Manghi et al., 2012b), which defines a graph of entities relative to articles, datasets, persons, projects, organizations, funding agencies across all research disciplines and countries. The infrastructure contains today around 9,000,000 publications and 4,000,000 persons. Its workflows disambiguate (de-duplicate) the information space of publications and datasets and run mining algorithms to identify relationships between such objects, e.g. article-project relationships. To make this possible, the information space has three mirrors, as shown in Figure 10: the first mirror is made of MDStore units caching metadata records harvested from OpenAIRE data sources and their "cleaned" counterpart (i.e. after vocabulary normalization, date formats, etc.). The second mirror is a table in the ColumnStore Service, configured to store the graph of metadata objects relative to the OpenAIRE data model. The third mirror is a full-text index, where the same objects are stored in full-text index physical model, apt to be queried from the OpenAIRE portal. Mirroring from the MDStores to the ColumnStore Service is performed by dedicated feeding modules, capable of handling the transformation from the XML metadata records to the HBASE rows. Modules are relative to publication repositories (Dublin Core), dataset repositories (DataCite data model, http://schema.datacite.org/), and CRIS systems (CERIF data model, EuroCRIS http://www.eurocris.org/). Similarly, mirroring from ColumnStore Service to Index units is performed via data fetching modules that collect and combine HBASE rows to yield the metadata XML records required by the portal. The De-duplication Service is deployed and configured to disambiguate the metadata objects (publication, person and organization metadata objects) in the Column Store Service before these are delivered to the Index Service. A dedicated workflow is run whenever new metadata objects are fed to the Column Store Service, e.g. when new XML records have been collected from the data sources.

6. Cost-effectiveness

D-NET is conceived to reduce the costs of designers and developers at realizing, refining, and maintaining aggregative infrastructures. Such benefits derive from (i) its service-oriented architecture, which supports loosely coupled components and light-weight encapsulation; (ii) the enabling layer features of resource discovery and orchestration; (iii) the existence of a pre-defined kit of data management services, designed according to the principles of modularity and customizability, and encapsulating the most common and state-of-the-art data management back-ends; and (iv) the possibility to easily extend the service kit with further services, i.e. functionalities.

On the other hand, D-NET is intended as a development framework, not as a “double-click” installation product. In order to use it, developers should have the same skills required to realize an aggregative infrastructure from scratch: consolidated experience in Java programming, service-oriented computing, and network protocols. In some cases, depending on the degree of customization, developers should also have knowledge of the internals of the Open Source software used to implement the D-NET services to be deployed in the given installation. For example, in order to customize the usage of D-NET Column Store Services, they must know how the data is organized and stored in an HBASE back-ends. Moreover, developers must acquire the internals of the framework, such as synchronous and asynchronous communication, configuration of the metadata formats, configuration of the workflows, in order to devise aggregative infrastructures matching their requirements. A working week of training proved so far to be enough to start working independently on a D-NET deployment: training sessions have been taken by FECYT (Fundación Española para la Ciencia y la Tecnología), MINCYT (Ministerio de Ciencia, Tecnología e Innovación Productiva Argentina), and the institutions International Institute of Social History (NL) and Library of the Friedrich Ebert Foundation (DE) in order to construct and maintain the respective National repository infrastructures and the HOPE project aggregative infrastructure. The alternative, for such organizations, would have been to design and develop the whole aggregative infrastructures from scratch, with definitely higher cost of realization and maintenance (Manghi et al., 2010c).

However, as previously mentioned, D-NET is also available in “double-click” installation packages. Its services can be pre-configured and packaged to include given workflows and handle given metadata formats, so as to be used as a final product, with no need for developers to dig in software configuration issues. A common D-NET package is the one for repository infrastructures, as described in section 2.

In terms of hardware costs, a D-NET installation does not generally need powerful machines. For example, an installation suitable for a repository aggregative infrastructure handling 10,000,000 documents has the following minimal requirements in virtual machines:

Virtual Machine A: Information Service, Index Service, MDStore Service, Web Portal: CPU 2.4 Ghz dual core, RAM 8 Gb, disk 70 Gb.

Virtual Machine B: Harvester Service, Transformation Service, Data Source Manager Service: CPU 2.4 Ghz, dual core, RAM 2 Gb RAM, disk 20 Gb.

Virtual Machine C : MDStore Service (replica), Index Service (Information Space mirrors replica): CPU 2.4 Ghz, dual core or quad core, RAM 8 Gb, disk 70 Gb.

In order to achieve optimal degrees of system robustness and availability, having physical machines, possibly over different networks, proves to be the optimal solution. This is the case for EFG and HOPE infrastructures. Clearly, D-NET deployments which comprise ColumnStore Services or Index Services with distributed installations would require machine clusters, in a number which depends on the size of the information space and expected performance, and possibly residing on the same local network.

7. Conclusions

The D-NET Software Toolkit is a general-purpose service-oriented framework for the construction of customized, robust, scalable, autonomic aggregative infrastructures in a cost-effective way. D-NET is today adopted by several EC projects, national consortia and communities to create aggregative infrastructures under diverse application domains and other organizations are enquiring for or are experimenting its usage. Its customizability and extensibility make it a suitable candidate for creating aggregative infrastructures mediating between different scientific domains and therefore supporting multi-disciplinary research.

In this paper we have described its architectural principles and focused on its data management kit. A further service kit is available, offering general-purpose user interface functionalities (e.g. user profiling, recommendations, alerts, statistics), whose internals and details are still unpublished.

The D-NET software is open source and available for usage, improvement and extension by any community of developers willing to contribute.

Acknowledgments. This work would have not been possible without the precious cooperation of the other members of the D-NET technical team: ICM Research Centre (Warsaw) for Authentication and Authorization Service, University of Bielefeld Library (Bielefeld, Germany) for development activities in the data management area, and the Department of Informatics, National and Kapodistrian University of Athens (Greece) for the Validator Service. Special thanks go to Marko Mikulicic, Antonis Lempesis, Katerina Iatropoulou, Natalia Manola, Yannis Ioannidis, Jochen Shirrwagen, Friedrich Summann, Wojtek Sylwestrzak, and Marek Horst for their design and development efforts.

Research partially supported by the European Commission FP7 projects: DRIVER-II (Grant agreement: 212147, Call: INFRA-2007-1.2.1 Scientific Digital Repositories), OpenAIRE (Grant agreement: 246686, Call: FP7-INFRASTRUCTURES-2009-1), and OpenAIREplus (Grant agreement: 283595, Call: FP7INFRASTRUCTURES-2011-2), EFG1914 (Grant agreement: 292276, Call: CIP-ICT-PSP-2011-5, 2.2. "Digitising content for Europeana"), and HOPE (Grant agreement: 250549, FP7 EU eContentplus 2009).

References

Clarin: Common language resources and technology infrastructure. URL http://www. clarin.eu.

Driver: Digital repository infrastructure vision for european research. URL http: //www.driver-community.eu.

Nicola Aloia, Cesare Concordia, and Carlo Meghini. Europeana v1.0. In Maristella Agosti, Floriana Esposito, Carlo Meghini, and Nicola Orio, editors, Digital Libraries and Archives -7th Italian Research Conference, IRCDL 2011, Pisa, Italy, January 20-21, 2011. Revised Papers, volume 249 of Communications in Computer and Information Science, pages 127–129. Springer, 2011. doi: http://dx.doi.org/10.1007/978-3-642-27302-5 16.

Michele Artini, Federico Biagini, Paolo Manghi, and Marko Mikulicic. OAI-Publishers in Repository Infrastructures. In Post-proceedings of the Forth Italian Research Conference on Digital Library Systems (IRCDL), pages 93–98, Padua, Italy, January 2008. DELOS: an Association for Digital Libraries.

Michele Artini, Alessia Bardi, Federico Biagini, Franca Debole, Sandro La Bruzzo, Paolo Manghi, Marko Mikulicic, Pasquale Savino, and Franco Zoppi. Data Interoperability and Curation: The European Film Gateway Experience. In Maristella Agosti, Floriana Esposito, Stefano Ferilli, and Nicola Ferro, editors, Digital Libraries and Archives -8th Italian Research Conference, IRCDL 2012, Bari, Italy, February 9-10, 2012, Revised Selected Papers, volume 354 of Communications in Computer and Information Science, pages 33–44. Springer, 2013. doi: 10.1007/978-3-642-35834-0.

Tim Banks. Web services resource framework specification v1.2. Technical report, OASIS, 2006.

Alessia Bardi, Paolo Manghi, and Franco Zoppi. Aggregative Data Infrastructures for the Cultural Heritage. In Juan Manuel Dodero, Manuel Palomo-Duarte, and Pythagoras Karampiperis, editors, 6th Research Conference, MTSR 2012, Cádiz, Spain, November 28-30, 2012. Proceedings, volume 343 of Communications in Computer and Information

Science, pages 239–251. Springer, 2012b.

Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data -the story so far. International Journal on Semantic Web & Information Systems, 5(3):1–22, 2009.

Geoffrey Boulton, Philip Campbell, Brian Collins, Peter Elias, Dame Wendy Hall, Graeme Laurie, Onora O'Neill, Michael Rawlins, Dame Janet Thornton, Patrick Vallance, and Mark Walport. Science as an open enterprise. Final report, The Royal Society, June 2012.

Sarah Callaghan, Steve Donegan, Sam Pepler, Mark Thorley, Nathan Cunningham, Peter Kirsch, Linda Ault, Patrick Bell, Rod Bowie, Adam Leadbetter, Roy Lowry, Gwen Moncoiffé, Kate Harrison, Ben Smith-Haddon, Anita Weatherby, and Dan Wright. Making data a first class scientific output: Data citation and publication by nerc's environmental data centres. International Journal of Digital Curation, 7(1):107–113, 2012. doi: doi:10.2218/ijdc.v7i1.218.

Leonardo Candela, Donatella Castelli, Paolo Manghi, and Pasquale Pagano. Recent Developments in the Design, Construction, and Evaluation of Digital Libraries: Case Studies, chapter Infrastructure-Based Research Digital Libraries, pages 1–17. IGI Global, January 2013a. doi: 10.4018/978-1-4666-2991-2.ch001.

Leonardo Candela, Donatella Castelli, Paolo Manghi, and Pasquale Pagano. Enabling Services in Knowledge Infrastructures: The DRIVER Experience. In Post-proceedings of the Third Italian Research Conference on Digital Library Systems (IRCDL), pages 71–77, Padua, Italy, January 2007. DELOS: a Network of Excellence on Digital Libraries.

Leonardo Candela, Donatella Castelli, Paolo Manghi, and Pasquale Pagano. Infrastructure-Based Research Digital Libraries. In Colleen Cool and Kwong Bor Ng, editors, Recent Developments in the Design, Construction, and Evaluation of Digital Libraries: Case Studies, chapter 1, pages 1–17. IGI Global, 2013b.

Donatella Castelli, Paolo Manghi, and Costantino Thanos. A vision towards scientific communication infrastructures. International Journal on Digital Libraries, pages 1–15, 2013. ISSN 1432-5012. doi: 10.1007/s00799-013-0106-7. URL http://dx.doi.org/10.1007/s00799-013-0106-7.

Kristina Chodorow and Michael Dirolf. MongoDB: The Definitive Guide. O'Reilly Media, 2010.

Consultative Committee for Space Data Systems. Reference Model for an Open Archival Information System. Technical Report CCSDS 650.0-B-1, National Aeronautics and Space Administration, January 2002. Blue Book.

Makx Dekkers and Stuart Weibel. State of the Dublin Core Metadata Initiative. D-Lib Magazine, 9(4), April 2003. URL http://www.dlib.org/dlib/april03/weibel/04weibel.html.

Erik Duval, Katrien Verbert, and Joris Klerkx. "Towards an open learning infrastructure for open educational resources: abundance as a platform for innovation." *Rainbow of computer science*. Springer Berlin Heidelberg, 2011. 144-156.

Michael Diepenbroek, Hannes Grobe, Manfred Reinke, Uwe Schindler, Reiner Schlitzer, Rainer Sieger, Gerold Wefer. PANGAEA - an information system for environmental sciences. Computers & Geosciences Journal, Vol. 28. Issue 10 issn:0098-3004, Elsevier, 2002, 1201-1210

European Committee for Standardization. En 15907 film identification -enhancing interoperability of metadata -element sets and structures. European Standard ICS 35.240.30; 97.195, European Committee for Standardization, July 2010.

Martin Feijen, Wolfram Horstmann, Paolo Manghi, Mary Robinson, and Rosemary Russell. DRIVER: Building the Network for Accessing Digital Repositories across Europe. Ariadne, 53, 2007. ISSN 1361-3200.

Fundación Española para la Ciencia y la Tecnología (FECYT). RECOLECTA or Recolector de

Ciencia Abierta: National repository aggregator of Spain. URL http://recolecta.fecyt.es

Betty Furrie et al. Understanding MARC bibliographic: machine-readable cataloging. Cataloging Distribution Service, Library of Congress, in collaboration with the Follett Software Company, 2003.

Steve Graham, David Hull, and Bryan Murray. Web Services Base Notification 1.3 (WS-BaseNotification). Oasis standard, OASIS, October 2006.

Martin Gudgin, Marc Hadley, and Tony Rogers. Web services addressing 1.0 -core. Recommendation, W3C, May 2006.

Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar, and Yves Lafon. Soap version 1.2 part 1: Messaging framework (second edition). Technical report, W3C, April 2007.

Mike Hapgood, Anna Belehaki, and Natalia Manola. Forecasting the future of near-earth space. International Innovation Journal, pages 67–69, September 2012.

B Haslhofer and W. Klas. A survey of techniques for achieving metadata interoperability. ACM Computing Surveys, 42(2), 2010.

HOPE. Heritage of the people's europe. "http://www.peoplesheritage.eu". Jane Hunter and Anna Gerber. Harvesting community annotations on 3d models of museum artefacts to enhance knowledge, discovery and re-use. Journal of Cultural Heritage, 11(1):81 – 90, 2010. ISSN 1296-2074. doi: 10.1016/j.culher.2009.04.004.

Katerina Iatropoulou, Antonis Lebesis, Paolo Manghi, Natali Manola, and Marko Mikulicic. On Constructing Repository Infrastructures -The D-NET Software Toolkit. In Proceedings of the Fifth International Conference on Open Repositories, Madrid, Spain, July 2010.

ICM. Ceon poland repository aggregator. URL http://agregator.ceon.pl/.

ICOM International Committee for Documentation. Lightweight Information Describing Objects, November 2010. URL http://network.icom.museum/ cidoc/working-groups/data-harvesting-and-interchange/lido-technical/ specification/.

Amy S. Jackson, Myung-Ja Han, Kurt Groetsch, Megan Mustafoff, and Timothy W. Cole. Dublin core metadata harvested through oai-pmh. Journal of Library Metadata, 8(1):5–21, 2008. doi: 10.1300/J517v08n01 02.

Ake Johansson and Mats Ola Ottosson. A national current research information system for sweden. In e-Infrastructures for Research and Innovation : Linking Information Systems to Improve Scientific Knowledge Production, pages 67–71. Agentura Action M, 2012.

Ankur Khetrapal and Vinay Ganesh. Hbase and hypertable for large scale distributed storage systems a performance evaluation for open source bigtable implementations. Evaluation, page 8, 2006. http://www.uavindia.com/ankur/downloads/HypertableHBaseEval2.pdf.

Petr Knoth and Zdenek Zdrahal. Core: three access levels to underpin open access. D-Lib Magazine, 18(11/12), 2012. URL http://oro.open.ac.uk/35755/.

Ilianna Kollia, Vassilis Tzouvaras, Nasos Drosopoulos, and Giorgos Stamou. A systemic approach for effective semantic access to cultural content. Semantic Web, 3(1):65–83, 01 2012. doi: 10.3233/SW-2012-0051. URL http://dx.doi.org/10. 3233/SW-2012-0051.

Rafall Kùc. Apache Solr 4 Cookbook. 2013.

Sandro La Bruzzo, Paolo Manghi, Alessia Bardi: OAIzer: Configurable OAI Exports over Relational Databases. In Garoufallou, Emmanouel and Greenberg, Jane, editors, Metadata and Semantics Research, Communications in Computer and Information Science, pages 35–47. Springer Berlin Heidelberg, 2013, doi:0.1007/978-3-319-03437-9_5

Carl Lagoze, Sandy Payette, Edwin Shin, and Chris Wilper. Fedora: An Architecture for Complex Objects and their Relationships. Journal of Digital Libraries, Special Issue on Complex Objects, 6(2):124–138, 2005.

Carl Lagoze and Herbert Van de Sompel. Interoperability for the discovery, use, and re-use of units of scholarly communication. CTWatch Quarterly 3.3 (2007): 32-41.

Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. SIGOPS Oper. Syst. Rev., 44:35–40, April 2010. ISSN 0163-5980. doi: 10.1145/1773912.1773922.

Library of Congress. Cql: Contextual query language (sru version 1.2 specifications). URL http://www.loc.gov/standards/sru/specs/cql.html.

Library of Congress. Encoded archival description, 2002. URL http://www.loc.gov/ ead/.

Library of Congress. MARC Standards Web Page. http://www.loc.gov/marc/, September 2005.

Linked Data community. Linked Data - Connect Distributed Data across the Web. http://linkeddata.org/home.

Claudia Loebbecke and Manfred Thaller. Digitization as an IT Response to the Preservation of Europe’s Cultural Heritage. In Andrea Carugati and Cecilia Rossignoli, editors, Emerging Themes in Information Systems and Organization Studies, pages 359–372. Physica-Verlag HD, 2011. doi: 10.1007/978-3-7908-2739-2.

Greg Lomow and Eric Newcomer. Understanding SOA with Web Services. Independent Technology Guides. Addison Wesley Professional, 2005.

Clifford A. Lynch. Institutional repositories: Essential infrastructure for scholarship in the digital age. ARL, (226):1–7, February 2003.

Matthew C. MacKenzie, Ken Laskey, Francis McCabe, Peter Brown, and Rebekah Metz. Reference Model for Service Oriented Architecture 1.0. Technical report, OASIS, February 2006. Public Review Draft 1.0.

Paolo Manghi, Leonardo Candela, and Pasquale Pagano. Interoperability Patterns in Digital Library Systems Federations. In Proceedings of the Second DL.org Workshop on Making Digital Libraries Interoperable: Challenges and Approaches, in conjunction with ECDL 2010, Glasgow, Scotland (UK), September 2010a. ISTICNR.

Paolo Manghi, Marko Mikulicic, Leonardo Candela, Michele Artini, and Alessia Bardi. General-Purpose Digital Library Content Laboratory Systems. In Proceedings of the 14th European Conference on Digital Libraries, Glasgow, UK, September 2010b.

Paolo Manghi, Marko Mikulicic, Leonardo Candela, Donatella Castelli, and Pasquale Pagano. Realizing and Maintaining Aggregative Digital Library Systems: D-NET Software Toolkit and OAIster System. D-Lib Magazine, 16(3/4), March/April 2010c. ISSN 1082-9873. doi: doi:10.1045/march2010-manghi.

Paolo Manghi, Lukasz Bolikowski, Natalia Manola, Jochen Shirrwagen, and Tim Smith. Openaireplus: the european scholarly communication data infrastructure. D-Lib Magazine, 18(9-10), September-October 2012a. doi: 10.1045/ september2012-manghi.

Paolo Manghi, Nikos Houssos, Marko Mikulicic, and Brigitte Joerg. The data model of the openaire scientific communication e-infrastructure. In JuanManuel Dodero, Manuel Palomo-Duarte, and Pythagoras Karampiperis, editors, Metadata and Semantics Research, Communications in Computer and Information Science, pages 168–180. Springer Berlin Heidelberg, 2012b. ISBN 978-3-64235232-4. doi: 10.1007/978-3-642-35233-1 18. URL http://dx.doi.org/10.1007/ 978-3-642-35233-1_18.

Paolo Manghi, Marko Mikulicic, and Claudio Atzori. De-duplication of aggregation authority files. Int. J. of Metadata, Semantics and Ontologies, 7(2):114 – 130, 2012c. doi: 10.1504/IJMSO.2012.050014.

Cezary Mazurek, Marcin Mielnicki, Aleksandra Nowak, Maciej Stroinski, Marcin Werla, and Jan Weglarz. Architecture for aggregation, processing and provisioning of data from heterogeneous scientific information services. In Robert Bembenik, Lukasz Skonieczny, Henryk Rybinski, Marzena Kryszkiewicz, and Marek Niezgodka, editors, Intelligent Tools for Building a Scientific Information Platform, volume 467 of Studies in Computational

Intelligence, pages 529– 546. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-35646-9. doi: 10.1007/ 978-3-642-35647-6 32.

Ministerio de Ciencia, Tecnología e Innovación Productiva (MINCYT). La Referencia national repository aggregator of Argentina. URL http://lareferencia.redclara.net

Peter Millington and William J. Nixon. EPrints 3 Pre-Launch Briefing. Ariadne, 50, 2007. C Monash. Cloudera presents the mapreduce bull case. dbMs2. com blog, apr, 15, 2009. MongoDB. Mongodb, 2012. URL http://www.mongodb.org.

Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, Beth Plale, Yogesh Simmhan, Eric Stephan, and Jan Van den Bussche. The open provenance model core specification (v1.1). Future Generation Computer Systems, 27(6):743–756, 2011. doi: http://dx.doi.org/10.1016/j.future.2010.07.005.

National Technical University of Athens. Metadata interoperability services. URL "http://mint.image.ece.ntua.gr/redmine/projects/mint/wiki".

Christos Papatheodorou. On cultural heritage metadata. International Journal of Metadata, Semantics and Ontologies, 7(3):157–161, 01 2012. doi: 10.1504/IJMSO. 2012.050184.

Daniel V. Pitti "Encoded archival description: An introduction and overview." 1999.

Arcot Rajasekar, Reagan Moore, Chien-Yi Hou, Christopher A. Lee, Richard Marciano, Antoine de Torcy, Michael Wan, Wayne Schroeder, Sheau-Yen Chen, Lucas Gilbert, Paul Tooby, and Bing Zhu. iRODS Primer: Integrated Rule-Oriented Data System. Morgan & Claypool, 2010.

Diogo Reis, Nuno Freire, Hugo Manguinhas, and Gilberto Pedrosa. REPOX: a framework for metadata interchange. In Proceedings of the 13th European conference on Research and advanced technology for digital libraries, ECDL'09, pages 479–480, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 3-642-04345-3, 978-3-642-04345, URL http://dl.acm.org/citation.cfm?id=1812799.1812878.

Tomasz Rosiek, Wojtek Sylwestrzak, Aleksander Nowinski, and Marek Niezg´odka Infrastructural approach to modern digital library and repository management systems. In Robert Bembenik, Lukasz Skonieczny, Henryk Rybinski, Marzena Kryszkiewicz, and Marek Niezgodka, editors, Intelligent Tools for Building a Scientific Information Platform, volume 467 of Studies in Computational Intelligence, pages 111–128. Springer, 2013. ISBN 978-3-642-35646-9. URL http: //dblp.uni-trier.de/db/series/sci/sci467.html#RosiekSNN13.

Pasquale Savino, Franca Debole, and Georg Eckes. Searching and browsing film archives. the european film gateway approach. In 4th International Congress on Science and Technology on the Safeguard of Cultural Heritage in the Mediterranean Basin, Cairo, Egypt, December 2009.

Fabio Simeoni, Leonardo Candela, David Lievens, Pasquale Pagano, and Manuele Simi. Functional adaptivity for digital library services in e-infrastructures: the gcube approach. In Agosti M., Borbinha J., Kapidakis S., Papatheodorou C., and Tsakonas G., editors, Research and Advanced Technology for Digital Libraries. 13th European Conference, pages 51 – 62, Corfu, Greece, 2009. Springer Verlag. ISBN 0302-9743. doi: dx.doi.org/10.1007/978-3-642-04346-8\ 7. In: ECDL 2009 -Research and Advanced Technology for Digital Libraries. 13th European Conference (Corfu, Greece, 27 September -October 2 2009). Proceedings, pp. 51 62. Agosti M., Borbinha J., Kapidakis S., Papatheodorou C., Tsakonas G (eds.). (Lecture Notes in Computer Science, vol. 5714). Springer Verlag, 2009.

Peter Suber. Open access overview, 2004 – 2012. URL http://www.earlham.edu/ ~peters/fos/overview.htm#journals.

Robert Tansley, Mick Bass, and MacKenzie Smith. DSpace as an Open Archival Information System: Current Status and Future Directions. In Traugott Koch and Ingeborg Sølvberg, editors, Research and Advanced Technology for Digital Libraries, 7th European Conference, ECDL 2003, Trondheim, Norway, August 17-22, 2003, Proceedings, Lecture Notes in Computer Science, pages 446–460. Springer-Verlag, 2003. ISBN 3-540-40726-X.

The Apache Software Foundation. Apache HBASE, 2013. URL http://hbase. apache.org/.

The PostgreSQL Global Development Group. PostgreSQL 9.0 Reference Manual. Network Theory Ltd., 2010.

Rik Van de Walle and Rob Koenen. *The MPEG-21 book*. New York: Wiley, 2006.

Hollie C. White, Sarah Carrier, Abbey Thompson, Jane Greenberg, and Ryan Scherle. The dryad data repository: a singapore framework metadata architecture in a dspace environment. In Proceedings of the 2008 International Conference on Dublin Core and Metadata Applications, DCMI '08, pages 157–162. Dublin Core Metadata Initiative, 2008. URL http://dl.acm.org/citation.cfm?id=1503418. 1503435.

Ian H. Witten, David Bainbridge, and Stefan J. Boddie. Greenstone - Open-Source Digital Library Software. D-Lib Magazine, 7(10), October 2001.