D 12.5 – Final Implementation Report

Version 1

26th January 2017

Grant Agreement number:	**313193**
Project acronym:	**ARIADNE**
Project title:	**Advanced Research Infrastructure for Archaeological Dataset Networking in Europe**
Funding Scheme:	**FP7-INFRASTRUCTURES-2012-1**
Project co-ordinator name, Title and Organisation:	**Prof Franco Niccolucci, PIN Scri - Polo Universitario "Città di Prato"**
Tel:	**+39 0574 602578**
E-mail:	**franco.niccolucci@gmail.com**
Project website address:	**www.ariadne-infrastructure.eu**

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7-INFRASTRUCTURES-2012-1) under grant agreement n° 313193.

Authors: **Dimitris Gavrilis, ATHENA RC**

Christos Papatheodorou, ATHENA RC

Eleni Afiontzi, ATHENA RC

Dimitra-Nefeli Makri, ATHENA RC

Costis Dallas, ATHENA RC

Contributing partners **Johan Fihn, SND**

Olof Olsson, SND

Carlo Meghini, CNR

Franca Debole, CNR

Nicola Aloia, CNR

Douglas Tudhope, USW

Ceri Binding, USW

Sebastian Cuy, DAI

Achille Felicetti, PIN

Table of Contents

Acronyms and abbreviations

ACDM	ARIADNE Catalogue Data Model (see Annex)
API	Application Programming Interface
BC/AD	Before Christ / Anno Domini, i.e. dates before / after Jesus's birth
CIDOC	International Committee for Documentation of the International Council of Museums
CIDOC-CRM	CIDOC - Conceptual Reference Model; ISO 21127:2006 standard: A reference ontology for the interchange of cultural heritage information
CRUD	Create, Read, Update, Delete operations conducted through a REST API (see below)
CSS3	Cascading Style Sheets (level 3)
DC	Dublin Core (metadata standard)
DCAT	Data Catalog Vocabulary, an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web
GIS	Geographical Information Systems / Services
HTML	HyperText Markup Language
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
ISO/IEC 11179	International standard for representing metadata for an organization in a metadata registry
Java / Javascript	Programming languages
JDCB	Java Database Connectivity
JSON	JavaScript Object Notation
lat/lon	Latitude and longitude coordinates
LDAP	Lightweight Directory Access Protocol
LOD	Linked Open Data
Log4J	A Java-based logging package
METS	Metadata Encoding and Transmission Standard
MySQL	An open source relational database management system
OAI	Open Archives Initiative

OAI-PMH	OAI - Protocol for Metadata Harvesting
OAI-ORE	OAI - Object Reuse and Exchange Specification
OAuth	Authentication protocol
ODBC	Open Database Connectivity
PHP	PHP Hypertext Preprocessor (server-side scripting language for making dynamic and interactive web pages)
RDF	Resource Description Framework
REST	Representational State Transfer
REST API	RESTful Application Programming Interface
REST services	RESTful web services
SAML	Security Assertion Markup Language
SKOS	Simple Knowledge Organization System
SKOSify	Transforming a thesaurus, classification system or other knowledge organization system into the SKOS format
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
Syslog	Standard for computer message logging
UI	User Interface
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WGS84	World Geodetic System 1984
XML	Extensible Markup Language
XSD	XML Schema Definition Language

1 Executive Summary

This document is a deliverable (D12.5 Final Implementation Report) of the ARIADNE project ("Advanced Research Infrastructure for Archaeological Dataset Networking in Europe"), which is funded under the European Community's Seventh Framework Programme. It presents results of the work carried out in Tasks: 12.3 "Implementing Integration".

The overall architecture of the ARIADNE infrastructure has been laid down by the user requirements analysis (D12.1), and the infrastructure specifications (D12.2). Initial implementation has already been described in D12.3. The goal of the ARIADNE infrastructure is to integrate data and metadata from different providers into one common schema, and also to provide semantic integration along different axes (e.g. subject, space, time). This integration intends to provide useful and user-friendly information services for archaeology. The services are intended to be available not only to researchers and related stakeholders, but also to a wider range of potential users requiring access to collections and datasets.

The main goal of Task 12.3 is to accomplish the infrastructure implementation of ARIADNE. The infrastructure is specified on D12.2 and includes services such as the registry, vocabulary services, metadata enrichment services, preservation services, etc. with the following components:

a) a triple store with semantic features
b) the ARIADNE portal
c) the MORe aggregation infrastructure
d) an indexer component (Elasticsearch)
e) a metadata quality measurement service
f) a set of enrichment services

The API specifications presented in the Annexes are especially important, as they allow developers to build on the infrastructure and deploy services. The APIs presented are REST-based and require data formats such as JSON, XML and RDF.

2 Objectives

The main objectives of WP12 are:

- To adapt infrastructures provided to ARIADNE for integration
- To design, implement and configure the necessary mechanisms (crosswalks, mappings) and resources for interoperability
- To set up the internal (APIs) and external (human) interfaces to access the integrated resources

WP12 comprises the following tasks:

- Task 12.1 Assessment of Use Requirements
- Task 12.2 Design and Specifications
- Task 12.3 Implementing Integration
- Task 12.4 Testing

The main goal of Task 12.3 is to present the infrastructure implementation of ARIADNE. The infrastructure is specified on D12.2 and includes services such as the registry, vocabulary services, metadata enrichment services, preservation services, etc. (fig. 1).

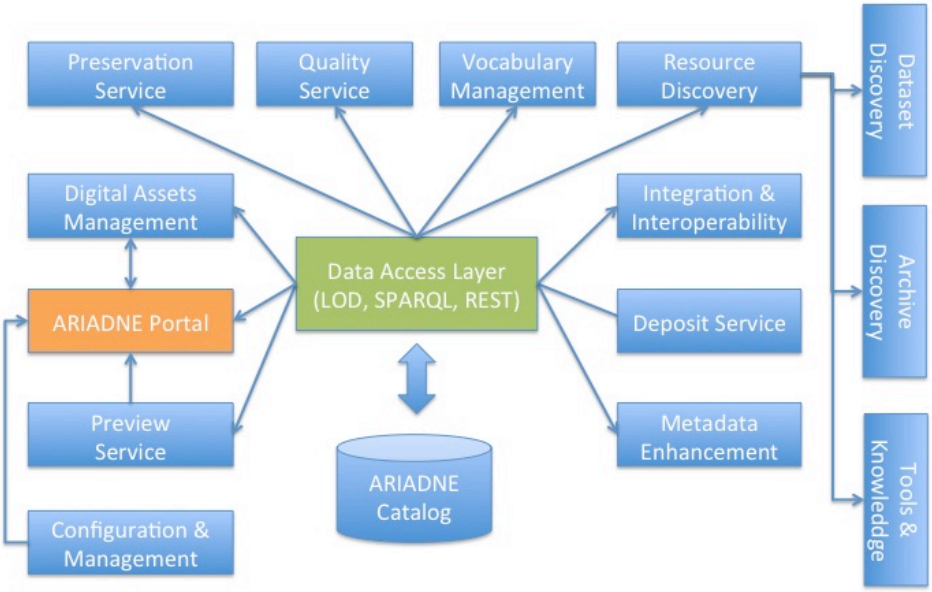


Figure 1. The ARIADNE data integration architecture.

This deliverable reports the results of Task 12.3. It consists of a report on the final implementation of the core components of the infrastructure.

3 ARIADNE Infrastructure Overview

The goal of the ARIADNE infrastructure is to integrate data and metadata from different providers into one common schema, and also to provide semantic integration along different axes (e.g. subject, space, time). This integration intends to provide useful and user-friendly information services for archaeology. The services are intended to be available not only to researchers and related stakeholders, but also to a wider range of potential users requiring access to collections and datasets. The overall architecture of the ARIADNE infrastructure has been laid down by the user requirements analysis (D12.1) and the infrastructure specifications (D12.2). Initial implementation has already been described in D12.3.

At Level 1, data is created by research projects and groups, and subsequently stored in institutional repositories at Level 2. Then, at Level 3, this data is aggregated by higher-level data managers, such as data centers, portals and thematic information gates. Finally, at Level 4, the ARIADNE infrastructure integrates all information and provides a set of services through the ARIADNE portal. These services include: resource discovery, integrated data search services, and other services (such as preservation, quality, etc.). Deliverable D12.1 "User Requirements" [1], which outlines the user needs for the interoperability framework of the ARIADNE Infrastructure, sets out a general view of the integration functionalities, organized in four successive levels (fig. 2) [1].

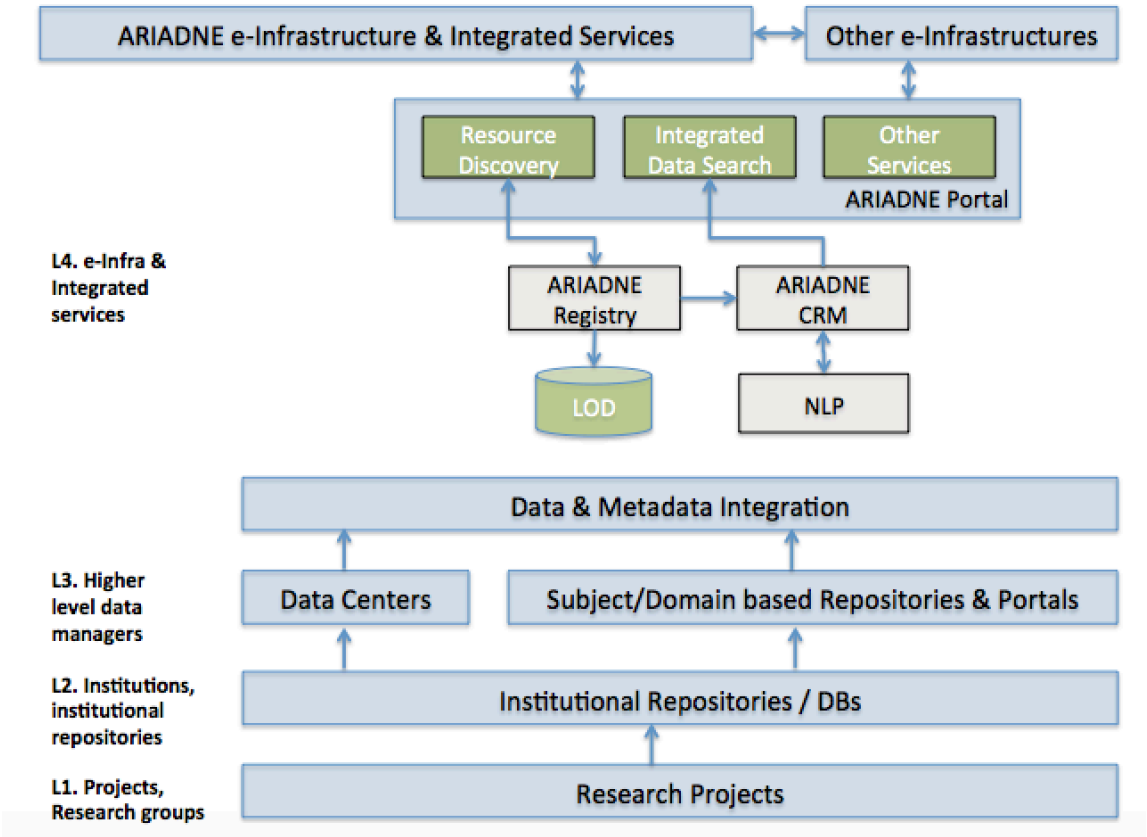


Figure 2. The ARIADNE data integration architecture.

The ARIADNE infrastructure services aim to provide useful services and tools to the archaeological community. Thus, a simple yet effective portal design is imperative in order to achieve these goals. This portal consists of three main pillars:

a) Resource discovery
b) Data integration services
c) Other services

The resource discovery services rely directly on the registry, whereas the data integration services provide more intelligent ways of data integration through LOD and NLP technologies or through the ARIADNE CRM (conceptual reference model).

The services that facilitate the integration (fig. 1) are described in detail in Deliverable D12.2.

The implementation of the infrastructure focuses on the creation and aggregation of content according to the ACDM model specification [4] (the current version is 2.6). According to the content aggregation workflow (fig. 3), the MORe aggregator (see respective section below) is used to harvest existing content in other schemas and map it to ACDM, or import directly from batches of XML or Excel files. ACDM encoded information is then pushed to an RDF store and to the Registry service. Another service is used to synchronize content from the RDF store to an Elasticsearch index service. The rest of the services within the infrastructure consume these three APIs in order to access/manage this information. The APIs are:

1. The RDF store SPARQL endpoint
2. The Registry REST API
3. The Elasticsearch API

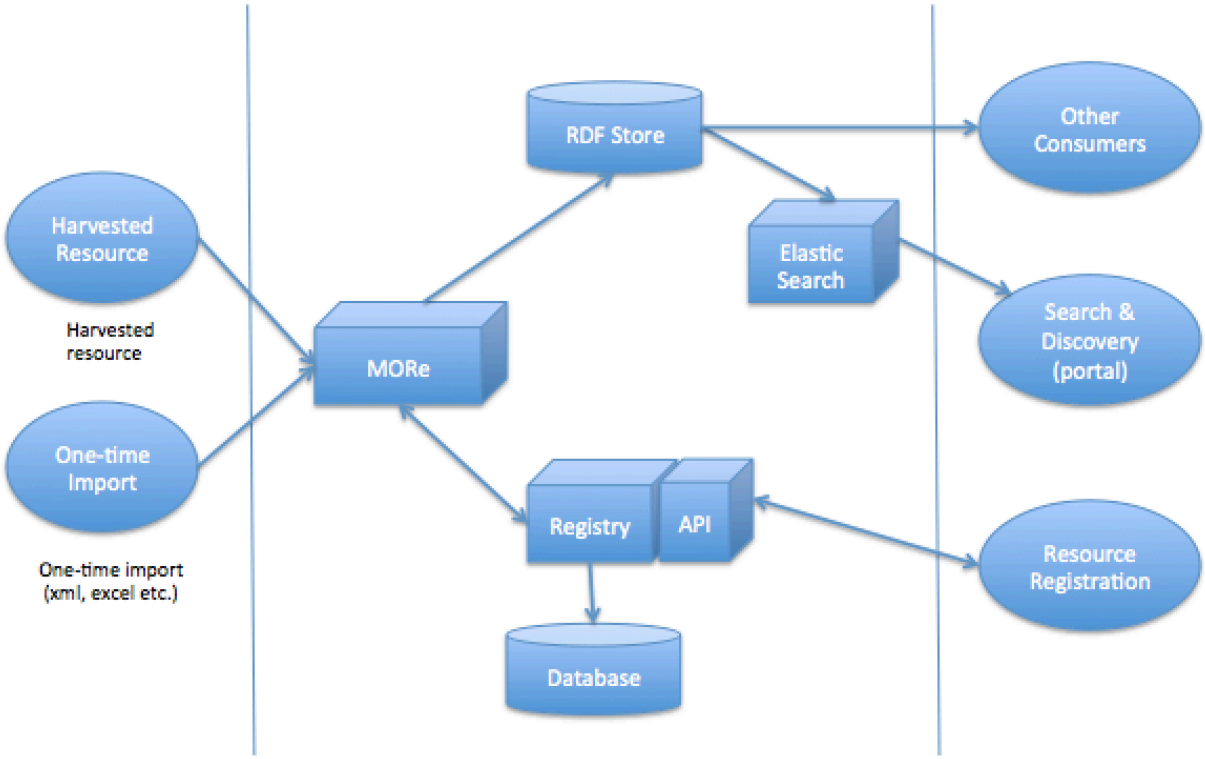


Figure 3. The ARIADNE data aggregation workflow.

4 ARIADNE Services for Data Collection and Integration

In ARIADNE the aggregation service is comprised of a set of tools and services that aim to harvest content related to archaeology, and to clean and transform it to a common format (in this case the ACDM). This content is then stored persistently in an RDF store and an index server. This process is integral to the data integration process of the project, which focuses on integration of data scattered amongst diverse collections, datasets, unpublished fieldwork reports (grey literature), and publications. This provides researchers with the ability to use heterogeneous distributed datasets as an integral component of archaeological research methodologies. Content aggregation is necessary in order to map all content available by the ARIADNE partners in a common schema (in this case, the ACDM), so as to facilitate the integration process. Aggregation includes a number of steps, such as validation, cleaning and enrichment.

In order to address these requirements, the Metadata & Object Repository (MORe) aggregator [6] has been employed in ARIADNE (http://more.dcu.gr/). The MORe aggregator, developed by the Digital Curation Unit – IMIS, Athena R.C., has been effectively used in diverse projects (CARARE, 3D-ICONS, and LoCloud) to aggregate and enrich millions of records and deliver them to the Europeana platform (http://www.europeana.eu). MORe provides a flexible architecture, and is used here to aggregate content in various formats, and from a variety of sources, to the RDF store, the registry and the Elasticsearch. MORe incorporates a micro-service oriented architecture (fig. 4), supporting core services for input, validation, transformation, eEnrichment and publication of the ingested content.

The core services are designed in a modular way so they can combine one or more micro-services to accomplish the various tasks. This approach allows the system to be extended more easily, and also re-use certain micro-services. The stackable and modular architecture of MORe (fig. 4) presents a data access layer used to access data on the storage and a core services layer, encompassing the various core services management. The different core management services are responsible for orchestrating numerous micro-services. For example, the enrichment management services are responsible for orchestrating and streamlining the execution of the enrichment micro-services.

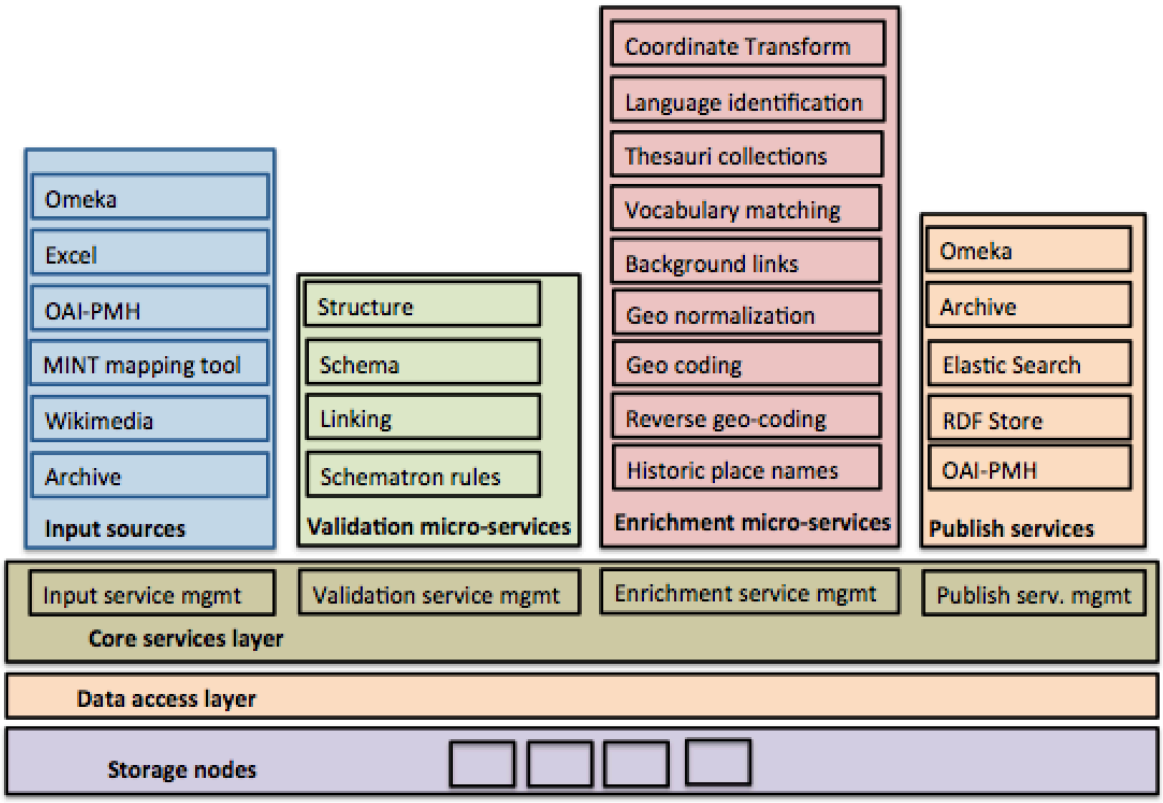


Figure 4. The MORe architecture.

Other important architectural aspects of the system (fig. 5) provide the capability of using multiple storage technologies simultaneously. The data access layer, which is instantiated through a storage API, provides seamless access to data objects regardless of the storage in which they reside. Two additional important components of MORe are the messaging and logging services. The messaging service is responsible for providing messaging across services. It is based on the RabbitMQ AMPQ broker, a framework that provides scalability and elasticity. The logging service provides a mechanism for monitoring all the services in the aggregation infrastructure. It is implemented based on the Graylog2 logging framework, which provides a mechanism for storing logs of any format and from any service. Graylog2 provides a way of organizing these log messages (using streams) and presenting them in a user-friendly way.

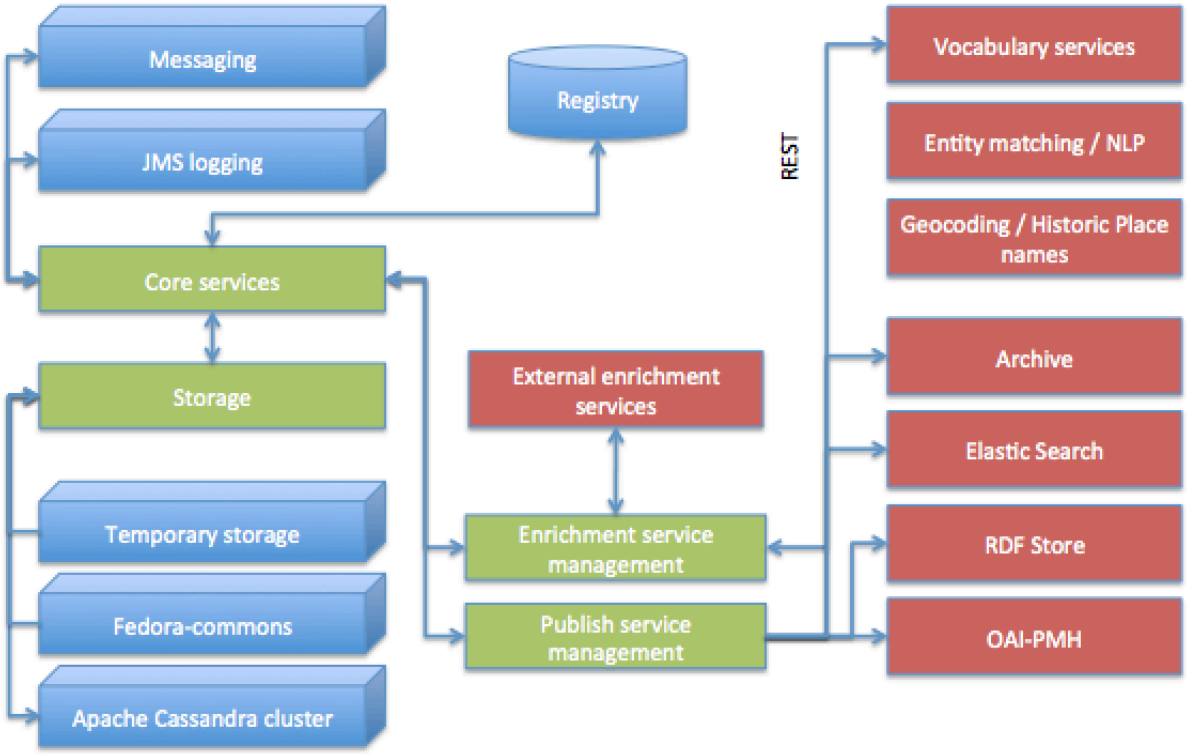


Figure 5. Significant architectural aspects of MORe.

The main aggregation workflow is presented below (fig. 6). An information package is harvested by a micro-service and is pre-validated (pre-validation includes basic validation, such as a record to be structurally checked to see if it is well formed). The package can be either deleted or ingested. During ingest, the package is stored in the storage layer (it first goes into temporary storage), and then it can be either rejected or transformed into a common schema (in this case the ACDM). During transformation, the new representation is validated and indexed. After it has been transformed, the package can be rejected, enriched or published. In the case of enrichment, a new representation is created (enriched ACDM), which has to be validated and indexed again. During the publishing process the user selects the publication target.

In the case of ARIADNE, the intermediate schema is ACDM. The publication targets that may be used are:

- Publish to Elasticsearch
- Publish to Virtuoso Triple Store
- Publish to Registry API

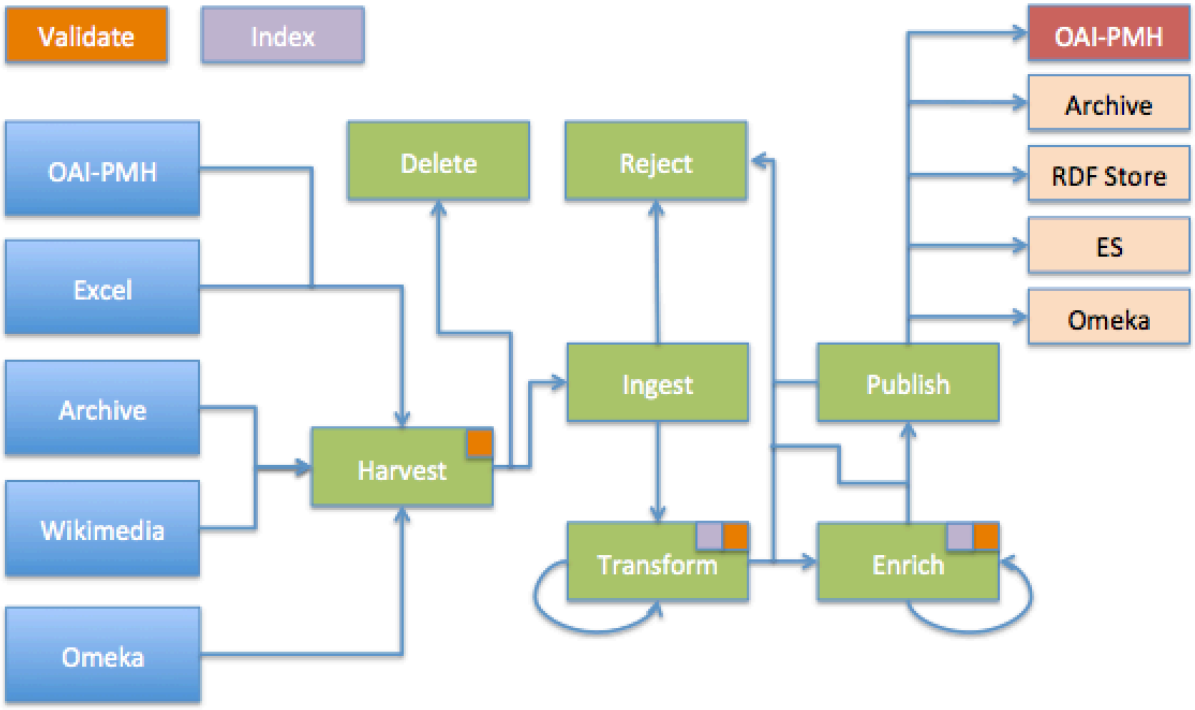


Figure 6. MORe Aggregation workflow.

5 ARIADNE Services for Data Access and Management

The ARIADNE services for data access and management consist of a layer that provides different ways of allowing access to the available data sources. These are native access to the RDF store, access through an Elasticsearch, and access through the Registry. Although they access the same information, each might contain only a subset (e.g. Elasticsearch), or provide access through different protocols and formats.

Furthermore, as ARIADNE enables trans-national access of researchers to data centres, tools and guidance, it is important to provide multiple access points in the data sources. This means that through the data access management services, researchers can preview their datasets in different formats, such as RDF, JSON or XML.

The Semantic Triple Store

ARIADNE implements an RDF store to hold the catalog information represented in RDF and available through a SPARQL endpoint. Taking into account the complexity of the ACDM, an RDF representation of

the ARIADNE Catalog is necessary to support the complex queries required. For that reason, the ACDM can be represented as RDF and URIs are assigned for each type of resource.

The data in the RDF store is updated by MORe dynamically during content aggregation. The Virtuoso semantic technology (http://virtuoso.openlinksw.com/) has been used for implementing the ARIADNE triple store. Virtuoso was selected primarily due its better performance, especially when handling large amounts of content. In addition, Virtuoso has been available since 1998, as opposed to Sesame, which was released in 2004. Virtuoso is also available in many programming languages (.Net, C, C#, C++, Java, JavaScript, Perl, PHP, Python, Ruby, Visual Basic), whereas Sesame is available to Java, PHP and Python. The most important characteristics that make Virtuoso more favourable over Sesame are its capability to partition the data and being able to be setup in a clustered environment. The SPARQL endpoint and a screenshot can be seen below (fig. 7).

SPARQL Console for Virtuoso:

http://ariadne-registry.dcu.gr:8890/sparql

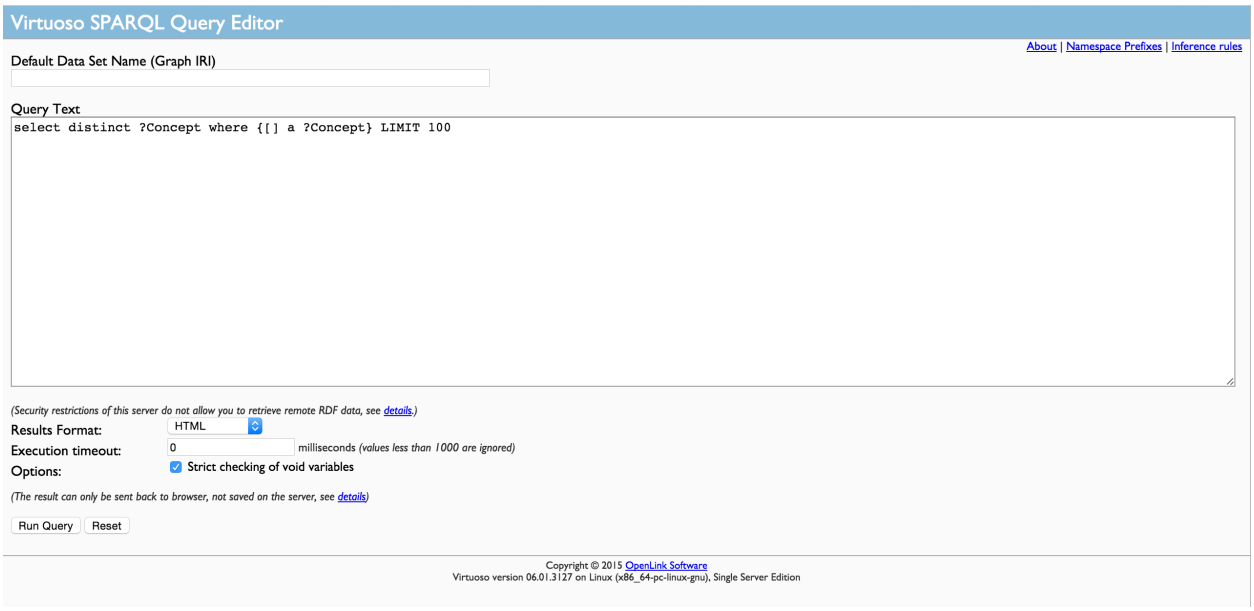


Figure 7. Screenshot of the ARIADNE SPARQL endpoint.

In order to store the entire catalog in RDF, an RDF representation of ACDM must be defined. This has been accomplished through a simple mapping process handled internally by the MORe aggregator.

The Indexing and Search Engine

The Elasticsearch component provides the indexing capabilities required in order to power the ARIADNE portal. Elasticsearch (ES) is based on the well-known Lucene indexing engine (the same that SolR uses) and provides extremely low response times for performing full text search and faceted browsing. The two main reasons for selecting Elasticsearch over other traditional interfaces to SolR are its simple and effective RESTful API (fig. 8) and its clustering capabilities. Elasticsearch provides very robust and easy to use clustering support.

The primary format that Elasticsearch accepts through its API is JSON. That means that all ACDM records must be first mapped into JSON in order to be pushed to ES.

Elasticsearch REST API:

http://ariadne-registry.dcu.gr:9200/_search

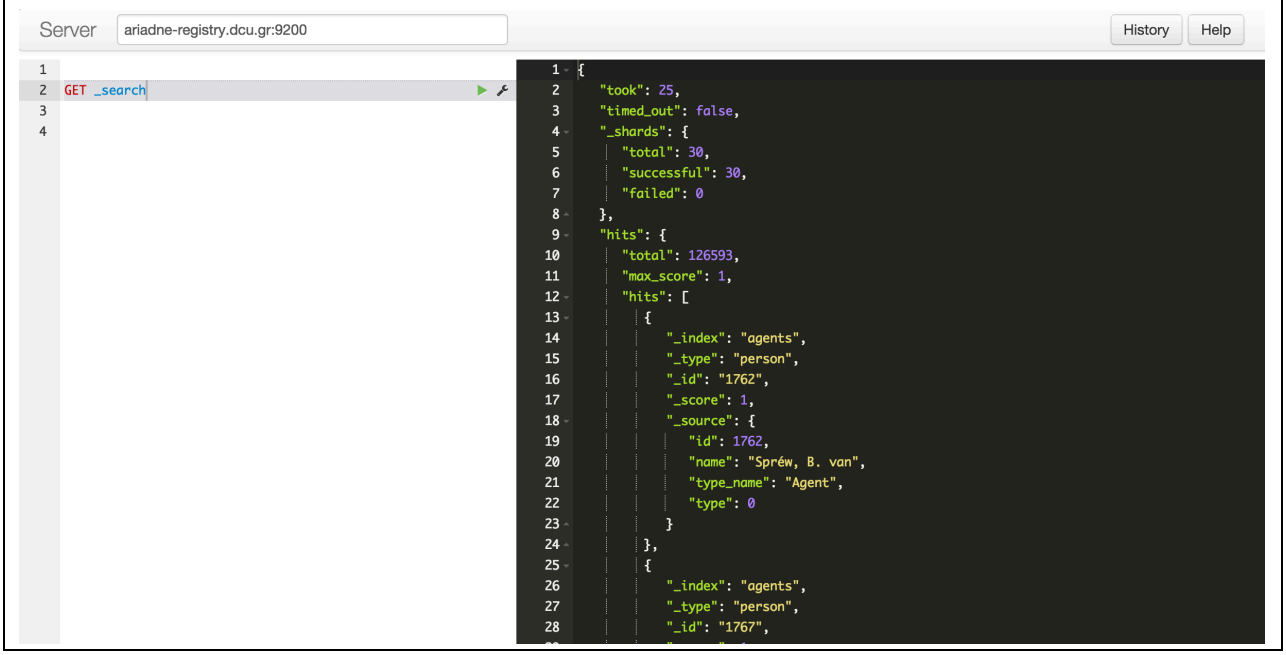


Figure 8. Screenshot showing Elasticsearch REST interface through Web http client.

Since the primary mission of ES is to facilitate the Resource Discovery section of the ARIADNE Portal, only the features associated with this functionality are kept. Furthermore, the search results should be specified in a way that minimises subsequent API calls. An example of such a record can be seen in the table 1.

Table 1. Example of a JSON representation of an ACDM record

```
{
_index: "dataresources",
```

```
_type: "dataset",
_id: "46949",
_score: 1,
_source: {
     accessRights: "Unrestricted access for all registered EASY users.",
     modified: "2010-01-11",
     distribution: {
          id: 46,
          title: "EASY OAI-PMH",
          modified: "2014-01-01",
          issued: "2014-01-01",
          description: "EASY OAI-PMH Endpoint",
          OAI-PMHServerURI: "http://easy.dans.knaw.nl/oai/",
     },
     type: "dataset",
     description: "Onderzoeks- en adviesbureau voor Bouwhistorie, Archeologie,
Architectuur- en Cultuurhistorie (BAAC bv) heeft een archeologisch
bureauonderzoek en inventariserend veldonderzoek met behulp van boringen
(karterende fase) uitgevoerd op een terrein in de bebouwde kom van Tilburg.
Tijdens het veldonderzoek is aangetoond dat de bodem is verstoord tot in de
Chorizont van de dekzandafzettingen. Archeologische indicatoren werden
daarbij niet aangetroffen. De verwachte duinvaaggronden zijn niet
aangetroffen. Hieruit kan worden geconcludeerd dat aan het plangebied een
lage archeologische verwachting voor archeologische resten uit alle perioden
kan worden toegekend.",
     rights: "BAAC bv",
     id: 46949,
     language: ["nl"],
     subject: "Event/intervention databases",
     title: "Tilburg Bredaseweg 421",
     originalId: "DMO_ID easy-dataset:22059",
     keyword: ["50F", "Bredaseweg 421", "Tilburg", "Noord-Brabant"],
     creator: [{
          id: 480,
          name: "Boshoven, E.H.",
          type: "person"
     }],
     spatial: [{
          id: 34449,
          coordinate_system: "http://www.opengis.net/def/crs/EPSG/0/4326",
          lon: 5.04075,
          label: "51.55941102,5.04075138",
          lat: 51.5594
     }],
     landingPage: "http://www.persistent-identifier.nl/urn:nbn:nl:ui:13-dox-
y9f",
     issued: "2008-01"
}
}
```


In order to provide accurate and robust thematic based results, an index with the complete AAT terms was created. This index contains the complete AAT hierarchy plus the content provider mappings from their native subjects to AAT. An example of an AAT is presented in fig. 9.

```
{
  "_index": "aat",
  "_type": "terms",
  "_id": "300201790",
  "_score": 1,
  "_source": {
    "prefLabels": [
      {
        "label": "onderarmstukken",
        "lang": "nl"
      },
      {
        "label": "avambrazos",
        "lang": "es"
      },
      {
        "label": "vambraces",
        "lang": "en"
      },
      {
        "label": "arambrazo",
        "lang": "es"
      },
      {
        "label": "arm guards",
        "lang": "en"
      },
      {
        "label": "arm-guards",
        "lang": "en"
      },
      {
        "label": "avambrazo",
        "lang": "es"
      },
      {
        "label": "vambrace",
        "lang": "en"
      },
      {
        "label": "guards, arm",
        "lang": "en"
      },
      {
        "label": "onderarmstuk",
        "lang": "nl"
      }
    ],
    "prefLabel": "vambraces",
    "providerMappings": [
```

```
            {
              "sourceURI": "http://purl.org/heritagedata/schemes/mda_obj/concepts/97103",
              "matchURI": "http://www.w3.org/2004/02/skos/core#broadMatch",
              "providerId": 1104,
              "sourceLabel": "ARM GUARD"
            }
          ],
          "id": "300201790",
          "broader": [
            {}
          ],
          "uri": "http://vocab.getty.edu/aat/300201790",
          "altLabels": [],
          "scopeNote": "Term applied to the ensemble of plate armor pieces for the arm below the shoulder, consisting of two cannons linked by a cowter, in use in Europe from the 14th to the 17th century. Also sometimes used to mean cannons worn on the forearm only, including such ancient Greek pieces."
        }
      }
```


Figure 9. The record of an AAT term.

The ARIADNE Registry

The registry service is a single point where all ARIADNE resources are stored and made available. It provides access to all ACDM records, and allows users to create/edit them both through a REST interface and through a Web UI. The registry consists of an SQL database and a REST API that provides access to the data. The users can register an API key and use the REST API to perform CRUD operations on all entities. An example of an API call is as follows:

Get Data Resources

Get all the Data Resources of the registry using some filters if necessary.

Method	GET http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/dataResources
Content Type	application/json
Request	**Parameter** / **DataType** / **Description** key / String / The API key of the user typeId / Integer / The type of the Data Resources (0:Collections 1:Datasets 2:Databases 3:GIS)
Response	**Parameter** / **DataType** / **Description** id / Integer / The id of each Data Resource name / String / The name of each Data Resource
Status	200 – SUCCESS 500 – INTERNAL_SERVER_ERROR

Example

Request: {"key":"tUzQwU2T1RqBRuHC1ENJmtaopTMfWF6sHE7OrjRPzr98DiB","typeId":"1"}
Response: [{"id":20,"name":"CulturaItalia - Archaeology"},{"id":33,"name":"Archaeological materials"}]

The full API documentation is provided in Annex II. The Web UI has been presented and documented in the ACDM model specification report.

Users without any content in an existing format and available through a repository can use the registry service to catalogue content from scratch, manually or through the REST API.

6 ARIADNE Services for Data Validation

This section documents the list of services that could be used to perform validation on the ACDM XSD. Validation is performed during content ingest and creation, and is facilitated automatically by MORe. All validation services provide a reporting mechanism through the UI to alert the user.

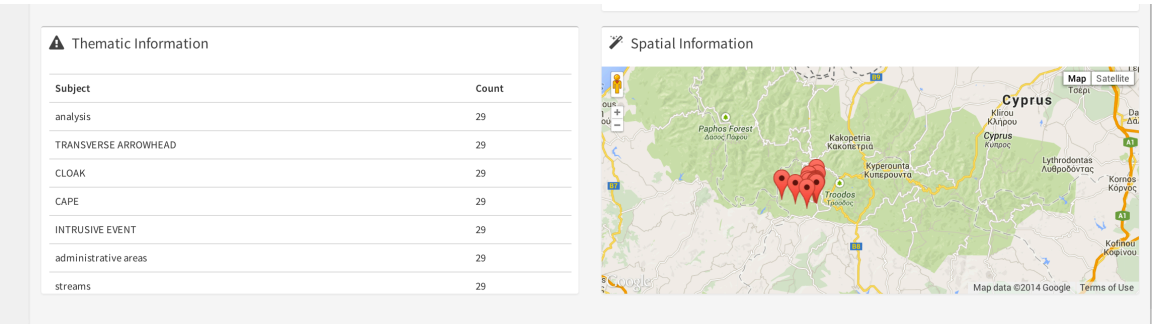


Figure 10. Screenshot of the thematic and spatial validation service.

Apart from the standard validation reporting mechanism, when a certain type of information is detected (spatial, temporal, thematic), respective blocks of information appear alerting the user visually. A screenshot presenting an example for thematic and spatial information can be seen in fig. 10.

Schema Validation

This service provides schema validation for every XML record ingested. This functionality is automatically provided by MORe. Schema validation requires an XSD (in this case ACDM XSD).

Normalization

This service performs normalization of the ingested datasets. This process ensures the proper identity management of each resource by matching provider and collection information, along with native identifiers, etc (fig. 11).

Link Checking

The link checking service requires a list of elements that contain URLs, for which it attempts to check if the links are available (or, else, it returns an HTTP error code). This service is already provided by MORe, but requires a list of elements to check. Elements that may contain links and need to be checked have to be marked beforehand.

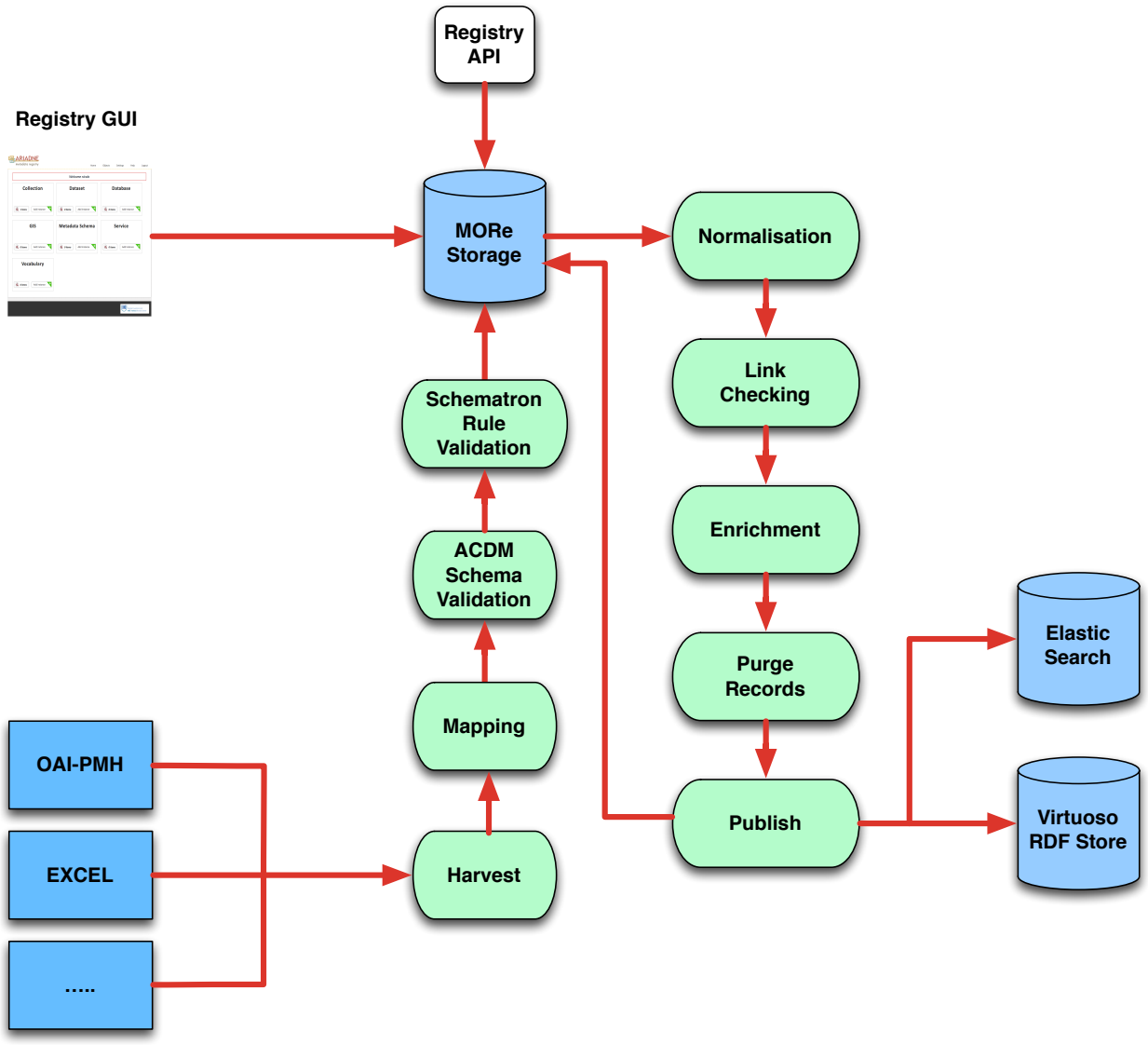


Figure 11. The aggregation workflow normalization perspective.

Schematron Rule Validation

This service performs more complex checks on an XML record. It provides a mechanism to perform complex conditional checks, and also provide feedback back to the user in a human-readable mechanism. The mechanism is provided by MORe but requires a Schematron rule for ACDM. A screenshot of an indicative Schematron rule validation report can be seen in fig. 12.

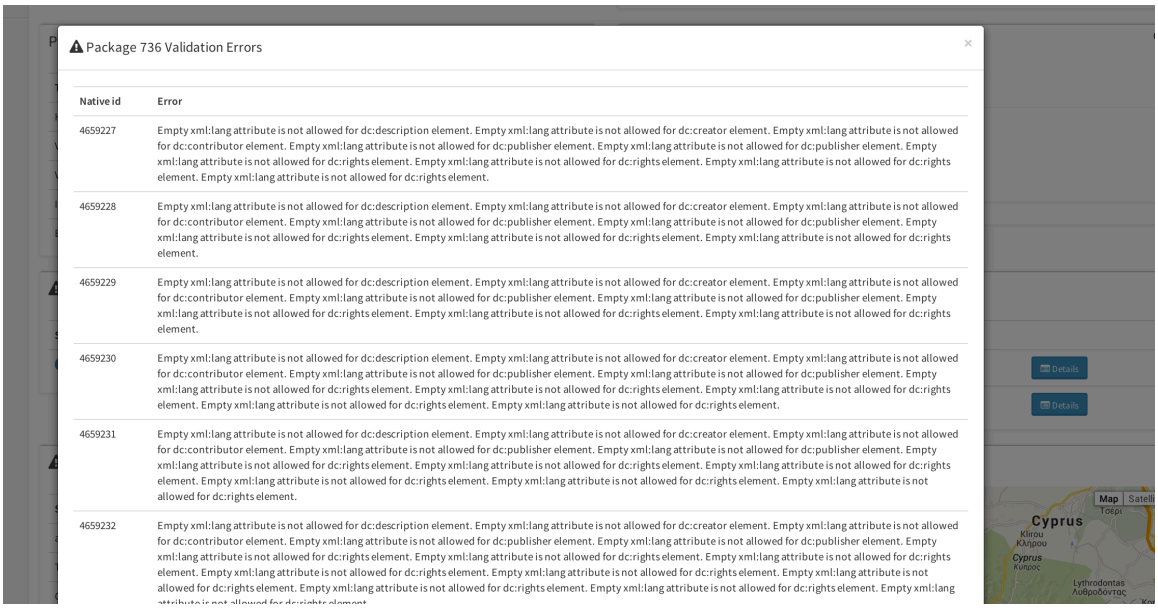


Figure 12. Screenshot of a sample Schematron rule validation report.

7 ACDM Metadata Enrichment

The system provides a service for metadata enrichment that consists of a series of micro-services that can be used effectively to augment ACDM content. These micro-services can be either used as enrichment services in MORe, or directly by the ARIADNE partners and services. In this section, a list of available micro-services is presented.

Once the metadata enrichment services are registered in MORe, users can utilize them by incorporating them into enrichment plans. Enrichment plans are a way of streamlining and executing enrichment services together. They apply to one schema and can be re-used in different packages. This means users can define them once, and apply them for all incoming packages with the same characteristics.

Although enrichment services can perform a wide variety of operations, only the main ones are presented in this section, and in particular those concerning subject, special and temporal enrichment.

Subject Enrichment

The thematic services focus on enriching subject-related information. This enrichment process can either be automatic (e.g. by automatically matching concepts from vocabularies) or manual (by allowing users to create collections of concepts and attach them to all items in a package). All thematic enrichment services have access to the large number of vocabularies listed in Annex III. These vocabularies are standardised, published and SKOSified, and contain hundreds of thousands of terms.

The browsing and searching for these vocabularies can be accomplished either through a REST API or through the MORe UI. An indicative screenshot of the latter case can be seen in fig. 13.

Other subject-related services have access to other resources, such as Wikipedia and DBpedia lemmas.

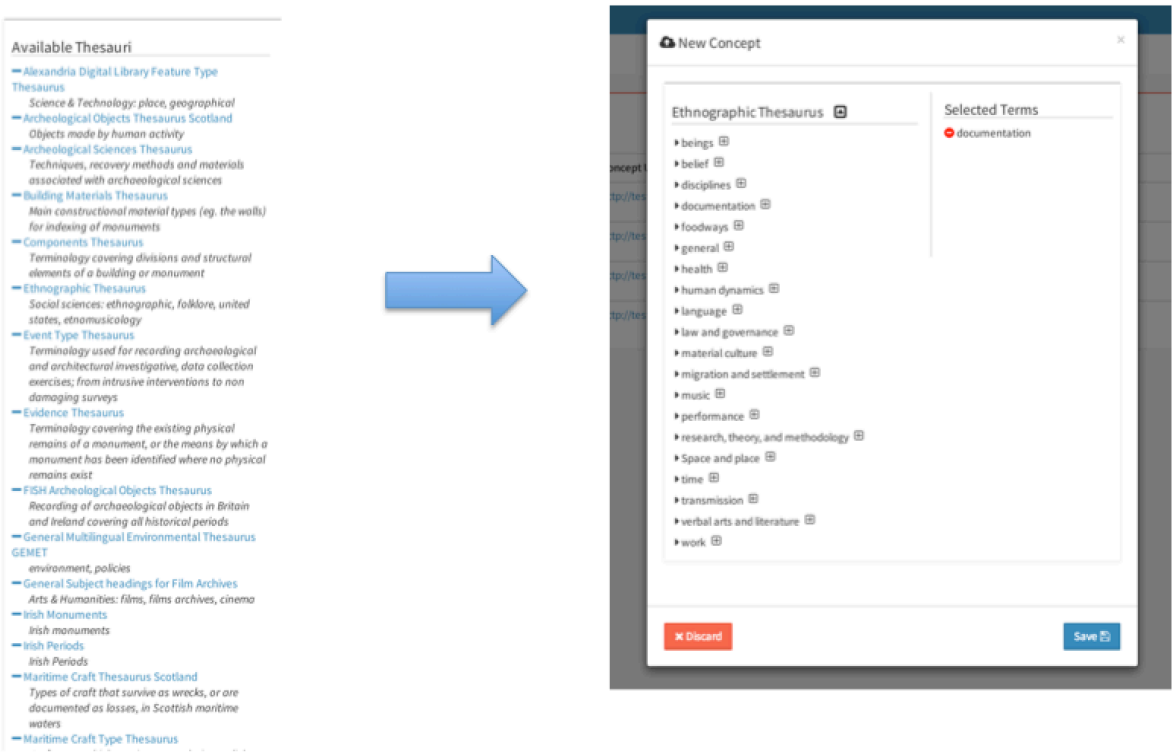


Figure 13. Screenshot of a browsing for vocabularies in the MORe UI.

In the following tables, the implemented subject related enrichment services are presented.

Service:	**Subject collections**
Status:	Provided as a service through MORe
Description:	This service allows the user to create collections of subject terms from standardized thesauri. These subject collections are then used to enrich the items of a package (by adding the concepts of the collection to the items of the package).

Service:	**Automatic metadata enrichment**
Status:	Provided as a service through MORe
Description:	A string (e.g. dc:subject or dc:description) is submitted to the service and a list of relevant concept terms are retrieved (based on a list of 29 SKOS vocabularies). The element dc:subject is updated.

Service:	**DBPedia and Wikipedia automatic enrichment**
Status:	Provided as a service through MORe
Description:	A string (e.g. dc:title, dc:subject or dc:description) is submitted to the service and a list of relevant Wikipedia and DBPedia URLs are retrieved. An open-annotations based list (RDF encoded) is returned that annotates the text.

Service:	**Subject mappings**
Status:	Provided as a service through MORe
Description:	This service allows the user to upload or create a list of mappings from their own native subject terms (found in their provided packages) to AAT. These mappings are used to enrich the incoming packages with AAT terms.

AAT Mapping and Conversion of Subject Terms

In order to ensure interoperability and facilitate integration, all ACMD records must contain AAT terms (AAT stand for Arts and Architecture Thesaurus and refers to the faceted thesaurus developed by the Getty Foundation, and intended for "vocabulary control of museum, cultural heritage and art collections"). Thus, a translation service, developed by the Hypermedia Research Group at the University of South Wales, is provided to allow partners to upload their native subject terms/vocabularies, map them to the AAT, and export a mapping file. This mapping service produces the mapping rules in various formats (such as JSON and CSV). The tool is available at:

http://heritagedata.org/vocabularyMatchingTool/

The service comes as a web application (thus only a web browser is needed) and provides an intuitive UI interface (fig. 14) consisting mainly of two columns (left for the source vocabulary, right for the target vocabulary).

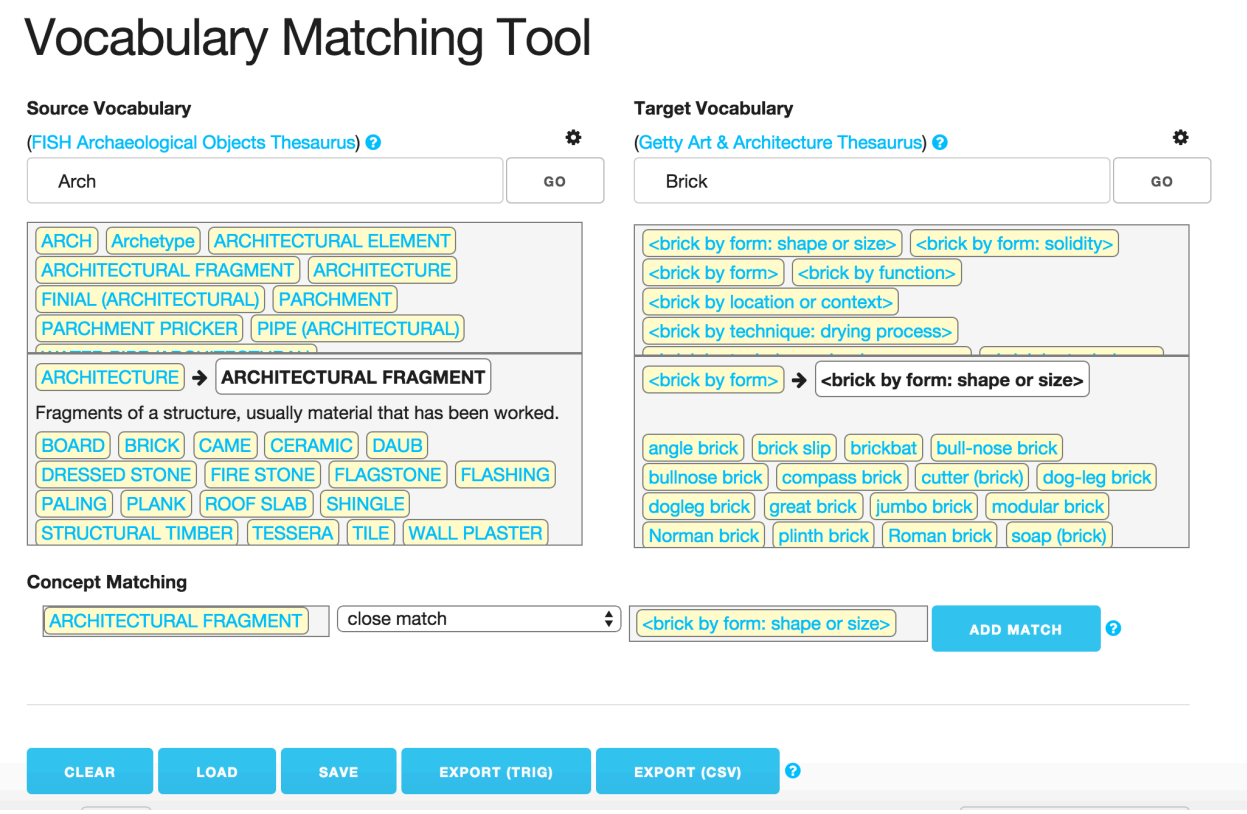


Figure 14. Screenshot of the Vocabulary Matching Tool.

Users can define the concept mappings, then save and export them in various formats. An example of a JSON encoded mapping can be seen in Table 2. This mapping could be also uploaded into the MORe subject translation enrichment service and used to automatically transform native subject terms to AAT concepts.

Terminological Information for the ARIADNE Infrastructure

The use of thesauri, and especially the required use/existence of at least 1 AAT term per record, adds a certain complexity that requires special handling by the infrastructure. Furthermore, hands-on testing of the portal (e.g. through running different queries) brought out the need for semantic expansion and use of multilingual search. The information flow (regaring subjects) can be seen in fig. 15, in which content providers provide two kinds of subject terms:
a) native subjects (could be anything, SKOSified, simple terms, etc)

b) provided subjects (have to be AAT)

Providers that do not provide AAT terms have to map their native subject terms to AAT using the vocabulary mapping tool. This tool in turn feeds all the mapping information to MORe and associates it with the provider. Thus, when a package from a provider is received that contains native subjects and a mapping exists, the derived subject is automatically computed and used to enrich the record.

Table 2. Example of a JSON representation of an mappingrd


```
[
 {
  "created": "2015-04-14T13:57:18.206Z",
  "sourceURI": "http://purl.org/heritagedata/schemes/mda_obj/concepts/96909",
  "sourceLabel": "ADZE",
  "targetURI": "http://vocab.getty.edu/aat/300023553",
  "targetLabel": "adzes",
  "matchURI": "http://www.w3.org/2004/02/skos/core#closeMatch",
  "matchLabel": "close match",
  "notes": "",
  "creator": "anonymous"
 },
 {
  "created": "2015-05-11T14:25:15.011Z",
  "sourceURI": "http://purl.org/heritagedata/schemes/mda_obj/concepts/95253",
  "sourceLabel": "COMPASS",
  "targetURI": "http://vocab.getty.edu/aat/300196707",
  "targetLabel": "compasses (direction indicators)",
  "matchURI": "http://www.w3.org/2004/02/skos/core#closeMatch",
  "matchLabel": "close match",
  "notes": "",
  "creator": "anonymous"
 }
]
```

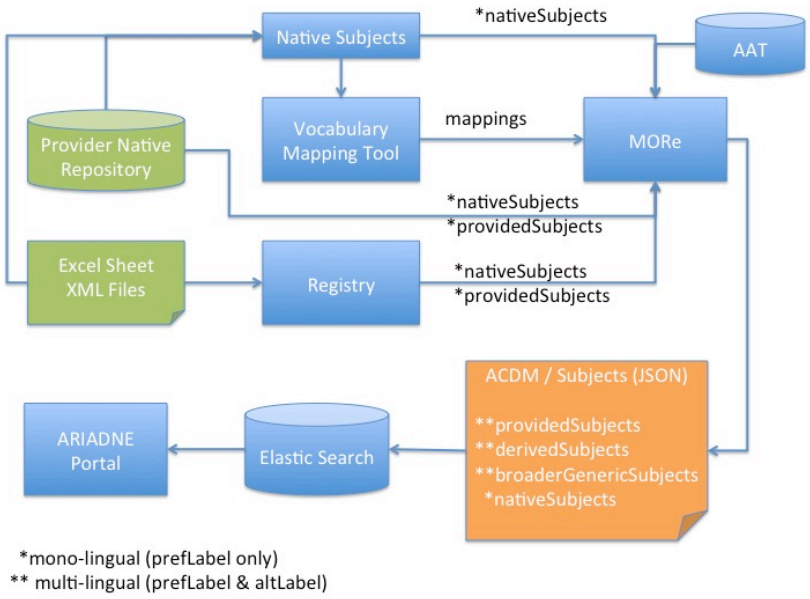


Figure 15. Handling of subject terms through the ARIADNE infrastructure.

During the enrichment and publication process the AAT thesaurus is used to perform the following for each ACDM record:
- expand the AAT terms in order to include the broaderGeneric term
- expand the AAT terms in order to include the altLabels
- expand the AAT terms in order to include the multi-lingual labels (prefLabels and altLabels) in order to facilitate multi-lingual search.

As mentioned, in the elastic search a separate AAT index is published that includes the provider mappings (under each AAT term the native subjects are associated according the mappings).

Geographic Enrichment

The space related enrichment services focus on enriching spatial content and currently they perform four main tasks:

- Geo-coding
- Reverse geo-coding
- Coordinate translation
- Geo-normalization

The various available micro-services can be seen in the following tables.

Service:	**Geonames**
Status:	Provided as a service through MORe
Description:	1. A string (e.g. place Label) is submitted for geo-coding and the Latitude/Longitude coordinates are returned. The lat/lon elements are updated. 2. A set of lat/lon coordinates are submitted and a place name is returned. The place name is updated. API available at: http://api.geonames.org/

Service:	**DAI Gazetteer**
Status:	Provided as a service through MORe
Description:	A string (e.g. placeLabel) is submitted for geo-coding and the lat/lon coordinates are returned. The lat/lon elements are updated. API available at: http://gazetteer.dainst.org/search

Service:	**Pleiades Plus**
Status:	Provided as a service through MORe
Description:	The Pleiades Plus ancient place names service allows resolving ancient place names to modern names. API available at: https://github.com/ryanfb/pleiades-plus

Service:	**British National Grid → WGS84 Transformation**
Status:	Provided as a service through MORe
Description:	A string containing the grid coordinates is submitted for transformation and the WGS84 lat/lon coordinates are returned. The lat/lon elements are updated.

Service:	**Transformation between EPSG codes**
Status:	Provided as a service through MORe
Description:	The lat/lon coordinates are transformed from an EPSG code to another. The lat/lon elements are updated.

Service:	**Geo-normalization**
Status:	Provided as a service through MORe
Description:	A string that contains a concatenated string of coordinates is submitted for geo-normalization. The user indicates the delimiter and the lat/lon coordinates are returned. The lat/lon elements are updated.

Temporal Enrichment

ARIADNE partners have contributed to the PeriodO gazetteer of period assertions, and PeriodO URIs are available for integration with MORe. More information can be found at http://perio.do/.

Service:	**PeriodO**
Status:	Provided as a service through MORe
Description:	The PeriodO service provides a SKOSified collection of period names. This makes it possible to resolve period names to absolute dates. Period names are encoded differently depending parameters such as the geographical coverage.

Other Services

This section currently includes a language identification service to automatically detect language information based on text. It is based on Apache Tika and contains trained models for all major languages.

Service:	**Language Identification**
Status:	Provided as a service through MORe
Description:	This service detects the language in which a piece of text is written. A text element is submitted and the language is returned. The xml:lang attribute is updated.

8 ARIADNE Services for Metadata Quality Checking

This section aims at documenting the list of services that could be used to provide quality related information for ACDM records. These metadata quality services mainly focus on completeness and consistency.

Metadata Completeness

Completeness refers to metadata completeness, estimated for an entire package (at item level) and falls under two separate cases:

1. Completeness of important elements
2. Completeness of mandatory/recommended element sets

The first case aims to visually provide the completeness of a list of the 10-15 most important elements of the ACDM (fig. 16).

The second case aims to provide information on the completeness of all elements, using two facets: mandatory and recommended sets (fig. 17).

Figure 16. Metadata quality information provided to the user.

Metadata Consistency

This service allows to check certain elements for consistency issues. Consistency refers to the following cases:

1. URL format checking (for example: elements like rdf:about should have a URL as value)

2. Date format (based on ISO dates format)
3. Person names (regex patterns that could detect patterns such as: Name Surname, Title Name Surname, etc.)

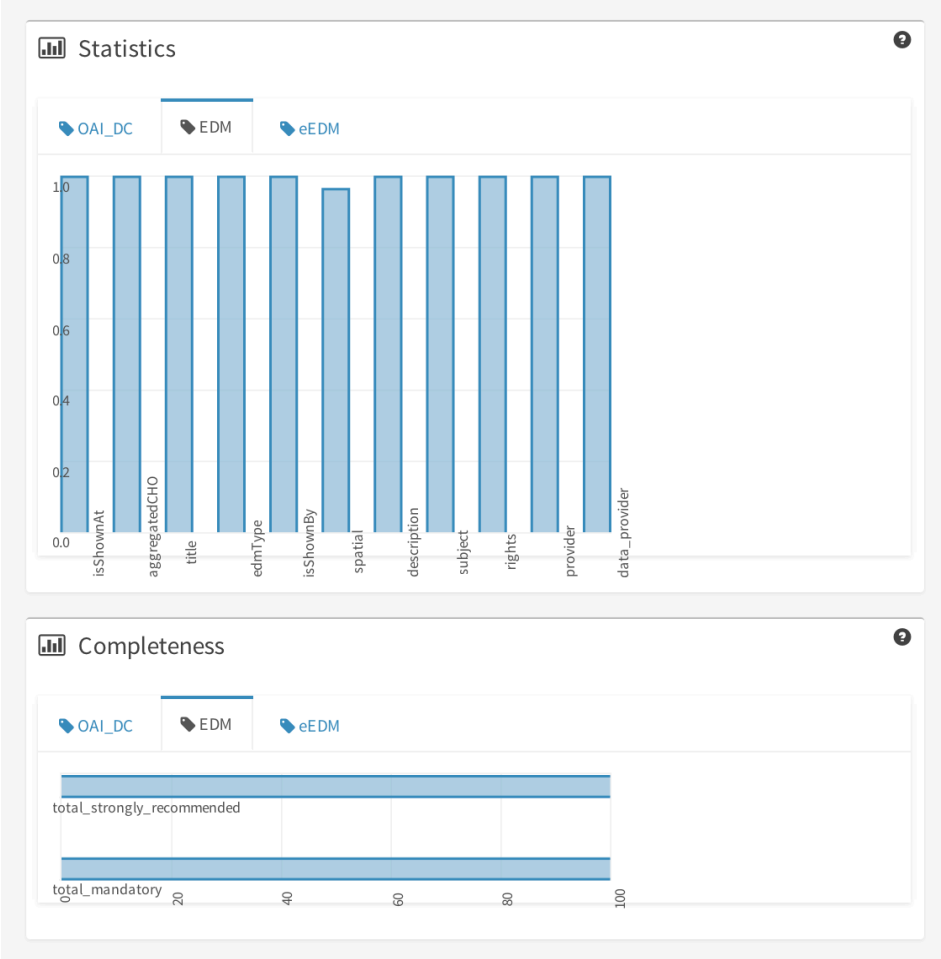


Figure 17. Metadata quality information provided to the user.

9 The ARIADNE Portal

This section presents the final implementation of the ARIADNE portal.

Portal Mock-ups

A series or mock-ups were created to start a discussion on the design and functionalities of the ARIADNE portal, and to act as a guideline for implementation. These mockups presented a design with three different sections: Browse, Search and Services. The Browse section was intended to act as a starting point for exploring the ARIADNE Info Space in different geo, time and subject based visualizations. The Search section presents a classic information retrieval interface for the different types of resources in the ARIADNE Catalog, and is intended to be complemented using facet-based search (fig. 18 – fig. 22). The Services section provides access to the different services developed in the project as part of WP13.

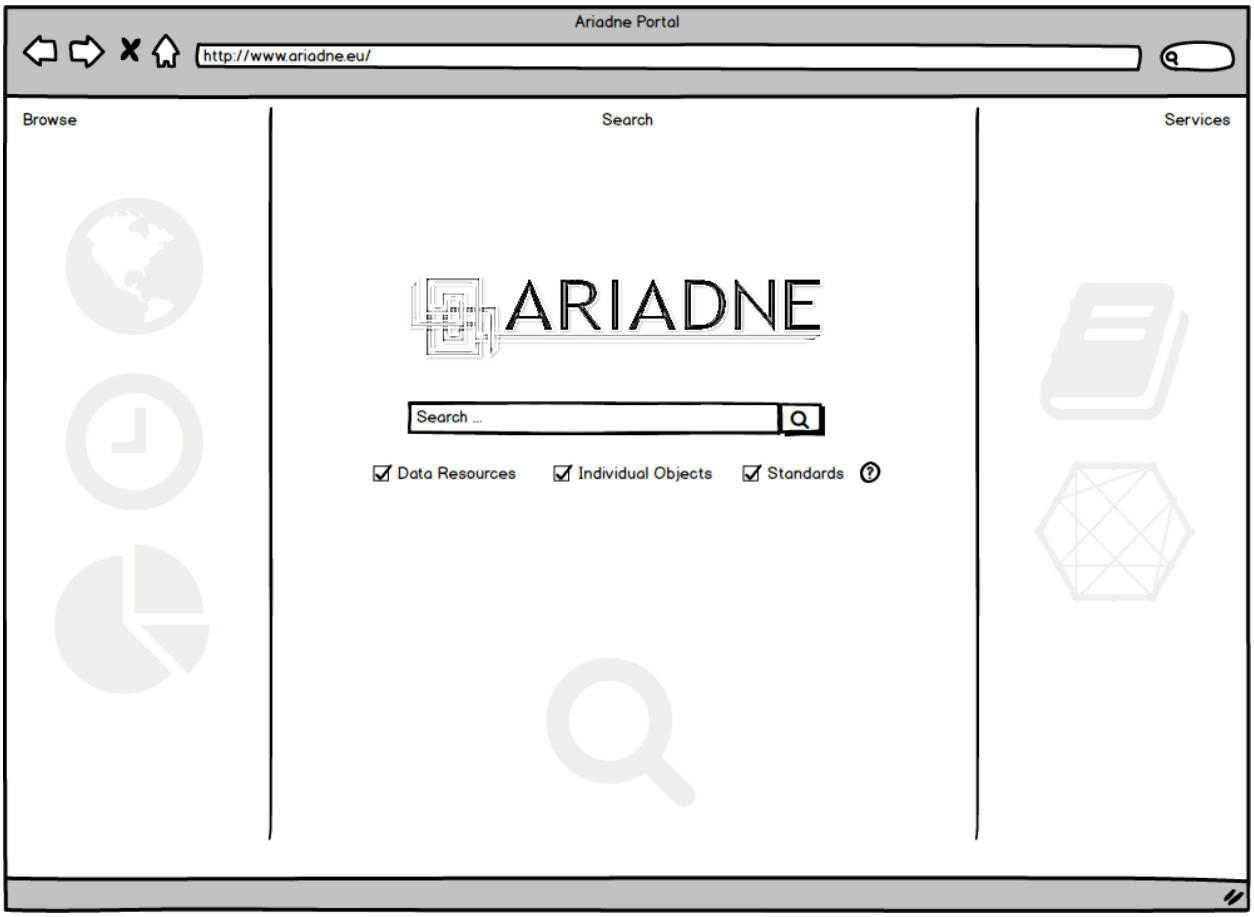


Figure 18: Mockup of the portal's home page.

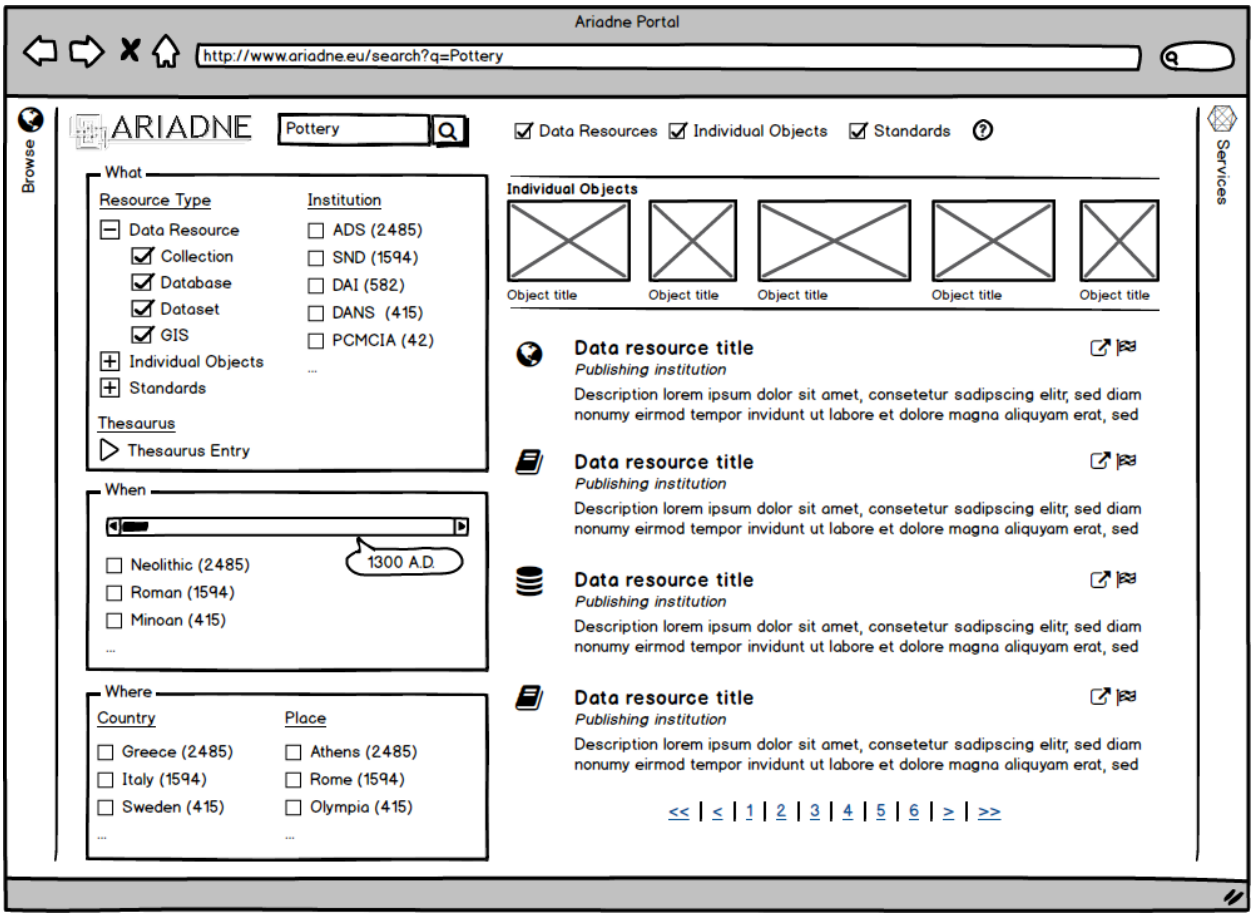


Figure 19: A search results page with faceted browsing/filtering.

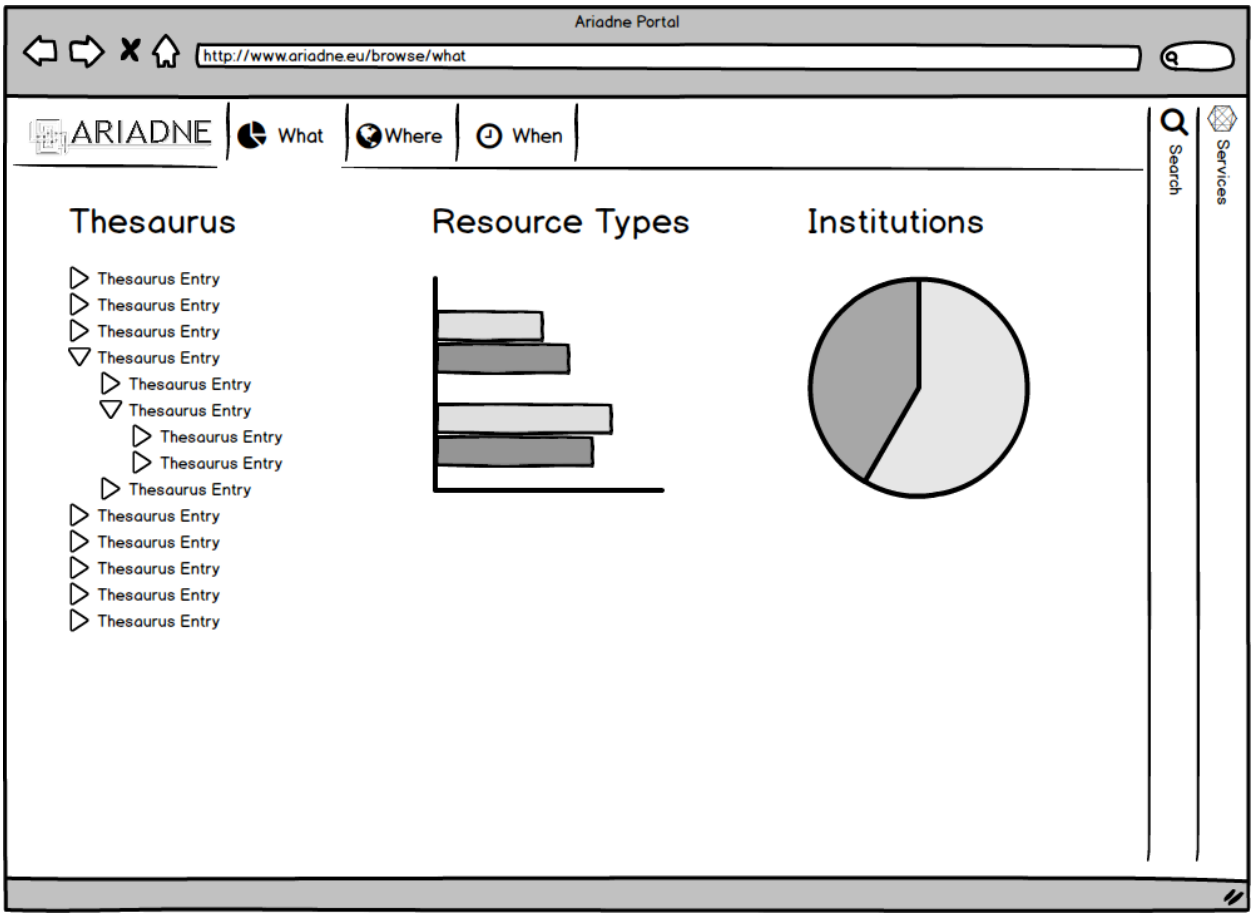


Figure 20: A demo of combining subject, place and time.

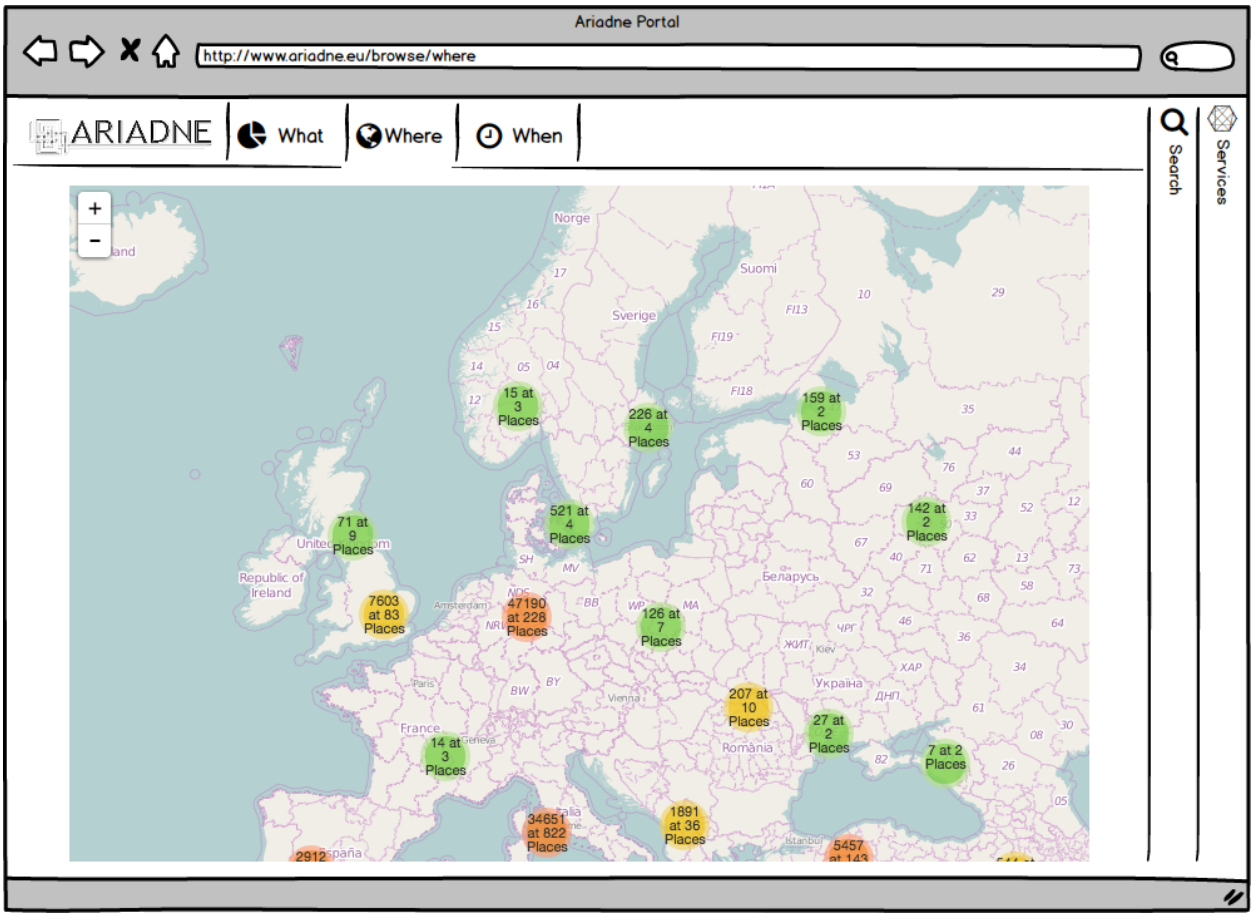


Figure 21. A page showing map-based browsing.

Figure 22: A page showing browsing through a timeline.

Development Framework

Although the ARIADNE portal is designed to be modular, the main framework selected for its implementation is Laravel (http://laravel.com/). Laravel is a PHP based framework designed to facilitate the development of model-view-controller (MVC) applications. Laravel is open source and already provides a good number of services, as can be seen from the table 3.

Initial Prototype

An initial implementation of the ARIADNE portal using the mentioned technologies is shown in the following screenshots (fig. 23 – fig. 25). They present the initial implementation, which has been used by partners to inspect their content and evolved according to the mockups presented in the previous section. The prototype was based on a Bootstrap 3.0 responsive theme and design, which allows use of the portal from virtually any kind of device, including workstations, tablets and smartphones. The prototype was designed using a modular architecture where each module was used to provide a specific

functionality, including searching, browsing and viewing of records, map and thematic views, content providers' facets.

Table 3. Laravel Services

- Authentication
- Billing
- Cache
- Collections
- Command Bus
- Core Extension
- Elixir
- Encryption
- Envoy
- Errors & Logging
- Events
- Filesystem/Cloud Storage
- Hashing
- Helpers
- Localization
- Mail
- Package Development
- Pagination
- Queues
- Session
- Templates
- Unit Testing
- Validation

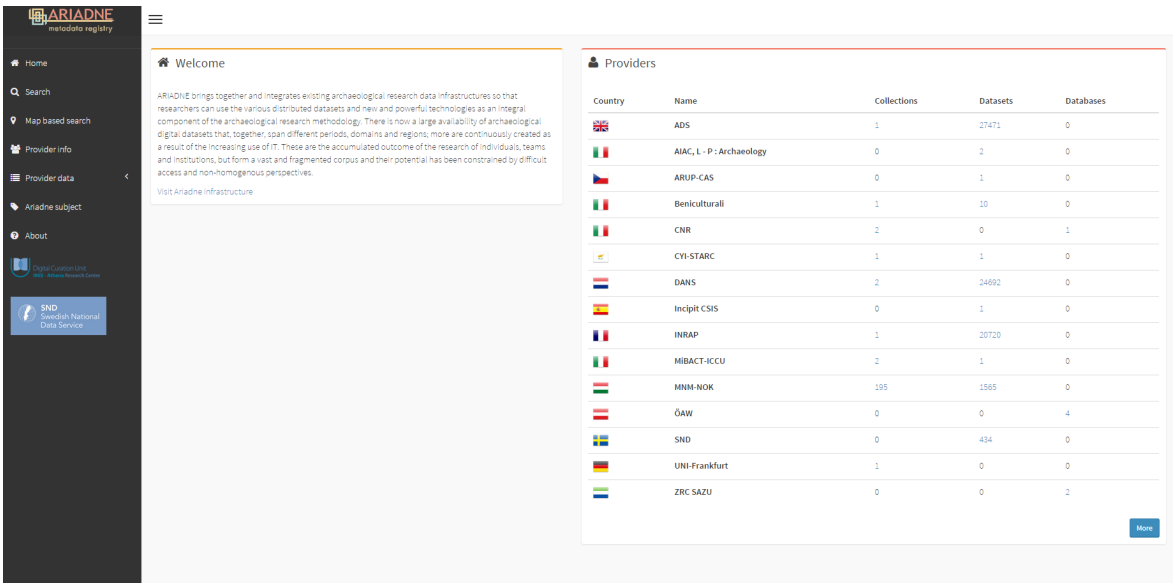


Figure 23: Home screen of the portal.

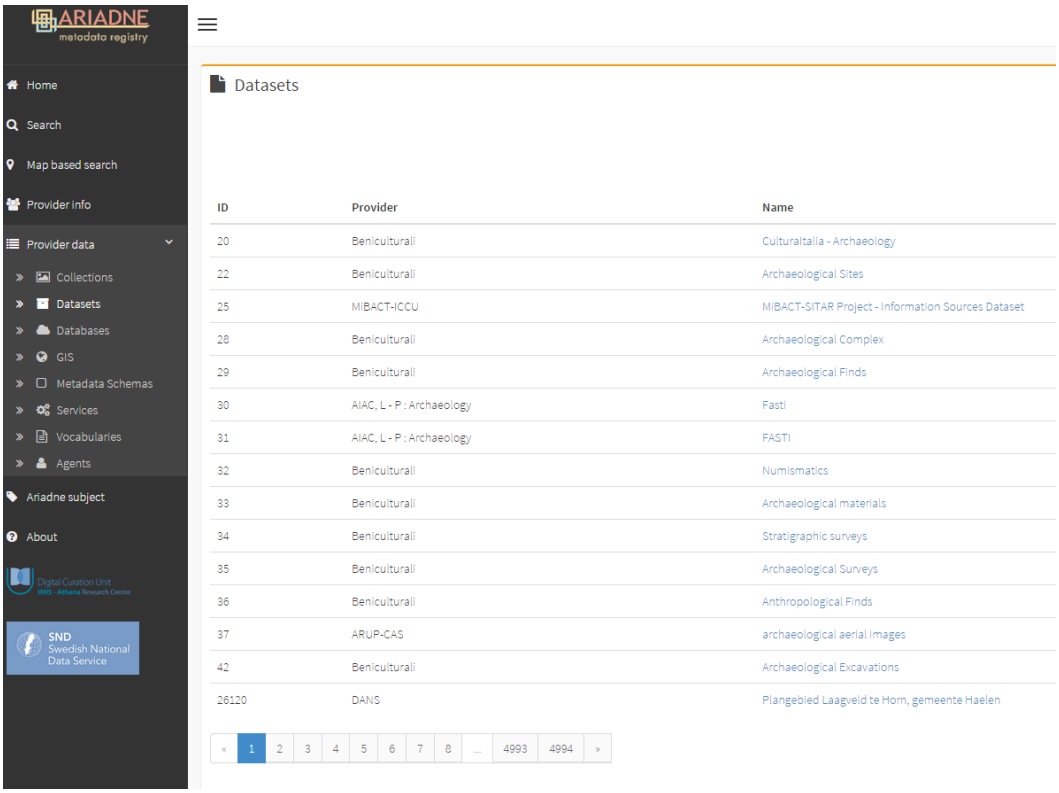


Figure 24: List of existing datasets.

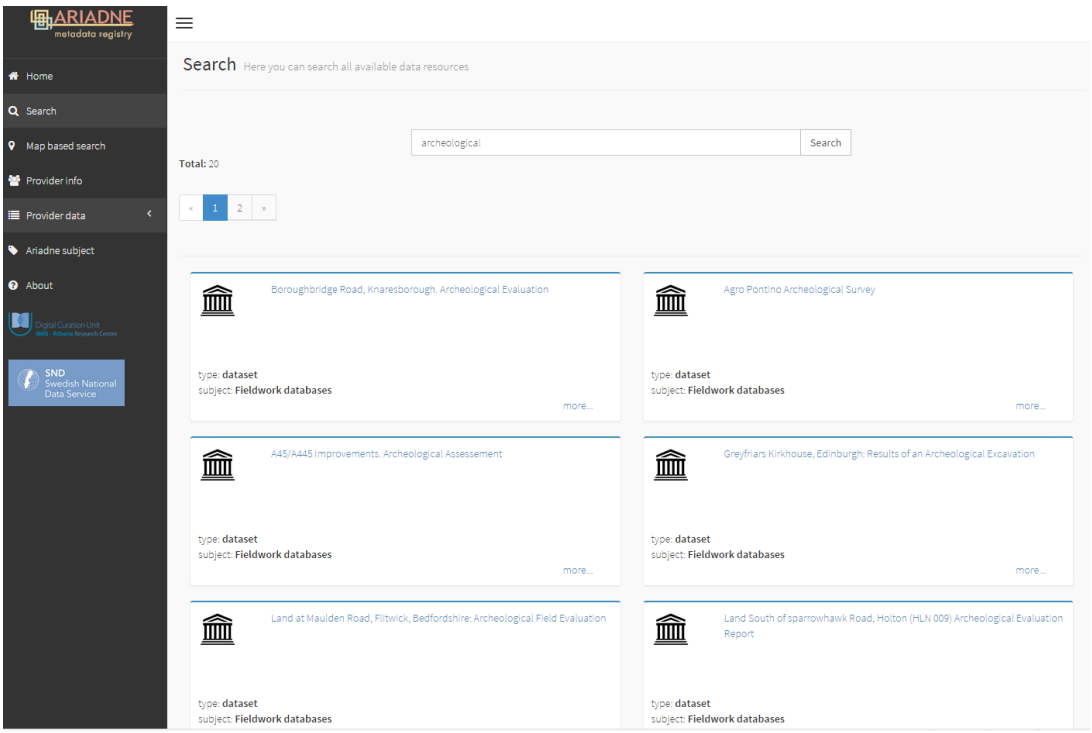


Figure 25: Search results for term ‘archeological’.

ARIADNE Portal Final implementation

The final implementation of the ARIADNE portal (http://portal.ariadne-infrastructure.eu/) was carried out according to the mockups and, as can be seen in fig. 26, it presents the user with the tools to perform a simple search (where he can select one or any field), an AAT based autocomplete search based on a subject, and a search based on space (using a map), time (using a timeline histogram) and subject (using a thematic cloud).

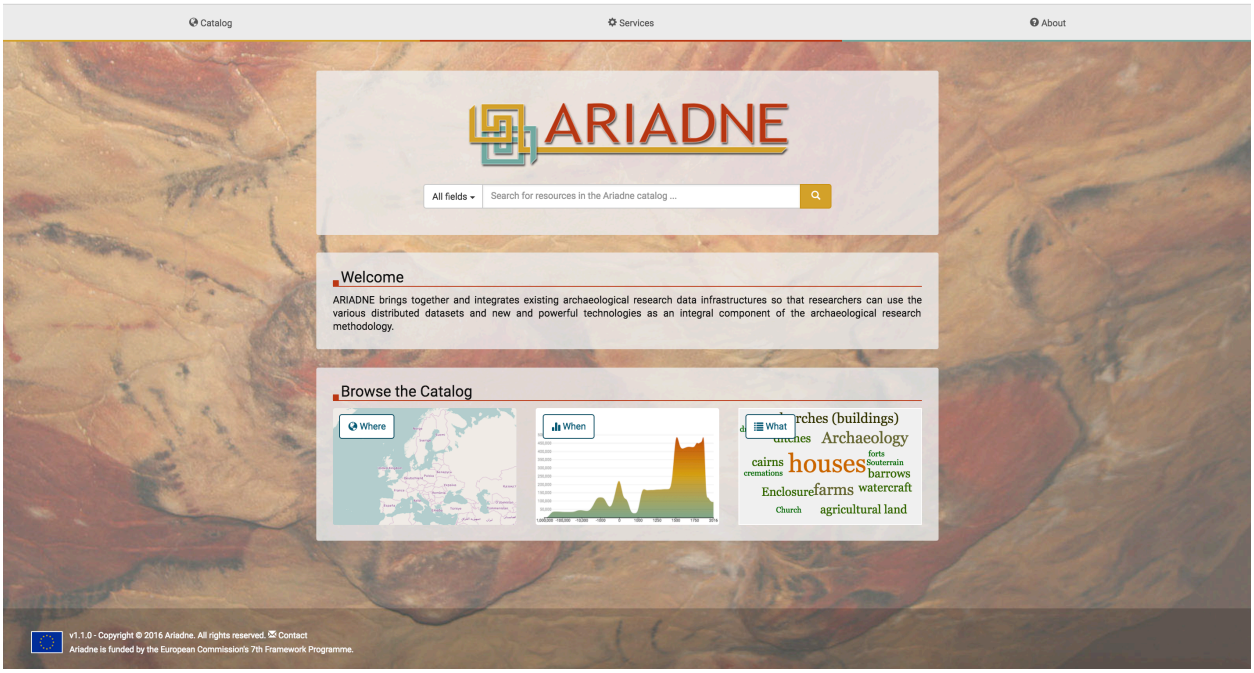


Figure 26: Ariadne portal home page.

After entering a query, the user is presented with a search results page where the matching items are listed on the right (initialy sorted by score, allowing the user to also change the sorting parameter) and a list of facets on the left. The facets include a query refine text box, a heat map where all the search results are depicted on the map, a timeline histogram and a list of boxes with listed attributes that correspond to those of the matching records.

The user can utilize any (or a combination) of these facets to refine his/her search, include the timeline and heatmap.

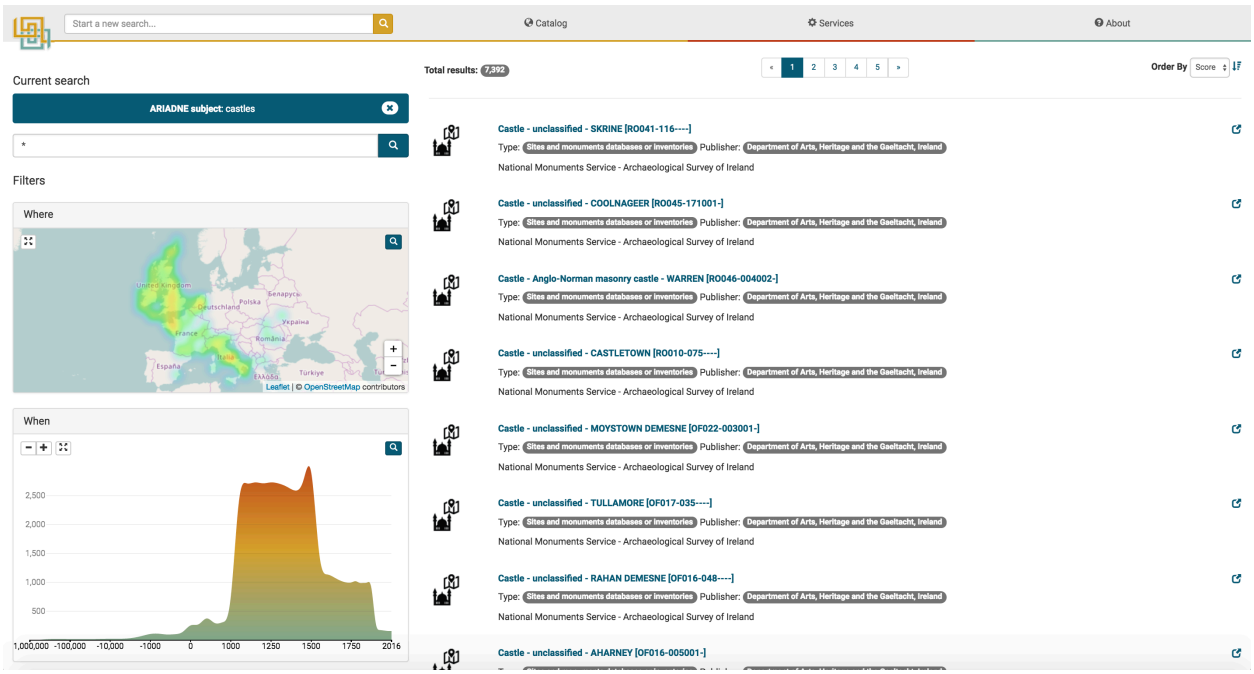


Figure 27: Ariadne portal search results.

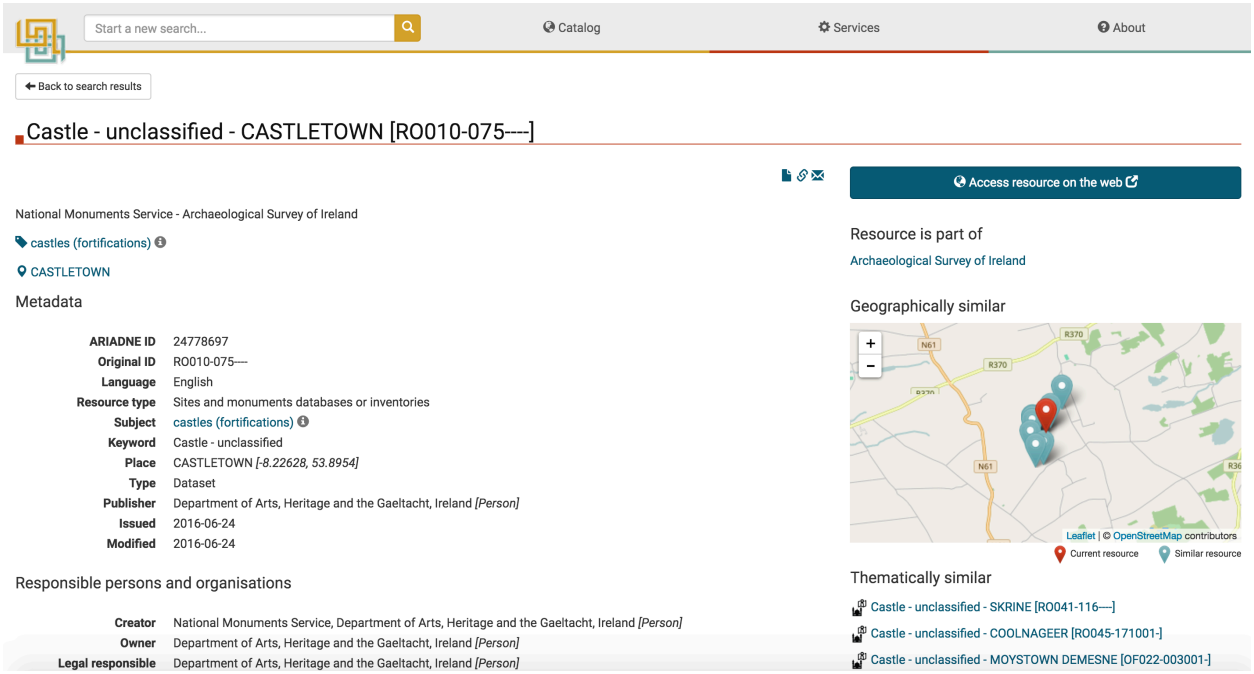


Figure 28: Ariadne portal record view.

Source Code

The source code of the Ariadne Portal (latest version 1.2.0) is hosted in GitHub repository (fig. 29), with open access. It can be found at https://github.com/dainst/ariadne-portal, together with the installation instructions (fig. 30).

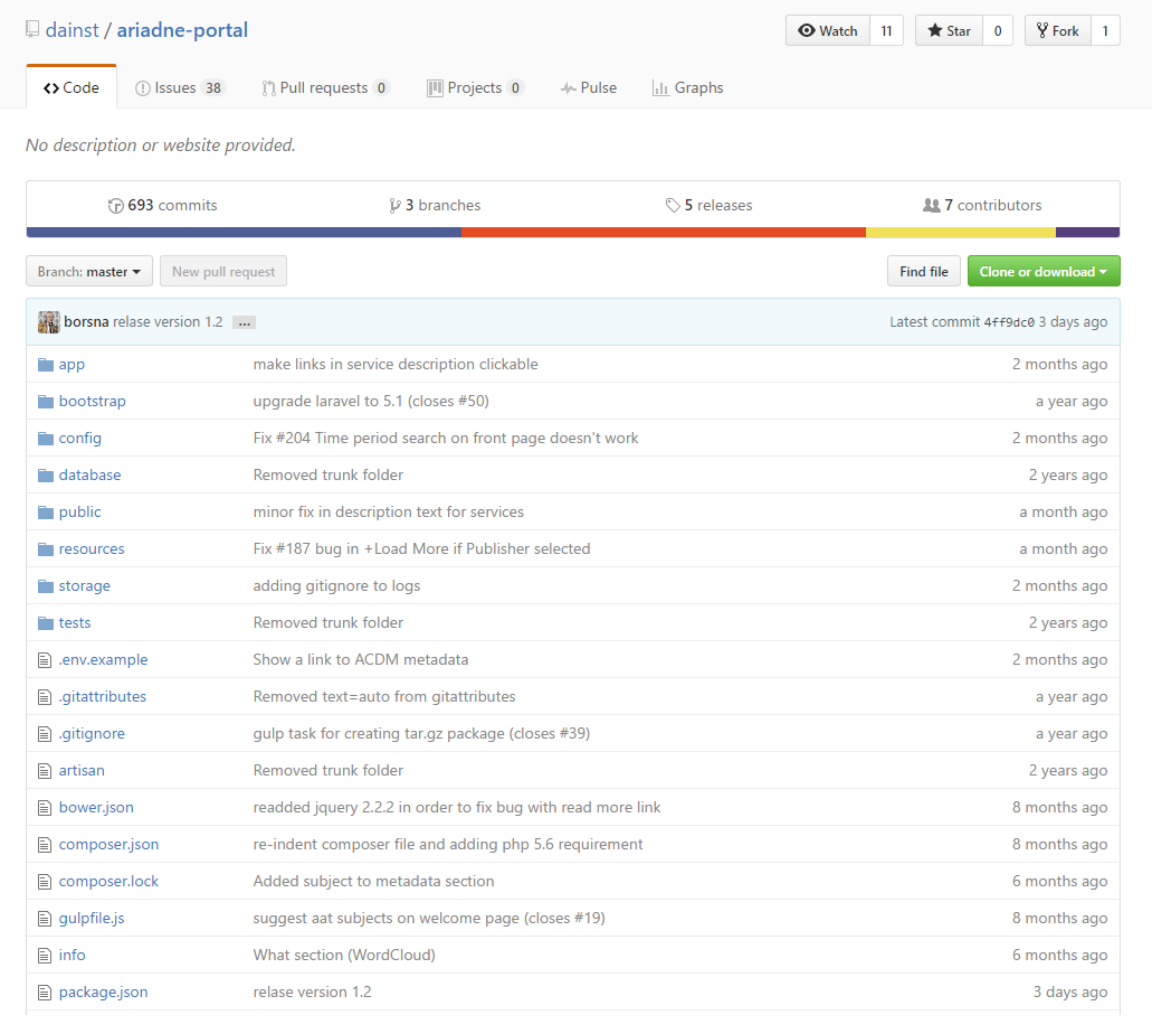
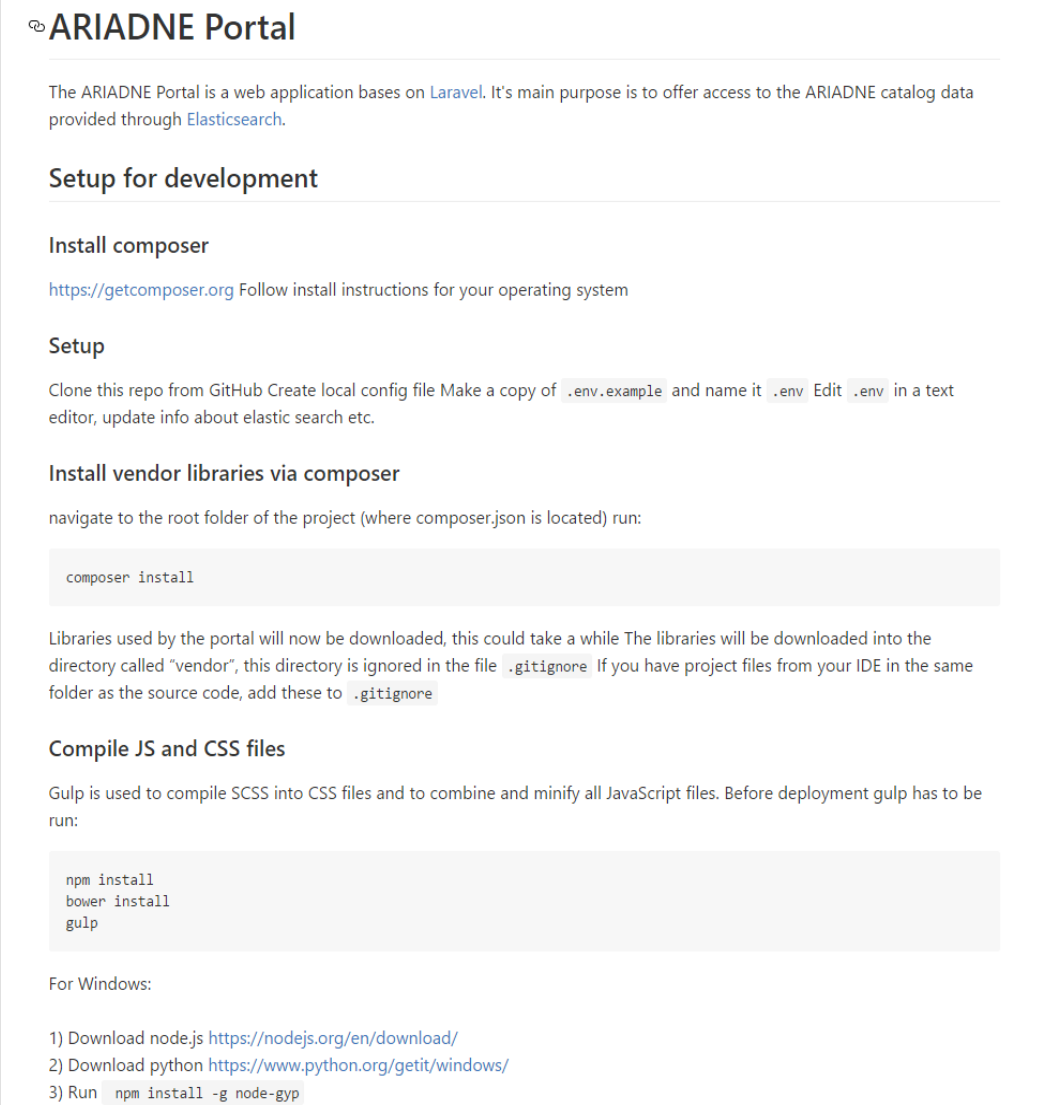


Figure 29: Ariadne Portal source code



ARIADNE Portal

The ARIADNE Portal is a web application bases on Laravel. It's main purpose is to offer access to the ARIADNE catalog data provided through Elasticsearch.

Setup for development

Install composer

https://getcomposer.org Follow install instructions for your operating system

Setup

Clone this repo from GitHub Create local config file Make a copy of `.env.example` and name it `.env` Edit `.env` in a text editor, update info about elastic search etc.

Install vendor libraries via composer

navigate to the root folder of the project (where composer.json is located) run:

```
composer install
```

Libraries used by the portal will now be downloaded, this could take a while The libraries will be downloaded into the directory called "vendor", this directory is ignored in the file `.gitignore` If you have project files from your IDE in the same folder as the source code, add these to `.gitignore`

Compile JS and CSS files

Gulp is used to compile SCSS into CSS files and to combine and minify all JavaScript files. Before deployment gulp has to be run:

```
npm install
bower install
gulp
```

For Windows:

1) Download node.js https://nodejs.org/en/download/
2) Download python https://www.python.org/getit/windows/
3) Run `npm install -g node-gyp`

Figure 30: Installation instructions

GitHub includes an issue tracking system (https://github.com/dainst/ariadne-portal/issues) that manages and maintains lists of reported issues, to create and update features or to resolve bugs. Each user can report an issue, which is then assigned to the person responsible for resolving it (fig. 31).

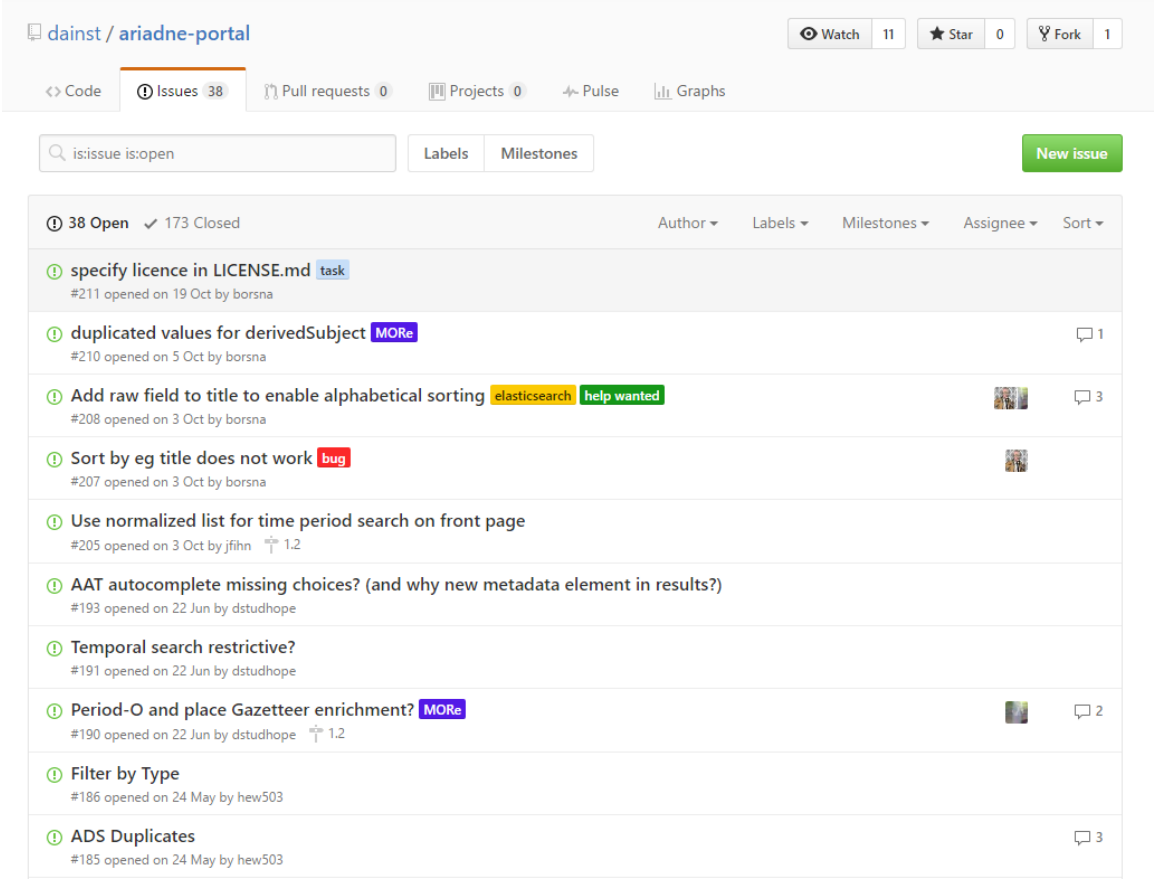


Figure 31: Issue tracking system

10 Support Portal

In order to provide support to the ARIADNE community, a support portal has been created to:

- inform the users and content providers of the infrastructure regarding the ACDM model,
- offer the mapping guidelines
- provide helpdesk
- provide a ticketing support system

An indicative screenshot is presented in fig. 32.

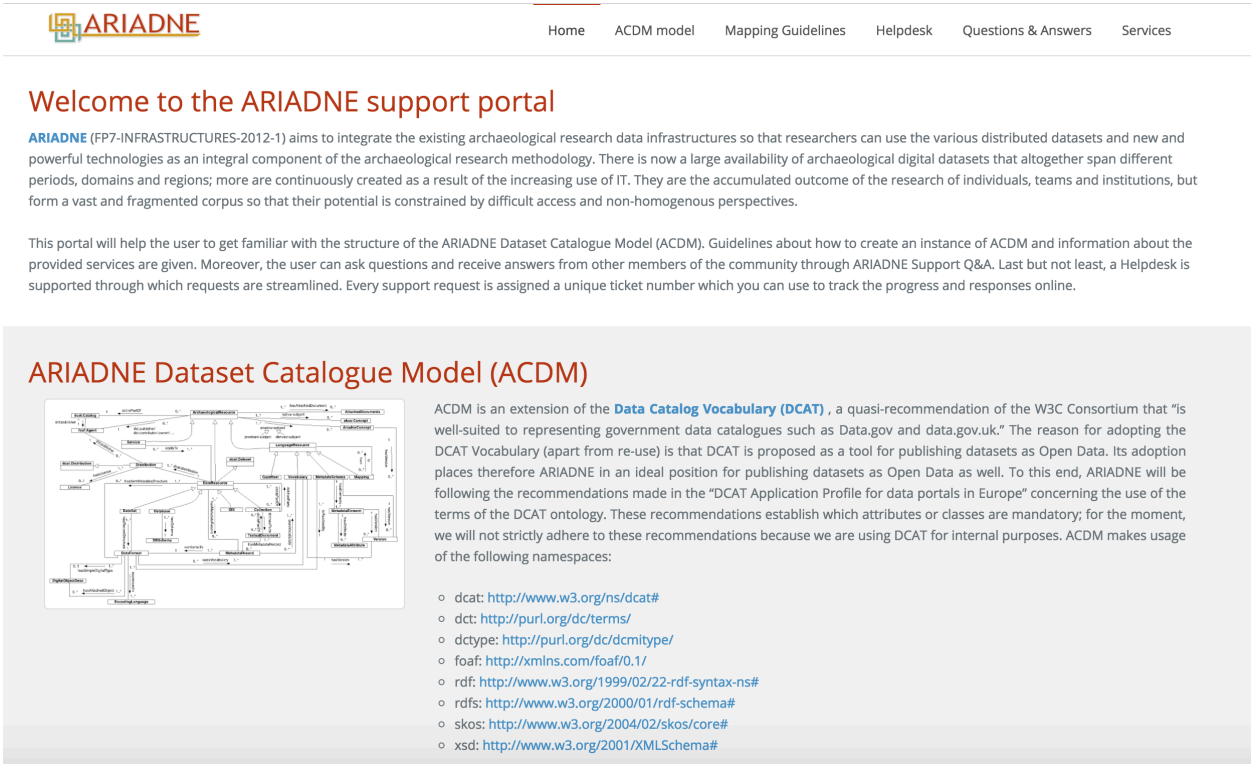


Figure 32: Ariadne support portal homepage

11 Summary of Improvements

The improvements that were made in the period from the initial implementation until the final implementation include:

- Changes in the JSON schema in order to better accomodate the different facets, native and provided subjects.
- Improvements on the search/browse queries in ElasticSearch.
- Improvements in the mapping of the ElasticSearch index to improve facets.
- Improvements in the Portal search and browse functionality.
- Button in each resource and general contact form in portal for reporting data issues.
- Creation of a date normalization service in order to better handle dates and improve the faceted functionality.
- Creation of a spatial normalization service that calculates the midpoint of a bounding box and enriches the resources, thus providing them a geographical representation.
- RDF micro-mapping and export of the Catalog.
- Creation of new and modification of existing import services that import data from excel files because many partners chose their own excel templates in which they encoded their data.
- Modification of import service that imports data from a single xml file to facilitate providers with custom schemas.

12 Conclusions

This document presented the final implementation of the ARIADNE infrastructure project. The technical design choices, as well as the primary components that are required to set up and make the infrastructure operational, have been presented. The infrastructure has aggregated over 2 million records, cleaned, enriched and published them to the ARIADNE portal (http://portal.ariadne-infrastructure.eu/), which is available and used globally.

The diversity of the components required to run the infrastructure, along with the large number of services, are indicative of the complexity of the infrastructure. During the testing of the infrastructure, the various issues were addressed and implemented.

This ARIADNE portal serves as the public face of the infrastructure, and was launched during the CAA 2016 conference on March 29th, 2016. Since then, and up to the November 17th, 2016 (almost 8 months), it has been visited by over 8,700 users, with over 45.000 views from 39 countries (referred to countries with more than 10 visits).

13 References

1. H. Wright, ARIADNE D12.1 Use Requirements, July 2014, http://www.ariadne-infrastructure.eu/Resources/D12.1-Use-Requirements.
2. H. Hollander, M. Hoogerwerf, ARIADNE D.13.1 Service Design, July 2014, http://www.ariadne-infrastructure.eu/Resources/D13.1-Service-Design.
3. P. Ronzino, K. Fernie, C. Papatheodorou, H. Wright, J. Richards, ARIADNE D 3.2 – Report on project standards, November 2013, http://www.ariadne-infrastructure.eu/Resources/D3.2-Report-on-project-standards.
4. C. Papatheodorou, D. Gavrilis,K. Fernie, H. Wright, J. Richards, P. Ronzino, C. Meghini, ARIADNE D 3.1 Initial report on standards and on the project registry, November 2013.
5. N. Aloia, C. Papatheodorou, D. Gavrilis, F. Debole, C. Meghini, Describing Research Data: A Case Study for Archaeology, 13th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2014), Amantea, Italy, October 2014, in R. Meersman et al. (Eds.): "On the Move to Meaningful Internet Systems: OTM 2014 Conferences", Lecture Notes in Computer Science (LNCS) No. 8841: Springer-Verlag, 2014, pp. 768-775.
6. D. Gavrilis, S. Angelis, C. Dallas, A Curation-Oriented Thematic Aggregator, Proceedings of the 17th International Conference on Theory and Practice of Digital Libraries, TPDL 2013, Valetta, Malta, September 2013, pp. 132-137
7. CIDOC-CRM (Conceptual Reference Model), http://www.cidoc-crm.org
8. M. A. Gonçalves, B. L. Moreira, E.A. Fox, L. T. Watson, "What is a good digital library?" - A quality model for digital libraries", Inf. Process. Manage. 43(5): 1416-1437 (2007)
9. Geonames, http://www.geonames.org
10. OAI-PMH - Open Archives Initiative Protocol for Metadata Harvesting, http://www.openarchives.org/pmh/
11. OAI-ORE - Open Archives Initiative, Object Reuse and Exchange Specification, http://www.openarchives.org/ore/
12. METS - Metadata Encoding and Transmission Standard, http://www.loc.gov/standards/mets/
13. DCAT - Data Catalog Vocabulary, http://www.w3.org/TR/vocab-dcat/
14. European Union Open Data Portal, http://open-data.europa.eu
15. RabbitMQ, https://www.rabbitmq.com/
16. GrayLog2, https://www.graylog.org/
17. Elasticsearch, https://www.elastic.co/products/elasticsearch
18. Lucene, http://lucene.apache.org/
19. SolR, http://lucene.apache.org/solr/
20. JSON, http://json.org/

Annex I – MORe API

To use the MORe API, an API key must be first registered through the Developers' section within MORe.

Get Metadata Sources

Get all the Metadata Sources of the MORE aggregator using some filters if it is needed.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/metadataSources
Content Type	application/son
Custom Headers	X-API-Key
Request	
Response	[{"id": 38, "name": "Metadata source 3 -KAMRA", "projectId": 3, "type": "OAI_PMH"}, {"id": 134, "name": "Test - duplications", "projectId": 3, "type": "OAI_PMH"}]
Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Get a specific Metadata Source

Get a spedific Metadata Source of the MORE aggregator given the id of the source.

Method	GET http://more.locloud.eu:8080/more_api/api/v1/metadataSources/{id}
Content Type	application/json
Custom Headers	X-API-Key

Request	
Response	{"id":134,"format":"EDM","name":"Test_duplications","projectId":3, "rootElement":"record","schema":"OAI_DC","set":"ARC3", "type":"OAI_PMH", "uri":"http://more.locloud.eu:8080/locloud/","webUrl":"" }
Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Create a Metadata Source

Create a Metadata Source in the MORE aggregator.

Method	POST http://more.locloud.eu:8080/MoRe_API/api/v1/metadataSources
Content Type	application/json
Custom Headers	X-API-Key
Request	{"format:"EDM", "name":"Test API","projectId":3,"rootElement":"record","schema":"OAI_DC", "set":"ARC1", "type":"OAI_PMH", "uri":"http://api.test.eu","webUrl":""}
Response	{"id":172,"format:"EDM","name":"Sample API MetadataSource","projectId":3,"rootElement":"record","schema":OAI_DC, "set":"ARC3","type":"OAI_PMH", "uri":"http://sample.locloud.eu:8080/locloud/","webUrl":""}
Status	201 : CREATED 500 : INTERNAL_SERVER_ERROR

Get Packages

Get all the packages of the MORE aggregator.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/packages
Content Type	application/json
Custom Headers	X-API-Key
Request	
Response	[{"id": 1144, "schema": "OAI_DC", "sourceId": 133, "status": "Rejected" }, {"id": 1146, "schema": "OAI_DC", "sourceId": 134, "status": "Published"}, {"id": 1149, "schema": "OAI_DC", "sourceId": 134, "status": "Enriched"}]
Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Get a specific Package

Get details for a spedific package of the MORE aggregator, given an id.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}
Content Type	application/json
Custom Headers	X-MoRe-API-Key
Request	

Response	{"id": 1146,"itemsCount":11,"itemsEnriched":0,"itemsIngected":11,"itemsPublished":0,"itemsTransformed":0,"pubDatastreamId":0, "rightsId":0, "schema":"OAI_DC", "sourceId": 134, "status": "Published" }
Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Get package datastreams

Get datastreams for a specific package of the MORE aggregator, given an id.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/datasreams/{schema}
Content Type	application/json
Custom Headers	X-API-Key
Request	
Response	[{"itemId":5322222, "uri":"http://more.locloud.eu/rest_get_item.php?package=1146&schema=OAI_DC&file=5322222.xml"}, {"itemId":5322223, "uri":"http://more.locloud.eu/rest_get_item.php?package=1146&schema=OAI_DC&file=5322223.xml"}]
Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Get package tasks

Get the tasks completed for a specific package of the MORE aggregator, given an id.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/tasks
Content Type	application/json
Custom Headers	X-API-Key
Request	
Response	[{"desc":"Package 498 is ready for ingestion.","timestamp":"2014-10-14 12:49:34.0"}]
Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Get package notifications

Get the notifications created for a specific package of the MORE aggregator, given an id.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/notifications
Content Type	application/json
Custom Headers	X-API-Key
Request	
Response	[{"desc":"A new package [498] has been created.","timestamp":"2014-10-14 12:49:14.0"},

{"desc":"Harvest complete for package 498","timestamp":"2014-10-14 12:49:24.0"},
{"desc":"Validation complete for package 498","timestamp":"2014-10-14 12:49:34.0"}
]

Status 200 : OK
500 : INTERNAL_SERVER_ERROR

Harvest a Package

Harvest a package to the MORE aggregator.

Method POST http://more.locloud.eu:8080/MoRe_API/api/v1/metadataSources/{id}/harvest

Content Type application/json

Custom Headers X-API-Key

Request

Response { "packageId":1299,"schema:"OAI_DC","sourceId":134}

Status 201 : CREATED
500 : INTERNAL_SERVER_ERROR

Ingest a Package

Ingest a package to the MORE aggregator.

Method POST http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/ingest

Content Type application/json

Custom Headers	X-API-Key
Request	
Response	{"msg":"Ingest request received.","packageId": 1299,"schema":"OAI_DC","sourceId":134 }
Status	201 : CREATED 500 : INTERNAL_SERVER_ERROR

Transform a Package

Transform a package of the MORE aggregator.

Method	POST http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/transform
Content Type	application/json
Custom Headers	X-API-Key
Request	{"mappingId":5,"rightsId":1}
Response	{"msg":"Transform request received.","packageId":1299,"mappingId":5,"rightsId":1}
Status	201 : CREATED 500 : INTERNAL_SERVER_ERROR

Enrich a Package

Enrich a package of the MORE aggregator.

Method	POST http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/enrich

Content Type	application/json
Custom Headers	X-API-Key
Request	`{"enrichPlanId":22}`
Response	`{"msg":"Enrich request received.","packageId":1299,"enrichPlanId":22 }`
Status	201 : CREATED 500 : INTERNAL_SERVER_ERROR

Publish a Package

Publish a package of the MORE aggregator.

Method	POST http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/publish
Content Type	application/json
Custom Headers	X-API-Key
Request	`{"schema":"EDM"}`
Response	`{"msg":"Publish request received.","packageId":1299,"schema":"EDM" }`
Status	201 : CREATED 500 : INTERNAL_SERVER_ERROR

Reject a Package

Reject a package of the MORE aggregator.

Method	DELETE http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/reject
Content Type	application/json
Custom Headers	X-API-Key
Request	`{"message": "Reject Message"}`
Response	`{"msg":"Reject request received.","packageId":1299 }`
Status	201 : CREATED 500 : INTERNAL_SERVER_ERROR

Delete a Package

Delete a package from the MORE aggregator.

Method	POST http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/delete
Content Type	application/json
Custom Headers	X-API-Key
Request	
Response	`{"msg":"Delete request received.","packageId":1301 }`
Status	201 : CREATED 500 : INTERNAL_SERVER_ERROR

Withdraw a Package

Withdraw a package of the MORE aggregator.

Method	POST http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/withdraw
Content Type	application/json
Custom Headers	X-API-Key
Request	key
Response	`{"msg":"Withdraw request received.","packageId":1299 }`
Status	201 : CREATED 500 : INTERNAL_SERVER_ERROR

Get mappings

Get the mappings that correspond to a spesific provider.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/packages/{packageId}/mappings
Content Type	application/json
Custom Headers	X-API-Key
Request	

Response	`[`
	`{"id":5,"title":"OAI_DC to EDM"}`
	`]`

Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Get rights

Get the rights appeared in all packages of each provider.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/rights
Content Type	application/json
Custom Headers	X-API-Key
Request	
Response	[{"id":1,"name":"The Public Domain Mark (PDM)"}, {"id":2,"name":"Out of copyright - non commercial re-use (OOC-NC)"}, {"id":3,"name":"The Creative Commons CC0 1.0 Universal Public Domain Dedication (CC0)"}, {"id":4,"name":"Creative Commons - Attribution (BY)"}, {"id":5,"name":"Creative Commons - Attribution, ShareAlike (BY-SA)"}, {"id":6,"name":"Creative Commons - Attribution, No Derivatives (BY-ND)"}, {"id":7,"name":"Creative Commons - Attribution, Non-Commercial (BY-NC)"}, {"id":8,"name":"Creative Commons - Attribution, Non-Commercial, ShareAlike (BY-NC-SA)"}, {"id":9,"name":"Creative Commons - Attribution, Non-Commercial, No Derivatives (BY-NC-ND)"}, {"id":10,"name":"Free access - no re-use"}, {"id":11,"name":"Paid access - no re-use"}, {"id":12,"name":"Orphan work"},]
Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Get enrichment plans

Get the enrichment plans of the provider.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/enrichmentPlans
Content Type	application/json
Custom Headers	X-API-Key
Request	
Response	`[ {"id":1,"schema":"EDM", "title":"Standard EDM Enrichment Plan"} ]`
Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Get projects

Get the projects that a provider belongs to.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/projects
Content Type	application/json
Custom Headers	X-API-Key
Request	
Response	`[ {"acronym":"LoCloud", "projectId":3} ]`

Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Get schemas

Get the available schemas.

Method	GET http://more.locloud.eu:8080/MoRe_API/api/v1/schemas
Content Type	application/json
Custom Headers	X-API-Key
Request	
Response	[{"schema":"CARARE", "schemaId":1,} {"schema":"CARARE20", "schemaId":2,} {"schema":"OAI_DC", "schemaId":3,} {"schema":"EDM", "schemaId":4,} {"schema":"eEDM", "schemaId":5,} {"schema":"ESE", "schemaId":6,} {"schema":"LIDO", "schemaId":7,}]
Status	200 : OK 500 : INTERNAL_SERVER_ERROR

Annex II – Registry API

The first thing the user who wants to use the API must do, is to find his own API key from the Settings. This way he can access only his own records and manipulate them. He will need the key for the API calls.

Get Data Resources

Get all the Data Resources of the registry using some filters if it is needed.

<table>
<tr><td>Method</td><td>GET http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/dataResources</td></tr>
<tr><td>Content Type</td><td>application/json</td></tr>
<tr><td>Request</td><td>
<table>
<tr><th>Parameter</th><th>DataType</th><th>Description</th></tr>
<tr><td>key</td><td>String</td><td>The API key of the user</td></tr>
<tr><td>typeId</td><td>Integer</td><td>The type of the Data Resources (0:Collections 1:Datasets 2:Databases 3:GIS)</td></tr>
</table>
</td></tr>
<tr><td>Response</td><td>
<table>
<tr><th>Parameter</th><th>DataType</th><th>Description</th></tr>
<tr><td>id</td><td>Integer</td><td>The id of each Data Resource</td></tr>
<tr><td>name</td><td>String</td><td>The name of each Data Resource</td></tr>
</table>
</td></tr>
<tr><td>Status</td><td>200 – SUCCESS
500 – INTERNAL_SERVER_ERROR</td></tr>
</table>

Example

Request: {"key":"tUzQwU2T1RqBRuHC1ENJmtaopTMfWF6sHE7OrjRPzr98DiB","typeId":"1"}
Response: [{"id":20,"name":"CulturaItalia - Archaeology"},{"id":33,"name":"Archaeological materials"}]

Get a Data Resource

Get all the properties of a specific Data Resource, given its id.

Method	GET http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/dataResources/{dataResourceId}
Content type	application/json
Request	**Parameter** **DataType** **Description** key String The API key of the user id Integer The id of the specific Data Resource
Response	**Parameter** **DataType** **Description** id Integer The id of the specific Data Resource name String The name of the specific Data Resource properties List The list of the properties for the specific Data Resource
Status	200 – SUCCESS 500 – INTERNAL_SERVER_ERROR

Example

Request: {"key":"tUzQwU2P2T1RqBRHC1ENJmtaopTMfWF63sHE7OrjRPzr98DiBG"}
Response: {"id":41, "name":"dFMROe","properties":{ "dct_issued":"1990s", "dct:accessRights":"access only to 75565 records", ":scientificResponsible":"282", "dcat:keyword":"Roman coins", "ARIADNEDistributionId":"44", "dct_identifier":"dat:41", "dct:subject":"Roman coins", "dct:extent":"140000 records", "dct:audience":"researchers", "ariadne_subject":"Artefacts", "dct_temporal":"116", "dct:creator":"281", "dct_language":"en", "dct:description":"information on coins of the Preroman (Celtic) period and of Roman Imperial Times found in Austria and in Romania", "dct:landingPage":"http://www.oeaw.ac.at/numismatik/fmroe/content/suche.de.php", "dct_spatial":"108", ":dbms":"MySQL" }, "type":2}

Create a new Data Resource

Create a new Data Resource with all its properties. All mandatory elements must have a value for the successful creation.

Method	POST http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/dataResources
Content Type	application/json
Request	**Parameter** / **DataType** / **Description** key / String / The API key of the user name / String / The name of the new Data Resource properties / List / The list of the properties for the new Data Resource type / Integer / The type of the Data Resource (0:Collections 1:Datasets 2:Databases 3:GIS)

Response	**Parameter**	**DataType**	**Description**
	id	Integer	The id of the new Data Resource
	name	String	The name of the new Data Resource
	properties	List	The list of the properties for the new Data Resource
	type	Integer	The type of the Data Resource (0:Collections 1:Datasets 2:Databases 3:GIS)
Status	201 – SUCCESS 400 - BAD_REQUEST 500 – INTERNAL_SERVER_ERROR		

Example

Request: { "name":" API Example Database", "properties":{ "ariadne_subject":"API Example Database ", "dct_language":"en", ":dbms":"MySQL" }, "type":2 }
Response: { "id":50820, "name":"API Example Database", "properties":{ "dct_language":"en", "ariadne_subject":"API Example Database ", ":dbms":"MySQL" }, "type":2 }

Get Distributions

Get all the Distributions of the registry.

Method	GET http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/distributions
Content Type	application/json

Request	**Parameter**	**DataType**	**Description**
	key	String	The API key of the user
Response	**Parameter**	**DataType**	**Description**
	id	Integer	The id of each Distribution
	name	String	The name of each Distribution
Status	200 – SUCCESS 500 – INTERNAL_SERVER_ERROR		

Example

Request: {"key":"tUzQwU2P2T1RqBRHC1ENJmtaopTMfWF63sHE7OrjRPzr98DiBG"}
Response: [{"id":6,"name":"dati.culturaitalia"},{"id":17,"name":" SITAR - Sistema Informativo Territoriale Archeologico di Roma"}]

Get a Distribution

Get all the the properties of a specific Distribution, given its id.

Method	GET http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/distributions/{distributionId}
Content Type	application/json

Request	**Parameter**	**DataType**	**Description**
	key	String	The API key of the user
	id	Integer	The id of the specific Distribution
Response	**Parameter**	**DataType**	**Description**
	id	Integer	The id of the specific Distribution
	name	String	The name of the specific Distribution
	properties	List	The list of the properties for the specific Distribution
Status	200 – SUCCESS 500 – INTERNAL_SERVER_ERROR		

Example

Request: {"key":"tUzQwU2P2T1RqBRHC1ENJmtaopTMfWF63sHE7OrjRPzr98DiBG"}
Response: { "id":45, "name":"DANS Web Portal", "properties":{ "dct_modified":"2014-01-01", "dcat:accessURL":"https://easy.dans.knaw.nl/", "dct_issued":"2014-01-01", "dct:description":"EASY online archiving system", } }

Create a new Distribution

Create a new Distribution with all its properties.

Method	POST http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/distributions

Content Type	application/json
Request	**Parameter** **DataType** **Description** key String The API key of the user name String The name of the new distribution properties List The list of the properties for the new distribution
Response	**Parameter** **DataType** **Description** id Integer The id of the new distribution name String The name of the new distribution properties List The list of the properties for the new distribution
Status	201 – SUCCESS 400 - BAD_REQUEST 500 – INTERNAL_SERVER_ERROR

Example

Request: {"key":"tUzQwU2P2T1RqBRHC1ENJmtaopTMfWF63sHE7OrjRPzr98DiBG", "name":"DANS Web Portal", "properties":{ "dct_modified":"2014-01-01", "dcat:accessURL":"https://easy.dans.knaw.nl/", "dct_issued":"2014-01-01", "dct:description":"EASY online archiving system", "dct:publisher":"5", "dct:licence":"8" } }
Response: { "id":45, "name":"DANS Web Portal", "properties":{ "dct_modified":"2014-01-01", "dcat:accessURL":"https://easy.dans.knaw.nl/", "dct_issued":"2014-01-01", "dct:description":"EASY online archiving system", "dct:publisher":"5", "dct:licence":"8" } }

Get Data Formats

Get all the Data Formats of the registry.

Method	GET http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/dataFormats
Content Type	application/json
Request	**Parameter** **DataType** **Description** key String The API key of the user
Response	**Parameter** **DataType** **Description** id Integer The id of each Data Format name String The name of each Data Format
Status	200 – SUCCESS 500 – INTERNAL_SERVER_ERROR

Example

Request: {"key":"tUzQwU2P2T1RqBRHC1ENJmtaopTMfWF63sHE7OrjRPzr98DiBG"}
Response: [{"id":149,"name":"aerial image"},{"id":146,"name":" Information Source Data Format"}]

Get a Data Format

Get all the the properties of a specific Data Format, given its id.

Method	GET http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/dataFormats/{dataFormatId}
Content Type	application/json
Request	**Parameter** **DataType** **Description** key String The API key of the user id Integer The id of the specific Data Format
Response	**Parameter** **DataType** **Description** id Integer The id of the specific Data Format name String The name of the specific Data Format properties List The list of the properties for the specific Data Format
Status	200 – SUCCESS 500 – INTERNAL_SERVER_ERROR

Example

Request: {"key":"tUzQwU2P2T1RqBRHC1ENJmtaopTMfWF63sHE7OrjRPzr98DiBG"}
Response: { "id":149, "name":"aerial image", "properties":{ "dct:description":"JPG image", } }

Create a new Data Format

Create a new Data Format with all its properties.

Method	POST http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/dataFormats
Content Type	application/json
Request	**Parameter** **DataType** **Description** key String The API key of the user name String The name of the new data Format properties List The list of the properties for the new data Format
Response	**Parameter** **DataType** **Description** id Integer The id of the new data Format name String The name of the new data Format properties List The list of the properties for the new data Format
Status	201 – SUCCESS 400 - BAD_REQUEST 500 – INTERNAL_SERVER_ERROR

Example

Request: {"key":"tUzQwU2P2T1RqBRHC1ENJmtaopTMfWF63sHE7OrjRPzr98DiBG", "name":"aerial image", "properties":{ "expressedIn":"xml", "dct:description":"JPG image", ":characterSet":"utf-8" } }
Response: { "id":149, "name":"aerial image", "properties":{ "expressedIn":"xml", "dct:description":"JPG image", ":characterSet":"utf-8" } }

Get foaf:Agents

Get all the foaf:Agent of the registry.

Method	GET http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/foafAgents
Content Type	application/json
Request	**Parameter** / **DataType** / **Description** key / String / The API key of the user typeId / Integer / The type of the foaf:Agent (0:Agent 1:Organization)
Response	**Parameter** / **DataType** / **Description** id / Integer / The id of each foaf:Agent name / String / The name of each foaf:Agent
Status	200 – SUCCESS 500 – INTERNAL_SERVER_ERROR

Example

Request: {"key":"tUzQwU2P2T1RqBRHC1ENJmtaopTMfWF63sHE7OrjRPzr98DiBG","typeId":"1"}
Response: [{"id":270,"name":"Emerenziana Usai"},{"id":259,"name":"Mateja Belak"}]

Get a foaf:Agent

Get all the the properties of a specific Agent, given its id.

Method	GET http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/foafAgents/{foafAgentId}
Content Type	application/json
Request	**Parameter** / **DataType** / **Description** key / String / The API key of the user id / Integer / The id of the specific foaf:Agent
Response	**Parameter** / **DataType** / **Description** id / Integer / The id of the specific foaf:Agent name / String / The name of the specific foaf:Agent properties / List / The list of the properties for the specific foaf:Agent
Status	200 – SUCCESS 500 – INTERNAL_SERVER_ERROR

Example

Request: {"key":"tUzQwU2P2T1RqBRHC1ENJmtaopTMfWF63sHE7OrjRPzr98DiBG"}
Response: { "id":110, "name":"Data Archiving and Networked Services (DANS)", "properties":{ "foaf_mbox":"info@dans.knaw.nl", }, "type":1 }

Create a new foaf:Agent

Create a new Agent with all its properties.

Method	POST http://ariadne-registry.dcu.gr:8080/ariadneWeb/api/v1/foafAgents
Content Type	application/json
Request	**Parameter** / **DataType** / **Description** key / String / The API key of the user name / String / The name of the new foaf:Agent properties / List / The list of the properties for the new foaf:Agent type / Integer / The type of the foaf:Agent (0:Agent 1:Organization)
Response	**Parameter** / **DataType** / **Description** id / Integer / The id of the new foaf:Agent name / String / The name of the new foaf:Agent

	properties	List	The list of the properties for the new foaf:Agent
	type	Integer	The type of the foaf:Agens (0:Agent 1:Organization)
Status	201 – SUCCESS 400 - BAD_REQUEST 500 – INTERNAL_SERVER_ERROR		

Example

Request: {"key":"tUzQwU2P2T1RqBRHC1ENJmtaopTMfWF63sHE7OrjRPzr98DiBG", "name":"Data Archiving and Networked Services (DANS)", "properties":{ "foaf_mbox":"info@dans.knaw.nl", "foaf:phone":"888999214", "foaf:homepage":"www.example.com", "foaf:skypeID":"dans" }, "type":1 }

Response: { "id":110, "name":"Data Archiving and Networked Services (DANS)", "properties":{ "foaf_mbox":"info@dans.knaw.nl", "foaf:phone":"888999214", "foaf:homepage":"www.example.com", "foaf:skypeID":"dans" }, "type":1 }

Annex III –Thesauri Available through MORe

This table presents some (of the most relevant) thesauri that MORe is able to access through the TemaTres API. These thesauri are accessible to the end user through the subject collections enrichment micro-services.

Author	Name	Subject	Language
University of California, Santa Barbara	Alexandria Digital Library Feature Type Thesaurus	Science & Technology: places, geographica	English
Royal Commission on the Ancient and Historical Monuments of Scotland (RCAHMS)	Archeological Objects Thesaurus Scotland	Objects made by human activity	English
English Heritage	Archeological Sciences Thesaurus	Techniques, recovery methods and materials associated with archaeological sciences	English
English Heritage	Building Materials Thesaurus	Main constructional material types (eg. the walls) for indexing of monuments	English
English Heritage	Components Thesaurus	Terminology covering divisions and structural elements of a building or monument	English
American Folklore Society	Ethnographic Thesaurus	Social sciences: ethnographic, folklore, united states, ethnomusicology	English
English Heritage	Event Type Thesaurus	Terminology used for recording archaeological and architectural investigative, data collection exercises; from intrusive interventions to non damaging surveys	English

Author	Name	Subject	Language
English Heritage	Evidence Thesaurus	Terminology covering the existing physical remains of a monument, or the means by which a monument has been identified where no physical remains exist	English
English Heritage	FISH Archeological Objects Thesaurus	Recording of archaeological objects in Britain and Ireland covering all historical periods	English
GEMET Thesaurus	General Multilingual Environmental Thesaurus GEMET	Environment, policies	Dutch, English, French, German, Italian, Portuguese, Spanish
Federation Internationale des Archives du Film (FIAF)	General Subject headings for Film Archives	Arts & Humanities: films, films archives, cinema	English
The Discovery Programme	Irish Monuments	Irish monuments	English
The Discovery Programme	Irish Periods	Irish periods	English
Royal Commission on the Ancient and Historical Monuments of Scotland (RCAHMS)	Maritime Craft Thesaurus Scotland	Types of craft that survive as wrecks, or are documented as losses, in Scottish maritime waters	English
English Heritage	Maritime Craft Type Thesaurus	Craft types which survive as wrecks in English Heritage's maritime record	English
English Heritage and Royal Commission on the Historical Monuments of England	MDA Archaeological Objects Thesaurus	Social sciences: archaeological objects; museums; vocabulary	English

Author	Name	Subject	Language
Royal Commission on the Ancient and Historical Monuments of Wales (RCAHMW)	Monument Thesaurus Wales	Classification of monument types in Wales by function	English
Royal Commission on the Ancient and Historical Monuments of Scotland (RCAHMS)	Monument Type Thesaurus	Monument types relating to the archaeological and built heritage of Scotland	English, Scottish Gaelic for some terms
English Heritage	Period Thesaurus	English Heritage Periods List	English
Royal Commission on the Ancient and Historical Monuments of Wales (RCAHMW)	Period Thesaurus Wales	A list of periods for use in Wales	English
Bibliographic Standards Committee of the Rare Books and Manuscripts Section (ACRL/ALA)	Relator Terms for Use in Rare Book and Special Collections Cataloguing	Arts & Humanities: Rare Book; Special Collection	English
Universidad de León	Tesauro de Ciencias de la Documentación	Social sciences: library sciences, information sciences, documentation, librarianship	Spanish
Library of Congress. Prints and Photographs Division	Thesaurus for Graphic Materials 1: Subject Terms	Arts & Humanities: graphic materials, subject terms	English
Library of Congress. Prints and Photographs Division	Thesaurus for Graphic Materials 2: Genre and Physical Characteristic Terms	Arts & Humanities: graphic materials	English
Ministero per i Beni e le Attività Culturali	Thesaurus PICO 4.1	Arts & Humanities: italian culture, heritage	English, Italien

Author	Name	Subject	Language
UKAT	UK Archival Thesaurus (UKAT)	General reference: archives, ukat, united kingdom	English
UNESCO	UNESCO thesaurus	General reference: education, culture, social and human sciences, information and communication, politics, law, economic science	English, French, Spanish