DELIVERABLE

Project Acronym:	**Europeana Cloud**
Grant Agreement number:	**325091**
Project Title:	**Europeana Cloud: Unlocking Europe's Research via The Cloud**

D2.2: Europeana Cloud Architectural Design

Revision: 1.0

Authors:

Marcin Werla (PSNC)
Georgios Mamakis (Europeana)
Markus Muhr (The European Library)
Petr Knoth (Open University)
Pavel Kats (Europeana)

Project co-funded by the European Commission within the ICT Policy Support Programme		
Dissemination Level		
P	Public	X
C	Confidential, only for members of the consortium and the Commission Services	

Revision History

Revision	Date	Author	Organisation	Description
1.0	31.07.2013	Marcin Werla	PSNC	First version of the architecture finalized at M6 of the project.
		Georgios Mamakis	Europeana	
		Markus Muhr	The European Library	
		Petr Knoth	Open University	
		Pavel Kats	Europeana	

D2.2: Europeana Cloud Architectural Design

Executive summary

Introduction

Existing ICT tools and infrastructures are not sufficient to serve the vision of European integration in the cultural domain. They are poorly orchestrated, they usually support only unidirectional flow of information, often employ different technologies and standards, and, last but not the least, are very costly. Additionally, they are mostly oriented towards operating metadata, leaving the need of providers for efficient content storage and access solutions unanswered. Also humanities scholars still face an immense amount of dispersed resources and resort to manual methods in order to reach to these resources and use them in research.

Europeana Cloud[1] is a new project funded by the European Union scoped to address the above issues. It is coordinated by Europeana Foundation and has a vision of creating new digital infrastructure for cultural content that will be used by Europeana and other entities from all over Europe, interested in sharing or reusing digital representations of cultural resources. This infrastructure aim is to provide new abilities for efficiently storing metadata and content, easily sharing cultural assets between institutions, improving abilities to access these assets and research them using innovative tools.

Who will use Europeana Cloud system and for what purpose?

Europeana Cloud system is intended for entities which are interested in storing, distributing and re-using cultural data: digital files representing cultural objects as well as their descriptions (called metadata). These entities include cultural heritage institutions, data and metadata aggregators, scholars and creative industry companies.

The initial purpose of the system, as defined by user stories gathered from metadata aggregators, is the following:

- To provide globally unique identifiers for cultural data records from diverse sources
- To provide storage and access capabilities for cultural data records, consisting of data and metadata streams in many formats and versions
- To provide annotation capabilities for cultural data records
- To provide cultural data records changes tracking capability
- To provide flexible, scalable and customizable cultural data processing capabilities

All the above should be done in a secure, reliable and scalable way, allowing to use the Europeana Cloud system as the underlying infrastructure for cultural applications and information systems – the backbone of digital ecosystem for cultural data.

[1] Europeana Cloud project website: http://pro.europeana.eu/web/europeana-cloud

How Europeana Cloud system will look like?

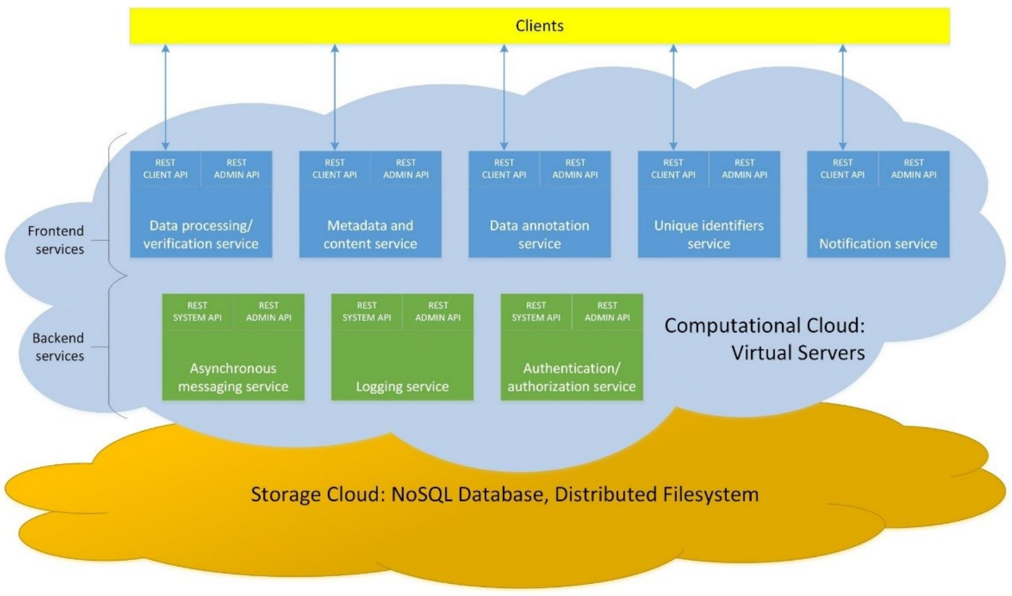
Europeana Cloud system will be a service-oriented infrastructure, consisting of one or more instances of a number of network services. Each service will be responsible for providing a particular set of functionality, giving together the full set of desired eCloud features. The Europeana Cloud Architectural Design document defines the following frontend functional services:

- Unique Identifier Service – provides the mechanism to create mappings between local identifier (scoped with the data provider) and the global identifiers inside the eCloud system scope.
- Metadata and Content Service – provides the create/read/update/delete operations for cultural data records in multiple representations and versions.
- Notification Service – provides the communication mechanism between the internal services and external clients for notifications about data changes in the eCloud system.
- Data Annotation Service – allows to store and access any additional information (annotation) related to any data record or its components (e.g. versions).
- Data Processing Service – offers the possibility to process data in a three stages workflow: extracting the data for processing from the eCloud system, processing the data, loading the processing outcomes to specified output location.

Beside the above functional services, the eCloud system will also include three backend system services: Authentication/Authorization Service, Logging Service and Asynchronous Messaging Service.

Where Europeana Cloud system will be deployed?

The eCloud system, which from the end user point of view will look like SaaS cloud (or IaaS in case of cultural data records storage), should be also deployed in cloud environment, in order to be reliable, available, scalable and cost effective. In order to achieve that, two types of underlying cloud will be needed: storage cloud (distributed database and file system) and computational cloud (virtual machines to deploy eCloud system services).


Clients
Frontend services
Backend services
REST CLIENT API
REST ADMIN API
Data processing/ verification service
Metadata and content service
Data annotation service
Unique identifiers service
Notification service
REST SYSTEM API
Asynchronous messaging service
Logging service
Authentication/ authorization service
Computational Cloud: Virtual Servers
Storage Cloud: NoSQL Database, Distributed Filesystem

These two types of underlying cloud will be constructed with the hybrid cloud approach. They will consist of a private, community-based part where the necessary hardware resources will be provided by voluntary, technically advanced institutional users of the eCloud system and a public part, based on resources leased from commercial providers. Such approach should allow to provide cost-effective service independent from any commercial provider, yet scalable to commercial resources if needed.

Table of Contents

List of figures

Glossary

This section of the document contains explanation of selected key terms used in this report. The aim of the glossary is to provide common understanding of such key terms, in order to minimize possible misunderstandings between creators and readers of this document.

If an explanation of a term contains other glossary term, this term is in [square brackets]. Relations between key glossary terms are shown on a Figure 1 below the glossary table.

Term	Explanation
aggregator	A [service] which main aim is to gather data from various [data providers]. For example Europeana is an aggregator, which collects data about digital representations of cultural heritage objects from portals all over the Europe. Aggregator can also be the data provider, if the aggregated data is properly exposed for other services.
client	A piece of software running on a computer and accessing/using a [service]. Client can be autonomous or operated by human. For example, if the service is a website, the client can be a web browser (operated by human [user]), but also a search engine indexing robot operating autonomously.
content	A digital representation of physical object (e.g. digitized book) or eventually a born-digital object. Content can be described with [metadata].
data provider	A [service] which is providing [data sets] to the [aggregator]. Data provider can be also an aggregator, offering data sets previously aggregated from other data providers.
data record	Unit of data transferred among [data providers], [aggregators] and [clients]. Can contain [content] and/or [metadata] in many different representations (e.g. different formats).
data set	A collection of [data records].
metadata	The descriptive information about [content]. Can be expressed in many different formats.
service	A piece of software running on a computer connected to the internet (a server), offering set of functionality to its [clients]. Service can offer wide set of functionality (e.g. Europeana portal is a service) or very narrow one (e.g. file compression service).
user	A human operator using the software [client].

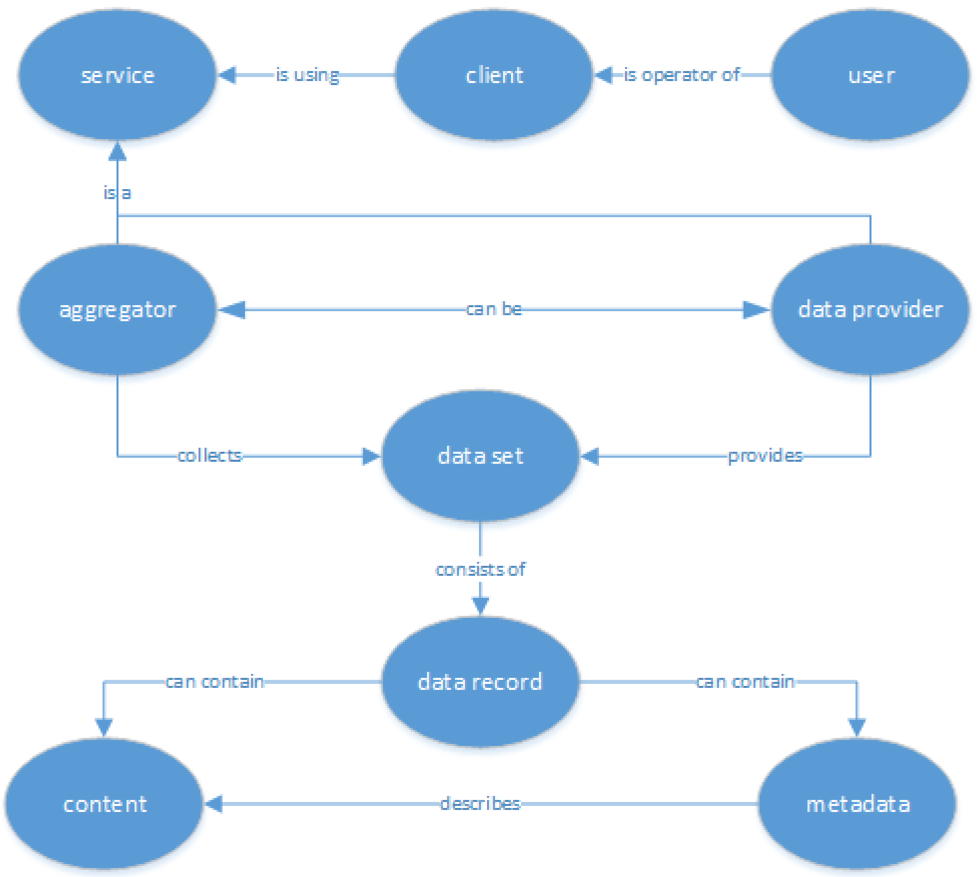


Figure 1. Relations between key glossary terms.

1 Introduction

1.1 Europeana Cloud Background

Europeana Cloud is a new strategic project funded by the European Union and coordinated by the Europeana Foundation. It has a vision of creating a new digital infrastructure for cultural content that will be used by Europeana and its partners (initially focusing on metadata aggregators). This infrastructure will exploit latest technological advances in the domain of cloud computing to provide new abilities for efficiently storing metadata and content, easily sharing cultural assets between institutions, improving abilities to access these assets and research them using innovative tools.

A need for new infrastructures for maintaining, sharing, and researching European cultural content has long been recognized by institutions, public and policy makers. Existing tools and infrastructures are not sufficient to serve the vision of European integration in the cultural domain. Existing aggregation infrastructures are poorly orchestrated, they usually support unidirectional flow of information, often employ different technologies and standards, and, last but not the least, are very costly. Additionally, they are mostly oriented towards operating metadata, leaving the need of providers for efficient content storage solutions unanswered. Finally, although aggregation infrastructures have shown an impressive progress over the last years, the access and research needs of primary consumers of the cultural content - humanity and social sciences scholars - are not yet satisfied by these infrastructures. Despite significant technological progress scholars still face an immense amount of dispersed resources and resort to manual methods in order to reach to these resources and use them in research.

As it often happens, the right technology ripens when the need for it becomes acute. In recent years cloud computing has become a dominating theme in building new cost efficient IT infrastructures. A growing number of software services and components building on the principles of cloud computing are being developed both as commercial products and as open-source software. Some open-source solutions have matured to a level where they can be safely used as of-the-shelf solutions for building cloud-based services by enterprises. It is not a coincidence, thus, that cloud computing has been chosen as the central technological motive of the new European infrastructure for cultural institutions.

The partners in the Europeana Cloud consortium are well aware of the importance of addressing in parallel several crucial aspects from the early stage of the project. These aspects are: investigating primary needs of data providers and researchers from the new system, developing its sound technical foundation, customizing and developing versatile research tools on top of this infrastructure, investigating various economical and legal aspects related to its sustainability after the end of the project, and sourcing content. The work packages in the project have been aligned to these aspects.

More information on the project and its division across responsibilities and work groups can be found in the project Description of Work[2] and on the project website[3].

The current document is prepared by WP2 of the Europeana Cloud project, which is responsible for building the technological infrastructure. The document is the first major deliverable of the work package and its aim is to outline the process of conceptualization of the architecture of the platform and the high-level description of the proposed architecture.

[2] http://pro.europeana.eu/documents/1414567/0/Europeana+Cloud+-+Description+of+Work

[3] http://pro.europeana.eu/web/europeana-cloud

1.2 Methodology for Specification of the Europeana Cloud Architecture

As mentioned above, this document contains the specification of the Europeana Cloud Architecture. It contains not only the description of the architecture itself, but it also introduces all information which was gathered during the process of defining the architecture and which justifies design decisions.

To have a strong ground for the new system, it was necessary to gather and analyse the needs which the system should fulfil and support them with benefits which the system will provide for its users. The initial set of needs and benefits was described in the form of 32 user stories and was collected from the following sources:

- Metadata aggregators:
    - Europeana
    - The Europeana Library
    - Polish Digital Libraries Federation
- Other institutions interested in using the system and participating in the working session of the project kick-off meeting.

These user stories were summarized into a coherent set of common stories, which was later on prioritized by the representatives of these three aggregators.

This resulted in a final set of 14 stories, which were after further analysis transformed into 9 functional requirements, stating what the system should do. This functional requirements were also extended by a set of non-functional requirements providing additional information on how the system should provide the desired functionality, including aspects like reliability, security or availability.

The design of the Europeana Cloud system architecture was made on the basis of those functional and non-functional requirements in a service-oriented manner. Two groups of services were identified: functional services providing the required functionality to the clients of the system and system services supporting security and common aspects of the backend like logging or asynchronous communication.

The designed architecture was also related to the available cloud infrastructure and different deployment options were analysed, focusing on IaaS and SaaS cloud types.

1.3 Overview of the Document

The document consists of four sections. The first one introduces the general aims of the Europeana Cloud project and the motivation for the creation of Europeana Cloud system. It also explains the approach which was taken in order to define the high level architecture of the system.
The second section contains user stories - information obtained from potential users of the Europeana Cloud, explaining their needs in the context of the system and benefits they would like to achieve by using the Europeana Cloud. The user stories come from four sources. First three sources are existing metadata aggregators involved in the creation of the Europeana Cloud system: Europeana, The European Library and Polish Digital Libraries Federation. The last set of user stories comes from other potential users of the system and was compiled on the basis of the feedback gathered during dedicated sessions organized as a part of the Europeana Cloud project kick-off meeting.
The third section of the document lists 9 functional requirements and 6 groups of non-functional requirements. They were defined on the basis of summarized and prioritized user stories from section 2. These requirements were used to define the architecture described in section 4. This section also discusses the possible options of the deployment of the designed system and its possible implementation roadmap.

1.4 Nature and Scope of the Document

This aim of this document is to represent the general architecture of the system developed within the WP2 of Europeana Cloud project. This system will be evolving throughout the duration of the project and after it ends, therefore the nature of the information in this document is also volatile. After each major release of the developed eCloud system, the architecture description presented in this document will be updated to represent the most recent system release. This should be done no less than each six months. The history of this document revisions can be found on its beginning (page 2).

Beside, at the early stage of the eCloud system development, not all of the services and other architectural components described here will be ready and publicly available. Therefore updates of this document will also include references to websites where code and up-to-date technical documentation of services can be found, as soon as these services will be released.

Finally, the vision of architecture described in this document may not be fully implemented during the Europeana Cloud project lifetime, as it exceeds the initial set of ideas expressed in the project Description of Work. The initial set of stages of the development of the eCloud system is described in section 4.4 titled "Possible Implementation Order".

2 User Stories for Europeana Cloud

The design of Europeana Cloud must be backed up by requirements gathered from the potential users of this service. In the context of Europeana Cloud WP2 we are targeting mostly content providers and therefore the main sources of requirements are provided by the partners listed in the table below.

Name	Type	Short description
Europeana	metadata aggregator	Europeana is Europe's multilingual digital library, museum and archive. More than 1,500 heritage institutions contribute cultural content in Europeana. Their number and geographic coverage are steadily growing. The objects relate to science, media and art. They are available in different formats (text, images, audio/video, etc.) and in every European language.
The European Library (TEL)	metadata aggregator	The main task of The European Library is to aggregate metadata from all members consisting of National and Research Libraries. It provides currently access to 117 million bibliographic records which will increase to 150+ million in the near future. Besides bibliographic records, The European Library hosts 25 million full-text records which will also increase in the near future by another 10 million newspaper pages. Furthermore, The European Library is the library aggregator for Europeana and it is the biggest content provider with over 4,5 million records.
Polish Digital Libraries Federation (DLF)	metadata aggregator	The main task of Polish Digital Libraries Federation (DLF, http://fbc.pionier.net.pl/) is to aggregate, process and provide access to metadata describing cultural heritage objects which are made available on-line by Polish cultural and scientific institutions. In middle of 2013 the service aggregated around 90 various data sources including 1.3M metadata records from several hundreds of institutions. The data aggregated by DLF is used by end users as well as by other services like Europeana or DART-Europe.

In this section these requirements are gathered in a form of user stories, documenting relatively high level ideas together with their motivation and/or benefits. User stories are usually told from the perspective of future user or customer of the system. For the purpose of this document the pattern *"As a/an .... I want to ... so that...."* for expressing user stories was selected[4]. In this case direct users will be mostly other information systems instead of humans, but the format itself seems to be relevant for the initial requirements elicitation.

User stories presented in this section served as a basis in the process of formulation of more precise functional and non-functional requirements documented in section 3 of this report.

[4] See *"Advantages of the "As a user, I want" user story template"* for more details on this template http://www.mountaingoatsoftware.com/blog/advantages-of-the-as-a-user-i-want-user-story-template/

2.1 As Europeana...

#	I want to...	so that...
E1.	Have a unique identifier service	I can trace updates of records and ensure that a Europeana ID will not break on any update
E2.	Have support for EDM external and internal metadata formats	I can retrieve metadata from the eCloud and push transformed data in UIM to the eCloud
E3.	Have support for versioning of metadata formats of the metadata	I can identify if content needs to be updated and remapped and I can provide versions of my metadata as a service
E4.	Have support for the current organization and representation of the Provider and Datasets in the Europeana SugarCRM instance	I can register and update aggregators, providers and datasets following the same workflow as the current one*, UIM import plugins can operate smoothly with the new infrastructure
E5.	Have support for retrieval of information based on the Aggregator/ContentProvider and CollectionId and Collection Name	I can represent, organize and search for content in a similar manner as in the current* infrastructure
E6.	Have support for URL based retrieval of content	I can gather statistics for link problems and generate images for the Europeana services as in the current** infrastructure

* EF through its CRM system organizes its content into Content Provider/Aggregators and their corresponding datasets. This is used in the naming convention used for the Europeana Collection and record Ids in the following manner:

- CollectionId: {ProviderId}{LocalCollectionId}
- RecordId: {CollectionID}/{LocalRecordID}

** EF ingestion workflow supports checking and reporting for links that are invalid, and thumbnail generation for use through its services. The plugins that perform this operations require a resolvable http address for access to the content.

2.2 As The European Library...

#	I want to...	so that...
T1.	Have the support of different aspects (e.g. formats, full-text, ...) for the same record	I can access a EDM version of an object as well as the original metadata format or full-text connected to a record, retrievable individually or as bulk

T2.	Have a unique identifier for each record and an identifier service to map local record identifiers to eCloud record identifier	I will have a unified identifier treatment and a way to look up different identifiers to the same eCloud identifier
T3.	Have access to records for individual provider and sub-grouping e.g. collections, catalogues	I will be able to access records in a bulk way for providers, but also with additional grouping like collections and catalogues or also virtual bags
T4.	Manage accessibility of aspects for records (potentially also geographically)	I will be able to hold different formats for a record with different access patterns (restricted for the full MARC record, but no restrictions for limited DC format). Potentially, we also need geographic restrictions, so that some data must stay in certain countries.
T5.	Keep track of changes/additions/deletions	I will be able to perform incremental access to data for individually partners or collections.
T6.	Have individual or bulk access to a certain aspect for records	I will be able to access for example full-text for records bundled by a collection or individually
T7.	Have support for custom formats by exploiting a converter concept	I will be able to store custom formats (serialization of java objects into binary formats) by providing converters between storage and native formats
T8.	Relations between records	I will be able to hold relations between records for example to express similar records or same authorship etc. (multi-graph).

2.3 As Polish Digital Libraries Federation...

#	I want to...	so that...
D1.	store information about a particular object in many different formats (both text and binary) and easily access to all information about this object	I can for example store and access thumbnail of a book cover, metadata of this book in MARC format and metadata of this book in ESE format
D2.	record relations between information about the same object expressed in different data formats	I am aware that information about an object in ESE format is in fact an

		outcome of processing of a MARC record about the same object
D3.	divide stored data by its providers	I can do a selective access to (and processing of) stored data depending on its provider, when for example all metadata records from one digital library will have to be updated after some major changes in this digital library system or metadata schema
D4.	access all metadata records stored in particular format	I can get all data stored in PLMET (DLF's internal format) format from all data providers to publish this data in the DLF portal
D5.	access all metadata records provided by data providers from particular country	I can get all data from Polish institutions to publish this data in the DLF portal
D6.	track dates of the first addition, last modification and also deletion of each data record	I can periodically and incrementally synchronize DLF's search index with the data stored in the eCloud, without the need of getting and processing of all the data from scratch
D7.	get instant notifications after each added/modified/deleted record in the eCloud, matching specific criteria defined by me	I can upon such notification update the DLF's search index and provide users the most recent information about digital objects from Polish cultural heritage institutions
D8.	do a quick scan of the current contents of the cloud without downloading all data records	in case of a disaster and loss of synchronisation between DLF's search index and the eCloud, I can quickly find and reindex missing data records without downloading all the data I am generally interested in (i.e. data from Polish data providers)
D9.	manage public visibility of data records stored in the eCloud at least be setting an "open data" flag	I can store raw data gathered from by data providers, process the data and use it for DLF's internal purposes, and when Europeana's DEA is signed by particular data provider I am able to set the open data flag and release the data covered by particular DEA

2.4 Other Sources of User Stories

The requirements below were extracted from initial high level ideas about the scope of the eCloud system gathered by the team of WP5. These requirements were mostly gathered during the working session of the project kick-off meeting. Not all of them are possible to be entirely fulfilled by the eCloud system, but the eCloud system should provide data storage and access capabilities to support future implementation of these requirements.

#	As a/an...	I want to...	so that...
O1.	End-user	have a manual faceting mechanism	I can create personalized navigation and search interfaces
O2.	Content Provider/Aggregator	get a hosting and storage services for easy access for people with no technical background	I can create personalized access services to my collections
O3.	Content Provider/Aggregator	provide API access profiles	I can modify the level of access I offer to my metadata stored in the eCloud
O4.	End-user	provide information selection criteria	I can manually personalize what information I want access to
O5.	Content Provider/Aggregator	support open file formats and standards for the storage of my content	I can ensure that my content will be re-usable
O6.	Content Provider/Aggregator	to provide me with write access to its content	I can add, remove and update my metadata and content
O7.	Researcher	provide versioning of metadata with persistent URIs	I can refer to a specific bibliographic reference by date
O8.	Europeana Cloud stakeholder	conform both to the international and national legislation framework	I can have full confidence on the use of the metadata and content through the eCloud
O9.	Content Provider/Aggregator	have an authentication system	I can trust that my metadata and content are accessed and used

			only by the appropriate stakeholders

3 Requirements for Europeana Cloud

3.1 Summary of User Stories

The table below contains a summary of user stories from the previous sections of this document. For each user story there are references to particular user stories from Europeana, TEL, DLF and other sources (see Section 2.1-2.4). The stories has been rephrased in order to merge similar stories from different sources.

#	As a user of Europeana Cloud...	Derived from
S1.	**I want to** have a unique identifier assigned for each stored record which will support updates of the data and will be persistent during all updates, mapping local data identifiers into eCloud identifiers **so that** I am able to track records after their updates and resolve eCloud identifiers based on local identifiers	E1, T2
S2.	**I want to** have the possibility to store and access multiple data formats (both text-based and binary) and different versions of particular data format for each record, **so that** I can transform the data taken from the eCloud and store the outcome also in the eCloud, provide parallel versions of data and track which data records require update between data format versions.	E2, E3, T1, T7, D1, D2
S3.	**I want to** have the data records grouped into data providers and datasets, (assuming that each data record belongs to a dataset and has an id unique in the context of that dataset, and that each dataset is operated by a provider and has an id unique in the context of that provider, and that providers also have unique ids in the system and are assigned to particular countries), **so that** I can manage and access the data in structures corresponding to the organization of the data aggregation process.	E4, E5, T3, D3, D5
S4.	**I want to** have all links in the stored data to be verified while placed in the eCloud, **so that** I can be sure that data stored in the eCloud contains resolvable links or be informed that for particular records/datasets/data providers some links are not accessible.	E6
S5.	**I want to** have the possibility to control access to the data records based on combination of parameters like data provider, dataset and data format, data license (explicit and generalized e.g. anything allowed for commercial use) and geographical location of client, **so that** internal or obsolete data formats are not externally accessible, and that I can ensure that data providers security requirements are fulfilled, at the same time allowing clients to flexibly select what they need.	T4, D9, O3, O8, O9

S6.	**I want to** be able to track changes (additions/modifications/deletions) of records in the eCloud, both on request and as subscribed notifications, **so that** I am able to incrementally process data of individual providers, data sets or data formats and also that I know when the record appeared in the eCloud for the first time and when it was deleted.	T5, D6, D7
S7.	**I want to** have the possibility to access data as individual records or as bulks of records (e.g. all records from particular data set, all records in particular format, all records where data provider is from particular country etc.), **so that** I can optimize data access for mass or individual processing, also for indexing the data and providing search/browse end-user interfaces for retrieved data.	T6, D1, D3, D4, D5, O1, O4
S8.	**I want to** be able to provide various data converters/processors and configure the eCloud to automatically convert/process the data records (e.g. between formats), **so that** I can allow data storage and access in different formats.	T7
S9.	**I want to** be able to hold relations between records **so that** I can express similar records or records describing objects with the same authorship etc.	T8
S10.	**I want to** track relationships between records pairs which are respectively input and output of processing/mapping operations, **so that** I am aware what is the source of records which are outcome of processing, for example to repeat the processing if data of processing rules will be updated.	D2, T7
S11.	**I want to** do a quick scan of the current contents of the eCloud without downloading all data records, **so that** I can make sure my database is in sync with the eCloud.	D8
S12.	**I want to** have the support for open file formats while storing the content in the eCloud, **so that** the content which is uploaded in closed file formats is losslessly converted to respective open file formats better prepared for long term storage and access.	O5
S13.	**I want to** have full read/write access to all my data stored in the eCloud, **so that** I can manage my data.	O2, O6
S14.	**I want to** have support for versioning of data records, **so that** using persistent links I can link and access particular version of data record.	O7

Stories ranking. The table below shows more analytical view on the summarised stories and their related source stories (columns 1-5). Columns 6-8 shows the importance rank of summarised stories for each of three main aggregators which will be using first prototypes of the eCloud. Rank values has the following meaning:

- **4** - Critical - features from this user story cannot be missing in the system from its very beginning - must have for the first usable prototype.
- **3** - Important - features from this user story can be introduced later than in the first prototype, but they must be completed to be able to use the system in production mode.
- **2** - Average - features from this user story are important for me, but I can imagine using the system in production mode without them, assuming that they will be implemented later on during the eCloud project.
- **1** - Optional - features from this user story would be interesting to have in the future (which may be also after the eCloud project).
- **0** - Not interesting - I am not interested in using features from this user story.

Story id	**Europeana stories**	**TEL stories**	**DLF stories**	**Other stories**	**Europeana rank**	**TEL rank**	**DLF rank**
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
S1	E1	T2			**4**	**4**	**4**
S2	E2, E3	T1, T7	D1, D2		**4**	**4**	**4**
S3	E4, E5	T3	D3, D5		**4**	**4**	**4**
S4	E6				**3**	1	1
S5		T4	D9	O3, O8, O9	2	**3**	**3**
S6		T5	D6, D7		2	**4**	**4**
S7		T6	D1, D3, D4, D5	O1, O4	**3**	**3**	2
S8		T7			2	**3**	2
S9		T8			1	**2**	1
S10		T7	D2		1	1	**3**
S11			D8		1	1	**2**
S12			O5		**0**	**0**	**0**
S13			O6		0	**4**	**4**
S14			O7		0	**1**	**1**

The ranking above defines the following main stages of system development in the scope of the eCloud project:

- **eCloud Alpha** - All stories with at least one (4) rank are implemented
  - S1. Unique identifiers (12 pts.)

- S2. Storage of multiple data formats and versions (12 pts.)
- S3. Data records grouped into datasets etc. (12 pts.)
- S6. Support for data changes tracking (10 pts.)
- S13. Full read/write access to own data (8 pts.)

- **eCloud Beta** - All stories with at least one (3) rank are implemented
  - S5. Access control (8 pts.)
  - S7. Bulk download (8 pts.)
  - S8. Custom data records converters (7 pts.)
  - S4. Links in records verified (5 pts.)
  - S10. Relations between record pairs (for processing) (5 pts.)
- **eCloud 1.0** - All stories with at least one (2) rank are implemented
  - S9. Various relations between records (4 pts.)
  - S11. Quick scan of eCloud content (4 pts.)
- **Other requirements** - Stories having only ranks (1) or (0)
  - S14. Persistent links to different versions of data records (2 pts.)
  - S12. Automated conversion to open file formats (0 pts.)

Stories dependencies. As the stories are relatively free from technical implementation details, it is hard to precisely define dependencies between them. Although some initial assumptions can be made on the basis of previous experiences of WP2 participants in the design and implementation of data aggregation, processing and provisioning systems. These assumptions are presented in the Figure 2 below.

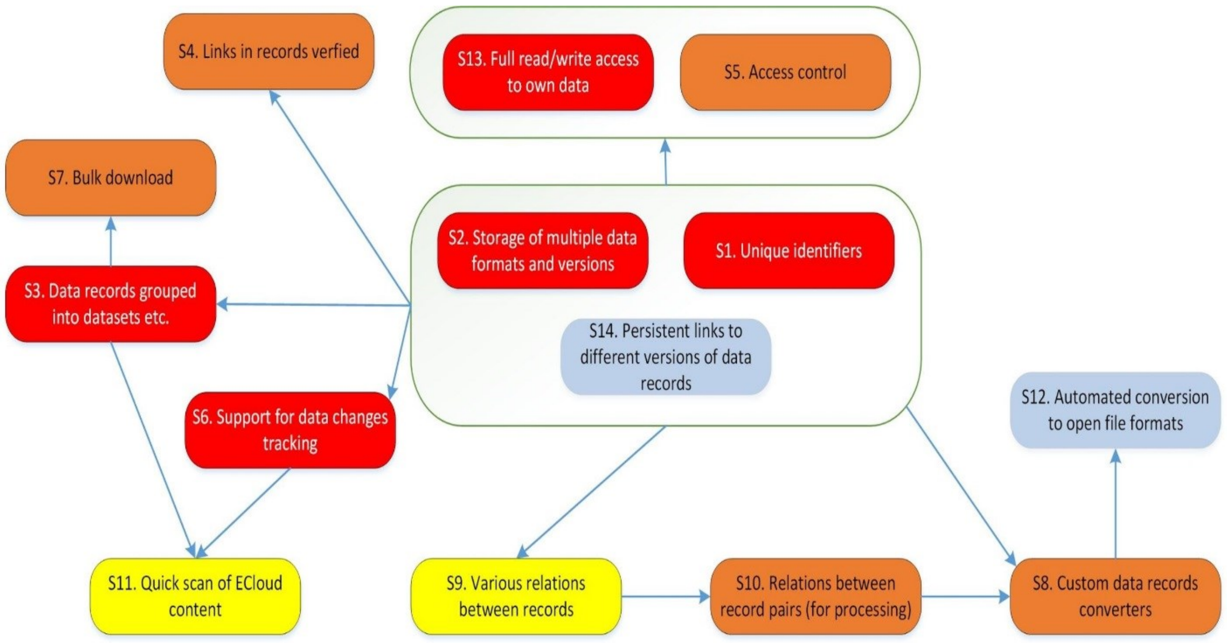

S4. Links in records verfied
S13. Full read/write access to own data
S5. Access control
S7. Bulk download
S2. Storage of multiple data formats and versions
S1. Unique identifiers
S3. Data records grouped into datasets etc.
S14. Persistent links to different versions of data records
S12. Automated conversion to open file formats
S6. Support for data changes tracking
S11. Quick scan of ECloud content
S9. Various relations between records
S10. Relations between record pairs (for processing)
S8. Custom data records converters

Figure 2. Initial dependencies between summarised user stories for Europeana Cloud system with three stages of development: ALPHA (red), BETA (orange) and 1.0 (yellow).

Each colored rectangle represents a single story (with id and a brief title). Blue arrows represent the probable dependencies between them, defining the possible order of implementation (from technical point of view). The story at the beginning of the arrow should be implemented first, as it probably delivers functionality needed by the story at the end of the arrow. Additionally green rectangles group requirements which are so tightly coupled, that at this stage it is hard to suppose that they will be implemented and released separately.

The ordering of stories which takes into account the importance and order of implementation can be the following:

- **Stage A**
  - S1. Unique identifiers (12 pts.)
  - S2. Storage of multiple data formats and versions (12 pts.)
  - S14. Persistent links to different versions of data records (2 pts.)
- **Stage B**
  - S3. Data records grouped into datasets etc. (12 pts.)
  - S6. Support for data changes tracking (10 pts.)
- **Stage C**
  - S13. Full read/write access to own data (8 pts.)
  - S5. Access control (8 pts.)
- ***eCloud Alpha functionality delivered** - All stories with at least one (4) rank are implemented*
- **Stage D**
  - S7. Bulk download (8 pts.)
  - S9. Various relations between records (4 pts.)
  - S10. Relations between record pairs (for processing) (5 pts.)
  - S11. Quick scan of eCloud content (4 pts.)
- **Stage E**
  - S8. Custom data records converters (7 pts.)
  - S4. Links in records verified (5 pts.)
- ***eCloud Beta and 1.0 functionality delivered** - All stories with at least one (2) rank are implemented*
- **Stage F - Other requirements** - Stories having only ranks (1) or (0)
  - S12. Automated conversion to open file formats (0 pts.)

3.2 Functional Requirements

This section contains description of functional requirements built on the basis of previously collected and summarized stories. Main sources of stories were Europeana, TEL and Polish Digital Libraries Federation development teams, which are supposed to develop first clients of Europeana Cloud cloud system. In the context of these three systems, Europeana Cloud is seen as a set of middleware services allowing easy sharing of cultural heritage data and metadata between various services (e.g. aggregators).

In the requirements b we assume that Europeana Cloud system is used by “a client” - a piece of software which is interacting with the eCloud via its API. The software may use the eCloud in a fully automated manner or may be used by its human operator in a direct control mode, executing command by command, as pointed by the operator. With such assumptions, further paragraphs does not cover any end-user interface beside API designed for service to service interaction.

The ordering of requirements below does not reflects their implementation order or priority. More information about the possible order of implementation works can be found in section 4.4.

R1. Unique identifiers service

Source stories: S1. Unique identifiers *(based on E1, T2)*

Description:
Records provided by eCloud clients have local identifiers which may not be unique in a global scale. eCloud keeps provided records in virtual identifiable and non-overlapping spaces, created by authorized clients. These spaces correspond to data providers in Requirement R3. Single client can operate on many spaces, which means that the same client can upload data from many data providers (if authorized).

The system provides service which is able to assign a new globally unique identifier on the basis of local identifier and data provider identifier. The service is also able to resolve already assigned global identifier to combination of data provider identifier and local identifier, and is able to resolve combination of data provider identifier and local identifier to global identifier (if already assigned). The global identifier which will be assigned to the particular combination of data provider identifier and local identifier should be permanent. Once such mapping is established, it should be permanent and even if the provided data record will be deleted from the eCloud, the global identifier should not be reused.
The service may be used directly by authorized clients or indirectly, while invoking create/read/update/delete (CRUD) operations on data records.

R2. Support for storage and access to multiple data formats and versions

Source stories: S2. Storage of multiple data formats and versions *(based on E2, E3, T1, T7, D1, D2)*, S14. Persistent links to different versions of data records *(based on O7)*

Description:
Requirement R1 specifies that each data record should have its unique identifier within the system. There is a need to support several data representations which have different format (e.g. different data schema like ESE-XML and MARC-XML or different encoding of data expressed in the same schema like MARC-XML and binary MARC). These representations should be associated with the same record identifier, but must be accessible separately. There should be also a possibility to list data representations available for particular record identifier. With each data record representation there should be associated information about the format of this representation. Each representations should have assigned one format, but one data record can have two different representations which have the same format. For example a sculpture can have two representations - two different photo sets - which are in the same format (JPG). Therefore representation format should have a unique identifier within the eCloud system. Such unique formats identifiers will create a dictionary of formats used in the eCloud system.

Moreover, each data record representation (expressed in some data format) may have several versions which also must be stored and accessible. Subsequent versions will in most cases correspond to changes which were made to the data. There should be a possibility to add new version of a data record representation, retrieve the list of versions and retrieve particular version. As the system can be used also to store temporary data (e.g. during a loop of data processing and quality check-up), the system should support two kind of data records versions: persistent and temporary versions. While adding a new version, the client must have the possibility to decide what is the status of this version. The temporary versions should be deleted by the system automatically each time there is a new persistent version added. Example sequence of versions in the system is shown on the Figure 3 below.

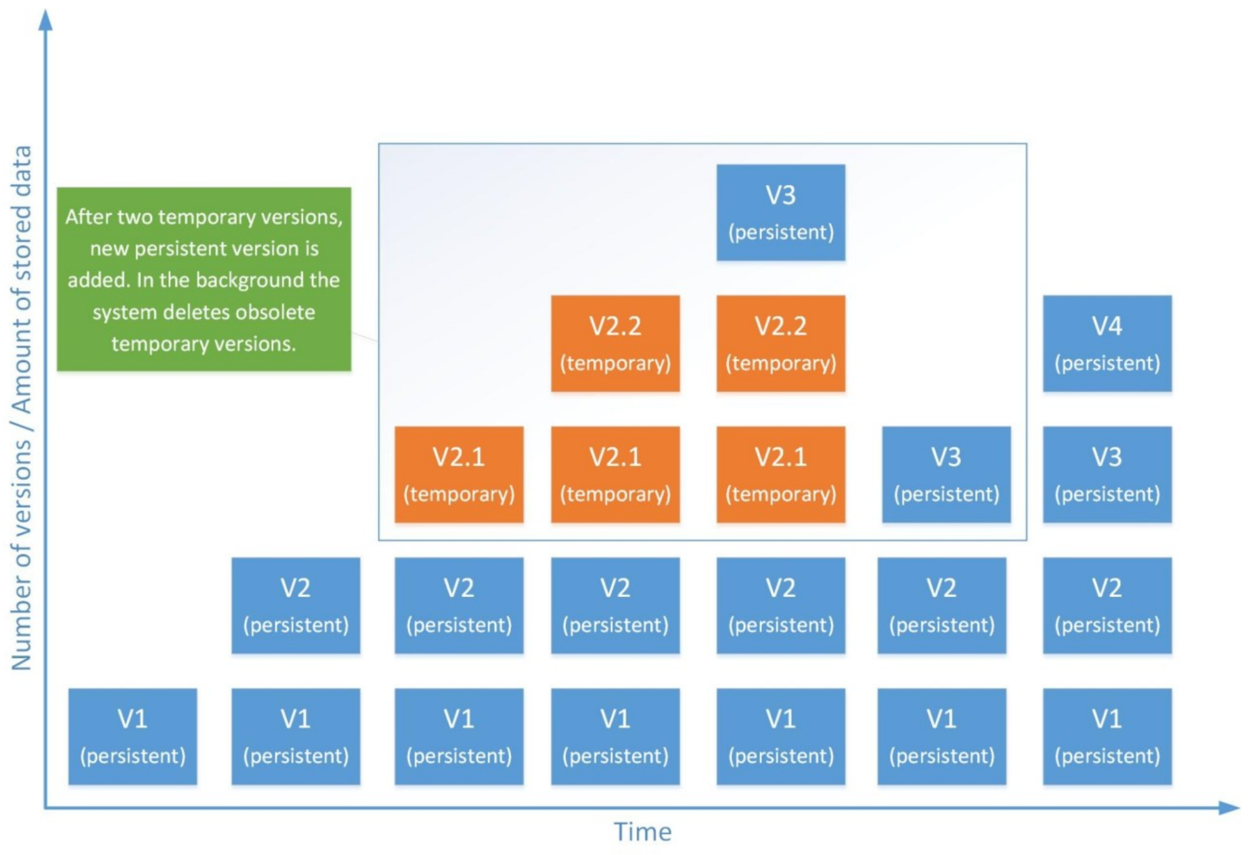

After two temporary versions, new persistent version is added. In the background the system deletes obsolete temporary versions.
Number of versions / Amount of stored data
Time
V1 (persistent)
V2 (persistent)
V2.1 (temporary)
V2.2 (temporary)
V3 (persistent)
V4 (persistent)

Figure 3. Example versions history of a data record.

Deletion can be applied to entire record (includes all representations) or to a representation of a record (includes all versions of this representation). Deletion of particular version is not allowed, to make sure that the record history is consistent. What is possible is an update/replacement of the latest version. This update may include the replacement of the version content or just change of the version kind from temporary to persistent. This is possible to allow more optimized way of interacting with the system in case when:

- Version which was initially temporary was accepted (e.g. at quality assurance stage) and can be now stored as persistent.
- Version which was recently stored in the system has some obvious flaw which makes it unusable and there is no sense in storing such version for the future. The content of this version is replaced with a proper one.

If no version will be specified by the client when accessing the data, client should get the newest version which can be provided (including authorization aspects).

The final data model described in this requirement can be illustrated by the following sample tree structure:

- Record 1
    - Representation 1 (in format A)
        - Version 1
    - Representation 2 (in format B)
        - Version 1
        - Version 2

        - Version 3
- Record 2
    - Representation 1 (in format A)
        - Version 1
    - Representation 2 (in format C)
        - Version 1
        - Version 2
- Record 3
    - Representation 1 (in format C)
        - Version 1

System should offer persistent links to each element of the above tree structure, using the reference methods listed in the table below.

Kind of object	**How it can be referenced?**
Record	● Globally unique identifier (assigned by eCloud system, see requirement R1) or alternatively ● Combination of local identifier and data provider identifier
Representation of a record	● Representation identifier combined with the reference to particular record (see cell above)
Version of a format of a record	● Version identifier (version number) combined with the reference to particular representation of particular record (see cell above)

Identifier of particular version will be assigned by the eCloud system. The identifier must allow to order versions according to the order in which these versions were added to the system. The same ordering can be also achieved with additional technical metadata (like exact date and time when the record was added to the eCloud), but it should be enough to have just identifiers of versions to be able to say, which version is newer one.

R3. Ability to group data records into by data providers and data sets

Source stories: S3. Data records grouped into datasets etc. *(based on E4, E5, T3, D3, and D5)*

Description:
Requirement R2 introduced the complexity of a data record - it can have many representations in different formats and each representation can have many content versions. But there is also a need for organization of data records. Experience of metadata aggregators like TEL, Europeana or DLF shows that the basic structure which is necessary to make the eCloud system useful consists of two layers: data providers and data sets. The relation here is following:
- Each version of a data record is provided by a specific data provider.
- Version of a data record may be assigned by the data provider to some specific data set. This can be achieved by using annotations on the versions of data records.

As stated above, version of a data record is provided by a specific provider, but a data record can have several representations which have versions provided by different data providers.

The eCloud API should be designed in a way which respects the relations between different entities described in the data model. So for example for each version of a digital record available in the system it must be possible to identify:

- Its record identifier
- Its record representation identifier (and its format)
- Its data provider identifier

Additionally each data provider must be described within the eCloud system with basic administrative information like name, country, contact details etc. This information may be useful not only for internal administrative purposes, but also for eCloud clients. For example the following information about data providers can be collected:

- Name of organisation
- Official address
- URL of official organisation's website
- Name of website (organisation's digital library)
- URL of website (organisation's digital library)
- Contact person (name, e-mail, phone)
- Remarks

R4. Change/event tracking

Source stories: S6. Support for data changes tracking *(based on T5, D6, D7)*, S11. Quick scan of eCloud content *(based on D8)*

Description:
The eCloud system should allow changes tracking in two modes:

1. Client can ask about changes in the eCloud system which took place in given period of time (between two given dates of from one given date until now). This request may be about data providers ("which data providers were updated in the given period of time?") or about records of particular data provider ("which records of data provider X were updated in the given period of time?").
2. Client can subscribe (and unsubscribe) for push notifications with update information about data providers ("look, data provider X has new/updated data!") and/or about records of particular data provider ("look, record Q of data provider X has been added/updated/deleted!"). The system will not queue notifications for subscribed clients which went off-line. Such clients will be automatically unsubscribed after a number of unsuccessful notification trials.

In case of mode 1 above, when asking about records of a particular data provider, there should be a possibility to define, whether the response should include:

- just identifiers of records

or

- additional technical metadata of the record (like modification dates, checksum, etc.).

The first type of response will be smaller, and therefore probably faster and will assume that a separate request is needed to get access to the full record. The second type of response will provide more information, and may be useful when the client will be only checking whether his local copy of the eCloud content is in sync with the eCloud.

In order to make this possible the eCloud system must include the following temporal aspects in the technical metadata:

Kind of object	**Technical metadata - temporal aspects**
Data provider	● Set of dates when records of this provider were added/updated/deleted
Record	● Date of creation of a first (oldest) version of a first (oldest) format of this record ● Date of the latest modification (incl. deletion) of the most recently modified representation of this record ● Date of deletion (if the entire record is deleted)
Representation of a record	● Date of creation of a first (oldest) version of this representation of this record ● Date of the newest version of this representation of this record ● Date of deletion (if the entire representation of a record is deleted)
Version of a representation of a record	● Date of creation ● Date of deletion (in case of persistent versions, it must be equal to date of deletion of the entire representation of this record)

R5. Authorization

Source stories: S5. Access control *(based on T4, D9, O3, O8, O9)*, S13. Full read/write access to own data *(based on O2, O6)*

Description:
In the context of authorization, two tiers of system entities that can gain access have been identified:

1. Organisations - an organisation is sharing and accessing content in the eCloud infrastructure. It consists of one or more eCloud system users - clients.
2. Clients - a client communicates with the eCloud services to complete specific operations and should be classified to an organisation. Each client should be identified by a unique identifier and have its login credentials. A client can act on behalf of be a physical person (human operator) or fully automated service client.

The following system roles can be identified organized by type of user:

Human Users

- eCloud Admin - Able to perform any operation within the eCloud system.
- Local Admin - Institutional administrator, able to perform any operation within the eCloud system limited to the scope of the data provided by this particular institution.

- Regular User - Institutional client able to perform reads/writes/deletes on data which he provides, and read access on selected content provided by other users/organizations, according to the authorization rules set by these users/organizations.

Systems

- Service - Service that can perform operations interacting with components of the system.

The following levels of access need to be supported

- Private - Granted only to the organization that offers the content to the Cloud. Any other participating organization cannot retrieve this content
- Limited - Granted to the organization that offers the content to the Cloud and other specified organizations. Only selected organizations can access the content
- Public - Granted to anyone

The following levels of rights have been identified

- Write - Specifies that a client can modify (write/delete/update) records.
- Read - Specifies that a client can only read records.

R6. Records annotations

Source stories: S9. Various relations between records *(based on T8)*, S10. Relations between record pairs (for processing) *(based on D2, T7)*

Description:
The eCloud system should provide the functionality to annotate records with additional relations/statements. The mechanism should be generic and should allow to store and access statements about:

- particular records,
- particular formats of records,
- particular versions of formats of records.

The statements should consist of three elements (similarly to Semantic Web triples):

- subject
- predicate
- object

Subject should be an identifier of one of the entities in the system (record, format of a record, version of a format of a record), object also can be such identifier but it can also refer to some external entity (ideally by providing URI). The predicate part should clearly define the kind of relationship between subject and predicate. Example usage of this functionality can be the following:

- Version Z *is an outcome of migration of* version X
- Record C *is digital representation of the same physical object as* record F

These statements can be added to the eCloud system by the client, but the eCloud system can also generate such statements upon some automated activities (like migration of data records between formats). There should be a possibility to query statements in which providing one, two or three elements of the statement as a query, e.g.:

- Give me all statements where version X is a subject,
- Give me all statements where version X is an object,
- Give me all statements where version X is an object and the predicate is "*is an outcome of migration of*",

- ...

Access to these statements should be authorized. The creator of statement should be able to define whether the statement is publicly accessible or the access is restricted to some specific clients.

R7. Bulk download of data

Source stories: S7. Bulk download *(based on T6, D1, D3, D4, D5, O1, O4)*

Description:
There should be a possibility to do the bulk download of data records. The client should be able to submit a request specifying the selection criteria of records (e.g. all records from particular data set, all records in particular format, all records where data provider is from particular country, all records from particular provider modified after given date etc.) and get in response content of all records versions falling into the given scope. For performance reasons this operation may be executed in an asynchronous manner.

R8. Initial verification of records

Source stories: S4. Links in records verified *(based on E6)*

Description:
There should be a possibility to define automated verification procedures for records stored in the eCloud system. Such procedures may be applied only to selected records versions e.g. records in particular format or from particular provider. Example of such verification is the verification of all links provided in a data record (e.g. XML-encoded metadata record).

As the verification of records can take time, it should be an asynchronous operation. The outcomes of verification should be stored as a part of the technical metadata of the verified record version. This can be also done using records annotations (see R6).

The change tracking mechanism (see R4) can be used to notify the submitter of data that the data has been verified (positively or negatively). The negative verification of records should not lead to automated deletion of records.

R9. Automated conversion of records

Source stories: S8. Custom data records converters *(based on T7)*, S10. Relations between record pairs (for processing) *(based on D2, T7)*, S12. Automated conversion to open file formats *(based on O5)*

Description:
The eCloud system should give the possibility to perform automated conversion/processing of data records. The processing can take as an input a version of a format of a record and returns new version of this record in the same format (data processing) or new version of different format of the record (data conversion). There should be possibility to initiate the conversion/processing in two ways:

- by manually initiating conversion/processing of a selection of records;
- by defining a rule in the system that each new record meeting particular criteria should be conversed/processed in a specific way.

The system should allow authorized clients to define such conversion/processing mechanisms as references to on-line services or by providing processing code.

Such conversion/processing should automatically generate annotations (see R6) connecting the input and output of this operation.

3.3 Non-functional Requirements

Functional requirements described in the previous section determine how the eCloud system should behave to meet functional needs of its clients. It is necessary to fulfil them in order to provide the functionality for which the system is intended. But, according to the ISO/IEC 25010:2011 standard[5], the functional suitability and usability are just two of the eight core characteristics of the product quality model. This model describes both static properties of the developed software and dynamic properties of constructed computer system. Beside the functional suitability and usability, the characteristics and their coverage are following:

- Performance efficiency:
  - response, processing times and throughput rates of a system;
  - the amounts and types of resources used by a system, when performing its functions;
  - the maximum limits of a product or system parameter.
- Compatibility:
  - product can perform its functions efficiently while sharing environment and resources with other products;
  - a system can exchange information with other systems and use the information that has been exchanged.
- Reliability:
  - system is operational and accessible when required for use;
  - system meets needs for reliability under normal operation;
  - system operates as intended despite the presence of hardware or software faults;
  - system can recover data affected and re-establish the desired state of the system is case of an interruption or a failure.
- Security:
  - system ensures that data are accessible only to those authorized to have access;
  - system prevents unauthorized access to, or modification of, computer programs or data;
  - actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later;
  - actions of an entity can be traced uniquely to the entity;
  - the identity of a subject or resource can be proved to be the one claimed.
- Maintainability:
  - system is composed of components such that a change to one component has minimal impact on other components;
  - an asset can be used in more than one system, or in building other assets;
  - it is possible to assess the impact of an intended change (analysability);

[5] ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models

[5]http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733

5

  - system can be effectively and efficiently modified without introducing defects or degrading existing product quality;
  - test criteria can be established for a system.
- Portability:
  - system can effectively and efficiently be adapted for different or evolving hardware, software or usage environments;
  - system can be successfully installed and/or uninstalled in an efficient way;
  - product can be replaced by another specified software product for the same purpose in the same environment.

These characteristics are closely related to non-functional requirements - criteria that can be used to describe how the system operates while providing the required functionality. The sections below describe what are the requirements for the designed Europeana Cloud system in the six key quality characteristics described briefly above.

3.3.1 Performance efficiency

For performance and reliability reasons, the Europeana Cloud should consider that replication of information is required into more than one node. Replication both inside a single data center and between several data centres should be considered. Performance consideration should play a major role for how intensively and in what way replication should be done between data centres.

During data ingestion where large volumes of data comes in in a batches, the replication might be a costly operation.

During access to data the system should be able to do the load balancing and decide to fetch the content based on speed of retrieval. In this context, speed of retrieval refers to a factor of system load of the target node and node latency between the requesting system and the target node.

3.3.2 Compatibility

The system should have a well-documented API, tailored for the invocation by other systems (already known are the ingestion frameworks used by Europeana and The European Library which are part of the functional requirements) but also for surplus services that might be built on top of it.

3.3.3 Reliability

As mentioned above, the data shared within the eCloud should be replicated to >1 nodes, to ensure that even if a full data-center goes down, there would be at least 1 more node to serve the requested content. For safety reasons the data should be replicated in at least two geographical locations and there should be replication policy assuring that at least two copies of data are available even when one of the data centres will go down entirely. This means that in case of two data centres each must have at least two copies of data.

The eCloud should be carefully designed taking into consideration the H/W allocation of each participating stakeholder, so that uneven distribution of data, system/network load is either avoided or carefully managed.

The system should be self-organizing: if a node is removed for any reason, the eCloud should acknowledge that and stop serving requests through that node. When the node is back again, proper data should be replicated to it from other nodes. This must not affect client or system interaction.

If a node disappears from the eCloud during replication, data integrity must be ensured, while the replication should be able to be resumed from the point it was paused.

3.3.4 Security

Each client should have read/write access to the underlying infrastructure in a secured and reliable manner. The system should be responsible to grant or deny access for a specific operation based on client rights.

There must be roles defined both on data and content level and on system level. Certain subsystems have rights to perform specific operations according to:

- who invokes them
- what the operation is
- who the content is targeted for

Every action in the eCloud should be kept in a log file or database regardless if they are system or service calls, logging information about the client as well. These should be available to the main eCloud admin at any time. Part of the logs (related to activities/content of particular client) is also available to the client (possibly upon request).

3.3.5 Maintainability

System should be developed in a modular manner. Each piece of software developed should have documentation, maintaining compatibility across older and newer versions of the code developed, implementing a specific functionality. The code written should conform to code conventions and best practices of software development
e.g.
http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html

Each module or subsystem will complete a specific operation. If this operation is required by other modules then this module will become a dependency for the other modules rather than duplicating functionality.
If a technology becomes obsolete/substituted the new module that will replace the technology-bound functionality will have to have a minimum impact on the functionality of the rest of the modules. Therefore each such module should have its public API very well specified and documented. The API should be technologically agnostic (e.g. REST).

3.3.6 Portability

As part of the Europeana Cloud, each node should be able to meet the minimum requirements on the following

- Operating System
- Hardware configuration (e.g. OS specific H/W, VMs)
- Technology support

as they are going to be set from the technologies to be used.
Ideally, the technologies should be OS agnostic and they should be deployable on any of the major OSes.
The installation procedure should be well documented (and automated to maximum possible extent) for all the major operating systems clearly stating all the steps required to successfully install the system including:

- Software installation and configuration for custom code
- Software installation and configuration for external applications

The system should be developed in such a manner that a change in the underlying infrastructure (e.g. storage technology) can be implemented with

- the minimum amount of impact on the system
- the ability to rollback easily
- avoiding breaking functionality

Uninstalling the system should also be documented.

4 Europeana Cloud Architecture

4.1 Architecture Overview

On the basis of the requirements gathered in the previous section, the architecture of the Europeana Cloud system is composed as shown on the Figure 4. The entire system from the client point of view can be seen as a Software as a Service cloud and can be used as any other service available on the Internet. If only a very limited storage-related set of the eCloud system functionality will be used, the system can be seen as the Infrastructure as a Service cloud.

Client has several services which he can use, each service providing its own API. In order to assure the vertical scalability of services, each service is implemented in a stateless manner, according to REST architectural style.

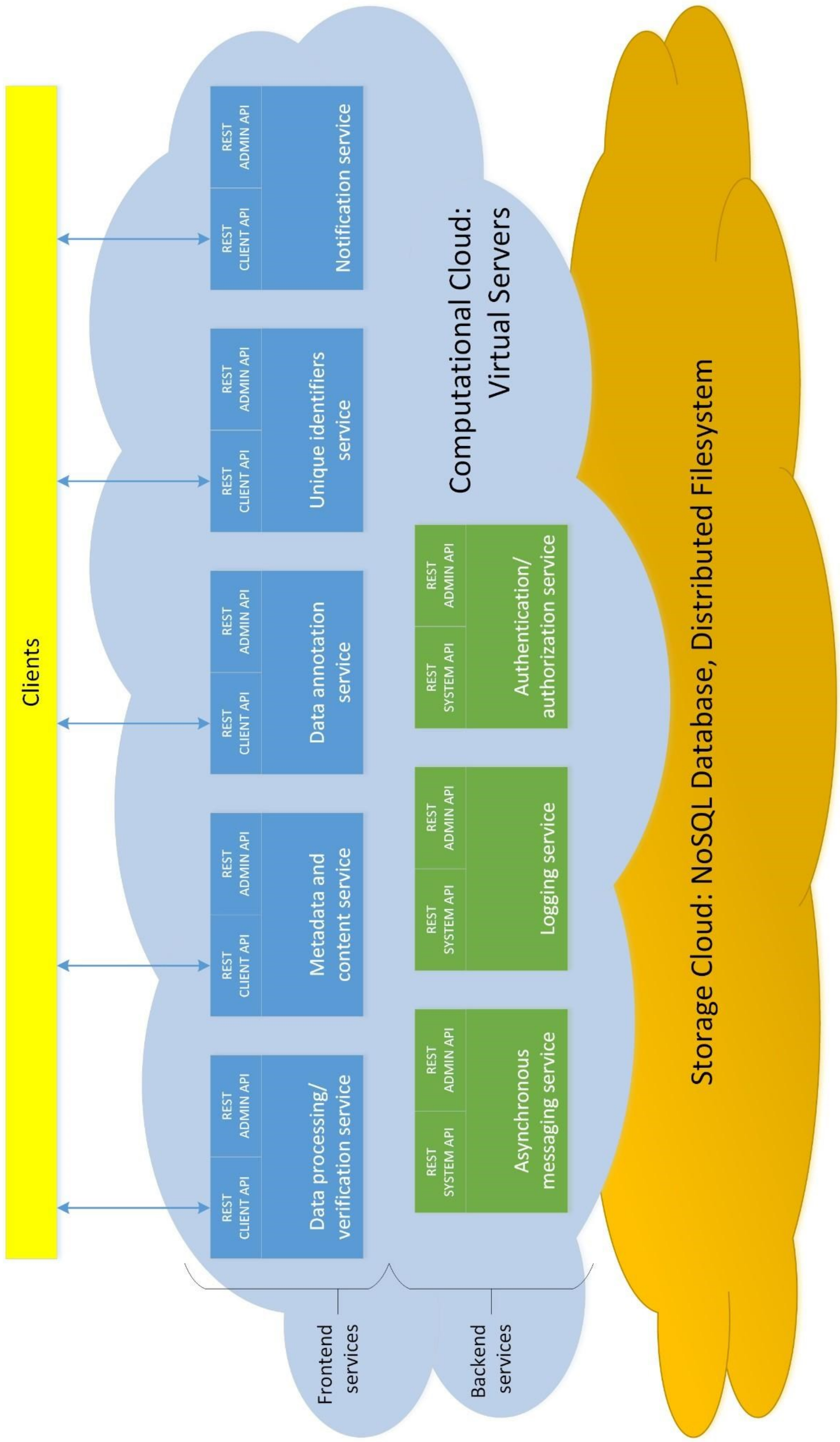
The system is deployed on two types of cloud:

- Computational Cloud (virtual servers) is used to provide computing capacity for services.
- Storage Cloud (NoSQL database and distributed file system) is used to provide storage capacity for services deployed in the Computational Cloud.

The discussion about different options of deployment of the Europeana Cloud in the public/private clouds environment is presented in section 4.3.

In the Computational Cloud there are two layers of services:

- Frontend services layer - services which are available for the clients of the eCloud system. These are so called functional services (blue colour), which are responsible for specific pieces of functional requirements fulfilled by the system. Functional services expose two kinds of API:
    - Client API - for using the service by clients;
    - Admin API - for managing the service.
- Backend services layer - internal services which are not available directly for end users, but are used by eCloud administrators and by other services. It contains so called system services (green colour), which are responsible mostly for non-functional requirements which appear in many aspects of the system. Majority of their functionality should be possible to achieve with out-of-the-box components. System services expose two kinds of API:
    - System API - for service to service communication, for accessing the system service functionality;
    - Admin API - for managing the service.



Clients
REST CLIENT API
REST ADMIN API
Notification service
Unique identifiers service
Data annotation service
Metadata and content service
Data processing/ verification service
REST SYSTEM API
Authentication/ authorization service
Logging service
Asynchronous messaging service
Computational Cloud: Virtual Servers
Storage Cloud: NoSQL Database, Distributed Filesystem
Frontend services
Backend services

Figure 4. General Europeana Cloud Architecture - services view

4.2 Description of Europeana Cloud Services

4.2.1 Unique Identifier Service (UIS)

Description: The Unique Identifier Service (UIS) provides the mechanism to create mappings between local identifier (scoped with the data provider) and the global identifiers inside the eCloud scope. Once these mappings have been established, they will never again be erased. Besides creation these mappings can be looked up both ways, in other words having a local identifier and a data provider identifier it is possible to do the lookup of a global identifier and vice versa. These unique identifier in the scope of eCloud uniquely identifies metadata and content entries in the storage.

The UIS should support linking multiple identifiers per record, as different organizations will have different identifiers.

Creation functionality

- **Create global identifier:** Creates a new global identifier for a given local identifier
  - Provide local identifier and data provider identifier
  - Returns new global identifier, if it does not exist
  - If global identifier already exist, return error message
- **Add unique mapping of identifier to global identifier:** Provide additional local identifier to global identifier mappings
  - Provide local identifier and data provider identifier
  - Provide global identifier
  - Return status message or error message
  - These mappings must be N to 1 (one global identifier can resolve into multiple local identifiers, but only one global identifier can be obtained for a local identifier)
  -

Lookup functionality

- **Retrieve global identifier:** Retrieve an existing global identifier for a given local identifier
  - Provide local identifier and data provider identifier
  - Returns new global identifier, if it does not exist
- **Retrieve local identifiers:** Retrieve all local identifiers connected to a specific global identifier:
  - Provide global identifier
  - 0 to N local identifiers (in most cases should be one)

Relation to other services:
Metadata and Content Service: UIS provides the unique persistent identifiers required by the MCS on the metadata and content records to operate correctly

Technological remarks:
This service should provide a good response time and established mappings must be highly consistent. It would be also good to generate global identifiers in a human friendly manner (for example easy to note down from the screen or use as a part of a URL in a message or a presentation slide). This service should be also optimized for batch use scenarios both in identifier creation and identifier lookup context.

4.2.2 Metadata and Content Service (MCS)

Description:
The Metadata and Content Service (MCS) provides the CRUD operations for metadata records as well as content objects in multiple representations and versions. This service provides the actual API for clients, while the physical storage solution is abstracted by an underlying cloud service. Access to a particular entity like metadata and content or a bulk of these entities can be achieved by a global identifier, a representation form and optionally a version number (otherwise the latest is returned). Besides access the service provides also creation, update and delete operation.

Retrieve functionality

- **Retrieve entity:** Retrieve an existing global identifier for a given local identifier
  - Provide global identifier and a representation form
  - Optionally provide version number or automatically choose the latest
  - Returns the entity in the requested representation (internally it could be automatically converted between formats) and version
- **Retrieve entities:** Retrieve all local identifiers connected to a specific global identifier:
  - Provide global identifiers and a representation form
  - Optionally provide version number or automatically choose the latest
  - Returns the entities in the requested representation (internally it could be automatically converted between formats) and version, number can be smaller than amount of global identifiers if they are not existing
- **Retrieve global identifiers:** Provides identifiers for a specific data set or data provider
  - Provide a data set or data provider
  - Global identifiers connected to a specific data set and data provider
- **Retrieve representations (and optionally version numbers):** Retrieve all representation forms and optionally version numbers available for a specific general identifier
  - Provide global identifier
  - Returns representation forms (and optionally version numbers) for this global identifier

Administration functionality

- **Create/Update entity:** Updates or implicitly creates a new entity given in a specific representation
  - Provide global identifier and representation form
  - Provide content of the entity in the given representation
- **Delete entity:** Delete an entity including all versions and representations
  - Provide global identifier
  - Return status message
- **Delete entity representation:** Delete an entity representation including all versions
  - Provide global identifier and representation form
  - Return status message

Relation to other services:
Unique Identifier Service: UIS provides the unique persistent identifiers for record identification and retrieval

Notification Service: When a record is updated the interested authorized subsystems should be notified of the changes
Data Processing service: This service is directly related to the Data processing/verification service, as the latter uses primarily this service to retrieve and save/update the data record.
Data Annotation service: This service is directly related to the Data Annotation Service, as the latter stores annotations on the data records, representations and versions stored in the MCS.
Logging Service: Every action on the metadata records via the data storage service should be logged.

4.2.3 Notification service (NS)

Description:
Notification Service (NS) provides the communication mechanism between the internal services and external clients. NS mechanism should support both on-demand client requests and push notifications to registered clients, for any change operation that alters the content of the database. It will support client registration for a Push messaging mechanism, de-registration, and automatic identification of users that go offline. The NS will support a REST API that will give access to the required information using JSON output. There are two types of identified user messages: Client messaging functionality and Administration messaging functionality.

Client messaging functionality
NS will expose to the registered clients all the modifications that affect data records, with the modification date and versioning information, if required. It is suggested that there are profiles of notification messages to minimize traffic generated. The types of messages supported are:

- **Provider-related messages:** Retrieval of information about a provider's data sets supporting:
  - Last update of a provider's dataset
  - List of provider's data sets
  - Date of creation of a related dataset in the system
  - Date of update of a dataset (if updated)
  - Date of deletion of a dataset (if deleted)

  An extended profile should also include
  - Versions/History of the provider's data sets
- **Record-related messages:** Retrieval of information about a record supporting:
  - Date of creation of the record
  - Date of update of the record
  - Date of deletion of the record

  An extended profile should also include
  - Versions of the record

Administration messages
The administration messages control the way the user can register and de-register from the service. In order for the client to register, it should specify the following:
- A unique identification of the client
- For what types of messages and data sets the client wants to register

Relation to other services:
Metadata and Content service: The MCS communicates with NS to inform that a modification happened on a record level so that the appropriate registered clients are notified

Authentication/Authorization service: The AAS provides the identification necessary for the registration of the clients to the NS

Technological remarks:
This service can be based on frameworks like Apache ActiveMQ. Possible notification protocol is XMPP.

4.2.4 Data annotation service (DAS)

Description:
The Metadata and Content Service allows clients to store and access data records which can have many representations and each representation can have many versions. Any entity in this three levels data model (data record - representation - version) has a unique identifier based on the data record identifier assigned by the UIS and can be precisely referenced.

The task of the Data Annotation Service (DAS) is to allow to store and access any additional information related to any entity of the data model. The information should be formed in triples (similarly to Semantic Web triples) expressing relations:

- **Subject:** The first element of the triple should be always an identifier of an entity (data record or its particular representation or even its particular version)
- **Predicate:** The second element should be a relatively short text string expressing/identifying the type of relation. The type of relation should not be limited in any way, but probably as the system will be used by more and more clients, a list of commonly used relations will be created and maintained.
- **Object:** The third element can be an identifier, exactly as the first element, but it can also be a text or XML encoded information. Larger pieces of data will probably not be allowed for performance reasons (they should be stored in the MCS and properly referenced).

Example usage of this functionality can be the following:

- Version Z *is an outcome of migration of* version X
- Record C *is another edition of a book represented by* record F

These statements can be added to the eCloud system by the client, but the eCloud system can also generate such statements upon some automated activities (like migration of data records between formats).

There should be a possibility to query the information stored in this service providing one, two or even three elements of the statement as a query. Queries with one element will allow the discovery of additional information. Queries with two elements will allow exploration of the data (traversal of the graph). Queries with all three elements will allow to confirm if particular information is known to the DAS.

Access to these statements should be authorized. The creator of statement should be able to define whether the statement is publicly accessible or the access is restricted to some specific clients.

Retrieve functionality

- **Retrieve all triples with particular subject (and optionally predicate)**
    - Provide global identifier of the subject

    - Optionally provide predicate
    - Returns all relations matching given criteria
- **Retrieve all triples with particular object (and optionally predicate)**
    - Provide global identifier of the object
    - Optionally provide predicate
    - Returns all relations matching given criteria

Create/Update/Delete functionality

- **Create/Update triple**
    - Provide global identifier for subject
    - Provide predicate
    - Provide object
    - Return status message - including proper error if given triple already exists, as triples must be unique
- **Delete triple(s) with given subject**
    - Provide subject - if this is the only parameter, all triples with this subject will be deleted. This can be done automatically, when the subject will be deleted from the MCS.
    - Optionally provide predicate - to delete all triples with given subject and predicate
    - Optionally provide object - to delete exactly one triple
    - Return status message on deletion
- **Delete triple(s) with given object**
    - Provide object - if this is the only parameter, all triples with this object will be deleted. This can be done automatically, when the objects will be deleted from the MCS.
    - Optionally provide predicate - to delete all triples with given subject and predicate
    - Return status message on deletion

Relation to other services:
Unique Identifier Service: UIS provides the unique persistent identifiers for identification of objects (and possibly also subjects)
Notification Service: When a triple is added, updated or deleted the participating organization subsystems may be notified of the changes
Logging Service: Every CRUD action should be logged

Technological remarks:
To assure proper performance, it may be recommended to use dedicated graph database like Neo4J (http://www.neo4j.org/).

4.2.5 Data processing service (DPS)

Description:
The MCS and DAS services described above allow to store and access data. To implement some of the functional requirements listed earlier in the document it is also necessary to assure that the eCloud system will provide data processing facility. The workflow of this service should consist of three fully configurable steps similar to the well-known extract-transform-load (ETL) cycle:

1. Extract - load data record(s) from MCS

2. Transform - process data record(s) according to specified configuration - the processing can be done by specific processing services which may be in the future deployed as a part of the eCloud (for example link verification service) or by services deployed by a client. There can be also scenarios where the transform step is omitted.
3. Load - store transformation results into MCS, DAS or other external service if provided in the configuration.

The data processing cycle described above can be triggered in two ways:

1. Client implicitly requests data processing and provides all necessary information:
    a. What data record(s) should be processed
    b. How they should be processed
    c. Where they should be stored
2. Client configures the DPS to react on specific notifications from the NS:
    a. Information what data should be processed is based on notification (for example DPS is configured to react on any new or modified data record version which is delivered in the Europeana Semantic Elements format)
    b. The way of processing is defined by the client when configuring the DPS
    c. The place where the outcomes should be stored is also defined by the client at the configuration stage

The overall interaction is presented on the Figure 5 below.

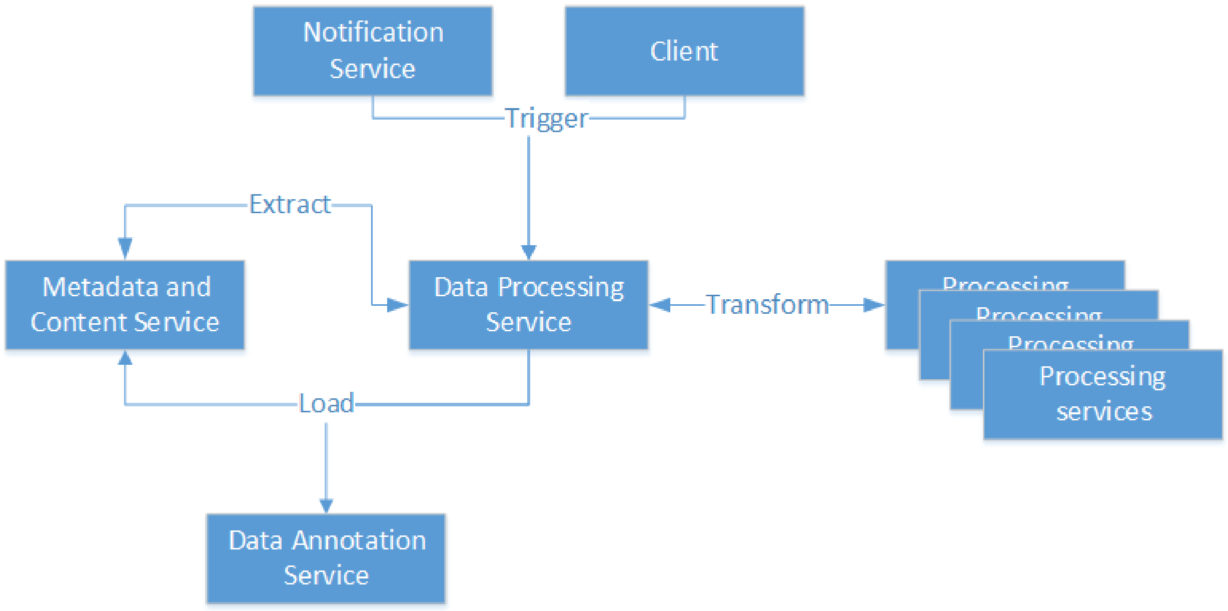


Figure 5. Interaction between eCloud services for data processing purposes.

Possible scenarios which can be implemented with this service:

- Processing of new data to open file formats:
    - Trigger: Upon each new or modified data record version stored in MCS in .doc format
    - Extract: Load given data record version from MCS
    - Transform: Do the transformation to Open Document Format with the service X

    - Load: Save the outcome of processing as a new version of the ODF representation of the data record in the MCS
- Processing of data to more accessible format upon client request:
    - Trigger: Implicit client request
    - Extract: Load given data record version from MCS
    - Transform: Transform the data record version from TIFF to JPEG, 80% quality
    - Load: Send the output of the transformation to URL specified by the client with HTTP PUT method
- Verification of data records stored in particular format:
    - Trigger: Upon each new or modified data record version stored in the Europeana Data Model format
    - Extract: Load given data record version from MCS
    - Transform: Check if the data record matches all requirements described in the EDM specification
    - Load: Store the check result in the DAS as new annotation to the checked data record version
- Verification of links (URLs) in data records stored in particular format:
    - Trigger: Upon each new or modified data record version stored in the Europeana Data Model format
    - Extract: Load given data record version from MCS
    - Transform: Check if the text string extracted from the content of the data record via the given XPath expression is a valid and active HTTP address.
    - Load: Store the check result in the DAS as new annotation to the checked data record version
- Feeding of external indexing services:
    - Trigger: Upon each new or modified record version stored in the EDM format
    - Extract: Load given data record version from MCS
    - Transform: Skip this step - do nothing
    - Load: Send the loaded data record version to URL specified by the client with HTTP PUT method

The service should support a reporting mechanism for the results of its operations, which can be achieved with the logging service. The API of the service should allow to define new processing tasks, execute them and delete them. The set-up of particular services used during the transformation stage is not covered by this service. This is the responsibility of interested clients or eCloud administrators.

If the transform step is taking longer period of time, the processing service should utilize the “progress resource” REST pattern[6]. Alternatively the Notification Service can be also used for this purpose.

Relation to other services:
Metadata and Content Service: Is used during extract and load stages.
Notification Service: Can be used to trigger the service processing cycle. Also when the operation of the service finishes off, the user/system should be notified for the results of the operation.

[6] See for example: http://www.adayinthelifeof.nl/2011/06/02/asynchronous-operations-in-rest/

Logging service: Each data processing cycle and all administrative API calls (for managing processing tasks) should be logged.

Technological remarks:
Possible bottleneck in the implementation of this service is that during the extract stage the data record version must be loaded via this service and sent to the processing service for the transformation stage. If possible instead the DPS should send to the processing service a dedicated, secret and preauthorized URL allowing to access the data directly from the MCS.

4.2.6 Authentication/authorization service (AAS)

Description:
This service should provide the functionality of user and system authentication and authorization.

Authentication process implements identification of users based on the identity proofs that they present. The community character of Europeana Cloud requires the authentication process to be compatible with authentication mechanisms of existing and future partners. The process will support pluggable data sources, such as LDAP or Active Directory.

Authorization process controls access by users to various resources by verifying that the user is permitted to perform a desired operation on a resource. The permission is granted or declined depending on the attributes of the user identity or on roles explicitly assigned to the user by a system administrator. Some roles will be assigned by default based on the user's affiliation with a certain community partner.

The permission of actions on resources will be granted or declined depending on the attributes of the resource such as: the type of resource (system or data), public or private access, licences protecting data resources (for in case of data resource, its institutional source, data type.

Relation to other services:
Logging service: Some registered actions, such as, for example, login or logout by users and systems or critical actions on data objects should be logged by the Logging Service.

Technological remarks:
This service can be based on a ready framework like Apache Shiro.

4.2.7 Logging service (LS)

Description:
This service should provide the functionality of logging the user/system actions. The logs should be written to a dedicated logging infrastructure which should support further processing of logs (e.g. logs analysis). Further, logs should be available for monitoring by human operator and automated tools. Each log entry should consist of a timestamp, identifier of the user identity, executed action, and the identifier of the object on which the action is performed, where applicable. The IP address of the client should be also recorded. Querying of logs should be also supported.

The service should support parallel logging by many services and should have very good performance for storing new entries and high reliability and availability. Archiving or deletion of old logs should be also supported.

Relation to other services:
Data storage service: Every modification operation on the data storage contents should be logged by the logging service
Data annotation service: Every modification operation of the data annotation service should be logged by the logging service
Data processing/verification service: Every execution of the data processing/verification service should be logged by the logging service
Authentication service: Every log-in attempt should be logged by the authentication service

Technological remarks:
This service can be based on frameworks like Apache Flume.

4.2.8 Asynchronous messaging service (AMS)

Description:
These services should provide the functionality of internal messaging mechanism for the data exchange between system services. This service should be decoupled by the NS, even though they share the same functionality, as NS provides the interface to the external communications while AMS to internal services. For usability reasons, it is also preferable to decouple external messages from internal messages, so that, even if the internal messaging mechanism is overloaded or has to be restarted, external services continue to operate, and vice versa. AMS should support the same operations as NS on an internal level. The service should support both on-demand and Push notifications, while each internal service instance can register and de-register as a typical client of the NS can. The messages should be provided via a REST API in a JSON format. The types of messages sent and consumed via this service can be split into the same two types of categories as the NS service: Client related messages and Administrative messages. The description of the types of messages is the same as the ones provided in the NS.

Relation to other services:
Metadata and Content service: The MCS communicates with AMS to inform that a modification happened on a record level so that the appropriate registered services are notified
Authentication/Authorization service: The AAS provides the identification necessary for the registration of the services to the AMS

Technological remarks:
This service can be based on the technologies like JMS (for example Apache ActiveMQ implementation) or ESB (for example implementations like Apache ServiceMix or Mule).

4.3 Discussion on Deployment Options

Deployment options and a selection of available cloud technologies are presented in “Appendix A. Deployment Options for Europeana Cloud: Report on the Evaluation of Available Cloud Technologies”. Section 5.2 of this appendix, titled “How is eCloud Delivering the Cloud Technology?” presents several recommendations (cited below) regarding the way in which the eCloud system should be deployed. These options are briefly discussed above, for more details please refer to the mentioned appendix.

- The eCloud system should be based on a distributed infrastructure which parts are owned by the cultural heritage institutions ready to participate in such technological enterprise.

Participating institutions should have the possibility to offer both high-end as well as cheap commodity hardware. If the infrastructure offered by these institutions will be not enough (in terms of storage or computing capacities), the system should have the possibility to scale – permanently or temporarily – to public (commercial) cloud systems.

- **Recommendation 1:** *The eCloud infrastructure should build on the hybrid cloud concept balancing the specific benefits and drawbacks of on premise and off premise resources to fit the needs of the consortium and potential customers. The software should be agnostic with respect to the infrastructure in which it is deployed.*
- **Recommendation 2:** *eCloud should support the process of allocating and balancing storage and computational resources between public and private cloud environments.*
- **Recommendation 3.** *The on-premise (non-public) part of the eCloud infrastructure should be delivered using a community-based approach. Organisations should be able to install eCloud software in their data centres and allow eCloud to make use of their own computational and storage resources.*
- **Recommendation 6.** *The community part of the underlying storage system should be able to utilize cheap commodity hardware and replicate across multiple data centres. To add an extra level of elasticity or improved durability, it should support the integration with cheap IaaS storage.*

- The infrastructure on which the eCloud system will be running should offer certain level of availability, reliability, performance and security which should be clearly defined, to avoid having infrastructure nodes which will not be able to recompense the effort required for their maintenance. However, the system should be implemented in a way which takes into account the possibility of unexpected temporal or permanent loss of all the infrastructure provided by a specific partner.
  - **Recommendation 4.** *Each partner organisation offering computational or storage resources for eCloud must comply with a set of availability and security requirements specified and regularly updated by the eCloud community.*
- The eCloud system is a long-term undertaking. It should be designed and deployed in a way which allows to migrate components of the system between different infrastructure providers relatively easy. This should allow to avoid vendor lock-in effect which may result in increased costs of running the system and may threaten the availability of the system in case of instability or unavailability of the vendor infrastructure.
  - **Recommendation 5.** *Dependence of eCloud software on specific commercial services and software should be avoided where possible.*
- The technical system should be accompanied by a proper organizational environment, taking care about aspects such as costs monitoring and optimisation, maintenance and development of the system, etc. Without a mature organizational model the system will not be reliable, never mind how good the technological basis will be.
  - **Recommendation 7.** *Computational and storage costs should be monitored over the project duration.*
  - **Recommendation 8.** *The business, sustainability and governance model will need to consider the need for establishing a Helpdesk with appropriate responsibilities and service guarantees and a procedure for requesting new eCloud features.*

4.4 Possible Implementation Order

The table below shows dependencies between services and requirements gathered in the previous section.

Europeana Cloud service	Connected requirements
FRONTEND FUNCTIONAL SERVICES	
Unique Identifier Service	R1
Metadata and Content Service	R2, R3, R4, R7, R5
Notification Service	R4
Data Annotation Service	R6
Data Processing Service	R8, R9
BACKEND SYSTEM SERVICES	
Authentication/Authorization Service	Non- functional requirements, R5
Logging Service	Non-functional requirements
Asynchronous Messaging Service	Non-functional requirements

In the context of the requirements analysis presented in section 3.1, the order of implementation of functional services should be following:

1. Unique Identifiers Service in parallel with Metadata and Content Service
2. Notification Service
3. Data Annotation Service
4. Data Processing Service

The computational cloud and storage cloud will have to be set up (at least in a minimum extent, as a testbed) at the beginning of the development in order to make the deployment environment available for the developed services.

The system services will be also useful from the beginning, but most likely the complete development of these services may be spread throughout the project.

Appendix A. Deployment Options for Europeana Cloud: Report on the Evaluation of Available Cloud Technologies

Report on Europeana Cloud Task 2.1.2

Author & editor:

- Petr Knoth (Open University)

Co-authors:

- Marcin Werla (PSNC)
- Georgios Mamakis (Europeana)
- Markus Muhr (The European Library)
- Pavel Kats (Europeana)

Introduction

This document reports on the work carried out in *Task 2.1.2 - Evaluation of available cloud technologies.* The document feeds into the Architectural Design Document (D2.2) and its purpose is to support the decision making on the instantiation of eCloud. The document does not have itself a deliverable status and therefore is submitted together with D2.2. It provides primarily information for the consequent development in WP2. However, it also serves as an important input for Tasks 5.2 and 5.4.

The document first identifies characteristic properties of cloud computing technologies and comments on their potential benefits for cultural heritage organisations in three main areas: Cost management (utility billing, predictability, adaption of costs), Risk & Quality (system availability, backup & disaster recovery, elasticity & scalability, security) and Control & Flexibility (organisational focus, maintenance & updates, time to market, portability & interoperability). The report also explores the available options for the deployment of the Europeana Cloud (eCloud) infrastructure and provides an analysis of their pros and cons. The available options are primarily classified according to how and where computational and storage resources are allocated and maintained as this is one of the key factors largely influencing: (a) to which extent will be Europeana and its aggregators able to benefit from cloud technologies (such as the elasticity of storage, failover or on demand allocation of resources) and (b) the economic model eCloud partners will use to control expenditure on computational resources.[7]

The document is structured into the following sections. In Section 1, we present a list of characteristic properties and benefits cloud computing can bring to the cultural heritage sector and discuss the challenges that should be considered when deciding which deployment option should be selected. Section 2 then lists some of the available options in terms of the types of cloud offerings eCloud can provide, the way how eCloud could be operationalized and finally provides scenarios regarding how it could be technically deployed. To enable the evaluation of strategies outlined in Section 2, Section 3 summarises the results of a cloud computing survey which was carried out with relevant organisations in the domain that could benefit from eCloud. Section 4 lists some of the notable cloud computing technologies that can be utilised in eCloud development. The pros and cons of the deployment options introduced in Section 2 are then discussed using the gathered evidence in Section 5. This section also provides a set of recommendations for the development and deployment of eCloud.

[7] The economic model will be discussed in a separate document and is part of WP5

1. Characteristic Properties of Cloud Computing to be Considered

The following list provides a set of characteristic properties, benefits and challenges of utilising cloud computing in the context of the eCloud project. The extent to which these properties can be achieved and offered by eCloud to its users is largely influenced by the cloud service model, cloud type and deployment options described in Section 2. The aim of this section, which was drafted and circulated prior to the rest of this document, was to list these characteristics to inform the discussions within WP2 and beyond regarding the assessment of available technology and architecture options. At the same time, the characteristics are important to understand the differences and implications of building the eCloud system on top of existing third party cloud services or using own (on premise) hardware resources.

1.1. Cost Management

Improved cost management of computational resources is at the core of cloud computing. Cloud computing can offer effective utilisation and allocation of computational resources when they are needed, which has the potential to drive costs down.

1.1.1. Utility billing/pay as you go

Description. Utility billing or pay as you go is one of the defining concepts of cloud computing. One can argue that considering computational resources as a utility (like water, electricity, etc.) is the single aspect that makes cloud computing novel, as the other core concepts behind cloud computing, such as failover, high availability or the distributive nature, have been around for quite some time.

Opportunities for the Europeana Network. The Europeana Cloud consortium can establish cloud services for itself and its aggregators (later just Europeana partners) with very low initial investment into computational resources literally avoiding all sunk costs if its services will be fully built on top of existing cloud infrastructures of third party providers. The utility billing/pay as you go model significantly reduces costs, should the service not be able to make effective use of its idle computational power. It also reduces the risks of not being able to fulfil demand if the service becomes suddenly very successful and provides a cost-effective model if the service needs increased computational resources during peak times.

Utility billing/pay as you go might be an attractive solution for eCloud at this point as there are currently little on premise computational resources available at the partnering institutions and thus moving to this model would not imply a large waste of already existing investment.

1.1.2. Predictability of costs

Description. The utility billing/pay as you go payment model holds the promise to make it easier to see, predict and control costs for computational resources regardless of the usage demands on the service. Spending on storage & computational power in this way should be more transparent as all costs including maintenance, backup, software licenses and upgrades are included in the charges and can be more easily tracked and predicted even if the rate of growth for the service is not clear. However, some organisations can have problems with adopting the utility payment model, specifically if their budget is not directly linked to the usage of their service (for example libraries or universities). Such organisations are often used to the strategy of buying the best hardware they can afford for a given cost. On the other hand, utility payments seem to improve budget control when organisations are spending a certain percentage of their budget on IT and their available budget is directly linked to the usage of the service.

Opportunities for the Europeana Network. Building infrastructure in a pay as you go environment, might be an attractive low risk option for a new service, such as eCloud, as it would ease the development of a realistic business plan which could pass the real computational and storage costs for running the service on the users (aggregators) also in a transparent pay as you go way. This solution can dramatically reduce financial risks associated with offering the eCloud services at the time when the scale of the computational resources to be needed is yet to be known. On the other hand, should the eCloud budget not be directly linked to the usage of the service, this option might actually not be the most optimal and lead to higher cost unpredictability.

1.1.3. Adapting costs to current needs

Description. The utility billing/pay as you go payment model makes it easier to increase (upscale) or decrease (downscale) the available computational and storage resources according to the current needs of organisations. It makes it also less difficult to carry out substantial changes in the architecture design of the solution as there are no costs, delays and risks associated with decommissioning or ordering of new hardware. This makes it possible for an organisation to operate in a more agile way.

Opportunities for the Europeana Network. The potential ability to modify the underlying hardware infrastructure if needed (if building eCloud on top of existing cloud services) can be an appealing feature for the eCloud project in its early stages when predicting the hardware type and parameters that would otherwise have to be bought can be very difficult. Such investment would also likely constitute a large sunk cost with no guarantee the hardware will be effectively utilised or meet the demand. Opting for a pay as you go option can give the eCloud project more flexibility and avoid possibly difficult decisions regarding the hardware infrastructure changes in the future.

1.2. Risk & Quality

Relying on cloud computing technologies can be used to shift some risks of an organisation to the cloud service provider. The principle idea is that the service provider manages a very large amount of resources, which are effectively utilised, backed-up, maintained, etc., and the organisation would not be able to provide such quality of service or the service would cost more. Another reason for organisations to shift risks towards the service provider is that the contractual nature of the service provision can allow the organisation to claim indemnities in case of failure and consequently reduce its losses. The following characteristics related to risk should be considered when designing the eCloud architecture.

1.2.1. High availability ("no single-point-of-failure")

Description. *Availability* refers to the ability of the user community to access and use a system. Availability can be measured as the fraction of the time the system can be used over a certain period. *High availability* refers to the approach to the design of a system that ensures that a certain level of operational performance will be met. Such design typical creates redundancies, which can be leveraged in case part of a system fails, and avoids *single points of failure*, i.e. parts of a system failure of which results to a failure of the entire system. While it is often difficult for an organisation (especially if it is a small business) to achieve high availability on its own, cloud computing service providers typically design their infrastructures so that they offer high availability.

Opportunities for the Europeana Network. The promise of high availability can be essential for the success of eCloud, particularly when eCloud starts offering its services to many content providers. To rely on eCloud, these providers will likely require some level of availability. It might be difficult or too expensive for the current or potential eCloud stakeholders to provide such high availability on their own and therefore the promise of high availability can potentially be in some cases even an incentive to move storage and functionality to the cloud environment. If eCloud offers high availability, various stakeholders might be able to start relying on the eCloud infrastructure and possibly save significant on premise resources. At the same time, it seems that if eCloud cannot offer high availability, it might be difficult for the Europeana partners to start fully relying on this infrastructure. This could result either in low adoption of eCloud or possibly increase of the total infrastructure costs (both eCloud and local services of content providers would need to be offered at the same time).

1.2.2. Backup, disaster recovery

Description. Data are today a critical asset of organisations. Cloud computing promises high backup reliability (data safety) and faster disaster recovery times at lower costs as shown in the following figure.

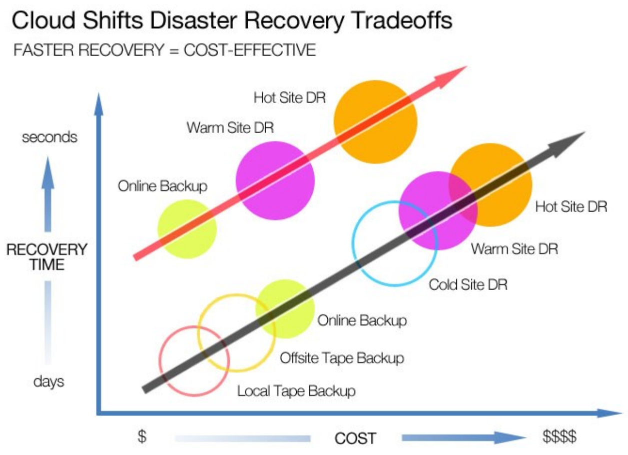


Figure A.1. The red line shows how cloud computing makes disaster recovery more affordable [source: http://www.onlinetech.com/resources/e-tips/disaster-recovery/benefits-of-disaster-recovery-in-cloud-computing]

Opportunities for the Europeana Network. Cloud based backup and disaster recovery provides for Europeana partners the opportunity to effectively and cost-efficiently reduce the risks of losing data in case of a disaster or compromise. eCloud can, for example, deploy software on its infrastructure that ensures the data of all Europeana partners are replicated across multiple data centres and provide support for its recovery in case of a failure at any partnering organisation. Alternatively, the eCloud architecture does not necessarily have to implement backup on its own, but can depend on an existing backup service provider. In this way, eCloud could provide highly reliable backup and disaster recovery for all partnering organisations with a relatively low initial investment at the partnering organisations.

1.2.3. Elasticity, scalability

Description. One of the key characteristics of cloud computing is its on-demand elasticity and scalability, which allows organisations to quickly dedicate more or less storage or computational resources to their services. This removes from an organisation the risk of under provisioning, i.e. not being able to fulfil demand, for example at peak times, and also reduces the risk that an organisation pays more for hardware resources than necessary. Finally, on-demand elasticity and scalability increase the agility of the organisation by making it possible to quickly grow the customer base. For example, there are cases where resource demand has spiked ten-fold overnight only to fall back to its original level afterward.[8] Therefore, if an organisation needs to make an order for a server when more resources are needed to meet demand, it might be hard to refer to this model as cloud computing.

Opportunities for the Europeana Network. The cloud elasticity and scalability hold the promise to make it easier for eCloud to import data of any size and support the joining of new partner organisations

[8] Animoto is a popular example: They scaled from 50 Amazon Servers to 3,500 servers in three days (16-19 April 2008).

at any time, since there are no theoretical limits to the availability of resources. The eCloud software must be horizontally extensible, i.e. it must support the addition of new commodity hardware or storage and computational resources provided by a third party at any time. In case some data are removed from the cloud, the infrastructure becomes immediately cheaper to run, thus this removes the risk of maintaining too much resources as well as the risk of not being able to meet demand. Elasticity and scalability are also one of the few cloud characteristics that are specifically mentioned in the DoW.

1.2.4. Security

Description. While security is usually and rightly seen as a challenge for cloud computing, cloud computing also provides some inherent benefits. The level of security depends on the practices of the cloud service provider. If the cloud will store sensitive data, it is therefore of high importance to use only trustworthy suppliers. However, the suppliers have often deployed rigorous processes to ensure the safety of the data. They often use latest security solutions and ensure professional and timely installation of software security updates. There is also an arguable benefit of the cloud, because virtual software images can move anywhere in the cloud which makes it more difficult for a hacker to launch an attack.

Opportunities for the Europeana Network. It is essential for eCloud to be seen by Europeana partners as a trustworthy service where they can upload both open and copyrighted content with restricted permissions. The opportunity for eCloud is that with the backing of the eCloud project, there is now a real possibility of establishing a service with high reputation. However, the architectural design, the quality assurance and maintenance processes must be set to minimise any copyright breach risks, which could easily undermine this trust. Some of the crucial decisions that need to be made are related to responsibilities for QA and platform maintenance. Since security updates are often best and timely installed by professionals in this area (who Europeana might not have) it is a question whether the desired level of security should be achieved by Europeana partners themselves or rather by making use of an elastic storage of a highly trustworthy vendor.

1.3. Control/Flexibility

Cloud infrastructure can add a considerable amount of flexibility to an organisation and ease its control. The following cloud infrastructure characteristics related to control and flexibility should be considered in the architecture design.

1.3.1. Organisational focus

Description. Once services are outsourced to a cloud, the IT experts within the organisation should be able to better focus on improving the core business processes of the organisation, because less or even no effort is required to perform (often unexpected) technical maintenance, provide user support or perform security updates. The overall strategy of an organisation can be to outsource services that do not require in-house knowledge, can be provided with the same or higher quality outside of the organisation or are more cost-effective when not run within the organisation.

Opportunities for the Europeana Network. The opportunity that expert staff could better focus and plan work that is at the heart of the organisation seems especially attractive for knowledge-intensive and innovative organisations that employ highly specialised staff, such as Europeana. If an organisation employs people with a unique skill set and deep knowledge of the organisation, it is wasteful to require them performing tasks that do not need this organisational insight, are outside of their area of expertise or restrict them in any way from exploiting their unique skills to the benefit of the organisation. Outsourcing a specific well-defined set of services, such as backup, server maintenance or storage, can help an organisation to exploit the skills of its staff more effectively. In the case of organisations that are largely funded from fixed-term projects with high staff turnover, outsourcing a specific set of services might even constitute an effective strategy of ensuring cost-effective, highly available and good quality solutions are sustained beyond the project funding and work on future projects does not interfere with maintenance of existing services. Building a specific set of eCloud services on the reliable infrastructure of existing third party cloud providers should therefore be considered as an option for reducing the costs for running eCloud.

1.3.2. Maintenance and infrastructure updates

Description. Infrastructure updates, such as hardware replacements and software & security patches, are often best performed by technicians with deep understanding of their nature. An organisation might not have such individuals (or it would be too expensive to acquire them) and can therefore benefit from (a) 3rd party technical support or (b) 3rd party maintenance of storage and computational resources.

Opportunities for the Europeana Network. There is an opportunity for aggregators joining eCloud to benefit from decreased maintenance requirements, as maintenance can be carried out centrally by the eCloud community or technology partners (eCloud providers) or even completely on their behalf by a third party provider. The eCloud providers can also control which maintenance tasks should be left to them and which maintenance tasks should be outsourced to a third party to best fit eCloud needs.

1.3.3. Decreased time to market

Description. The time in which software updates, security patches or other kinds of maintenance need to be performed and their scale are often difficult to predict. Consequently, if these can be largely outsourced, the ability of the IT experts to plan the development and deployment of new or improved service can be greatly enhanced. If the overall time from design to market decreases, the organisation can more timely respond to new user requirements, potentially leading to more satisfied users and better market position.

Opportunities for the Europeana Network. Although it seems that eCloud currently does not have a direct competitor on the market (as it aims to serve a relatively specific user group by offering a specific set of services), the benefits provided by decreased time to market can have for Europeana partners significant financial implications and can even determine the success of eCloud. For example, if several eCloud users (content aggregators) require a new feature to be implemented, such as support for expressing content relationship information in a new format, there must be a transparent business process in place which enables the individual eCloud users to decide if they want to wait for that feature to be developed or do something else. Once eCloud commits to the development of that feature in a certain time period, it should be delivered on time, as it otherwise undermines the confidence of users in eCloud. It is therefore crucial that the eCloud technical support team is able to reliably plan activities and that unexpected development is minimised.

1.3.4. Portability/interoperability

Description. Cloud computing should ideally allow an organisation to switch service providers in a similar way one can change a broadband or electricity supplier. While this sounds extremely desirable, it is often not that easy as the software that runs at a site of one provider might not be easily portable to the infrastructure of another. A factor restricting one's ability to move to a different provider is also the type of the cloud infrastructure, such as PaaS or IaaS. For example, it is unlikely that a software solution built using Google Web Services (PaaS) would be straightforwardly portable to Amazon EC2 (IaaS). Cloud computing, still needs some more time to solve its portability and interoperability problems. It is therefore crucial for organisations to think carefully about how they will deploy their solution and understand the differences in the available options to avoid vendor lock-in.

Opportunities for the Europeana Network. The ability to switch providers of cloud services can be important in the future to ensure eCloud can always benefit from a cost-effective cloud solution. It would be wise to ensure eCloud is designed to minimise the risk of vendor lock-in already in the architecture design stage. It a question whether eCloud should also offer support for its aggregators and potential new users to both move data in and move data out of eCloud. While moving data out of eCloud could be seen by many as something not entirely desirable, it can be an important feature for the joining institutions, who might otherwise be afraid of fully relying on eCloud in its early existence.

2. Available Options

This section first introduces the basic *cloud service models* (Section 2.1), explaining the types of cloud offerings that can potentially be provided or built upon, and the *cloud types* (Section 2.2), describing the possible ways in which eCloud could be operated. Finally, the section provides a set of scenarios explaining how eCloud could be technically deployed (Section 2.3). The purpose of this section is not to provide a complete list of all possible platforms and service options (as there are virtually infinite possibilities), but rather to provide a set of representative scenarios, which should be considered and can even be combined by eCloud. The pros and cons of each option are discussed with respect to the characteristics described in Section 1 and are summarised in Section 5 together with the Recommendations.

2.1. Cloud Service Models

The following four cloud service models are important in this context for two reasons:

a) *As a component to rely on.* eCloud can utilise economies of scale of a third party provider, offering one of these service models, to deliver its own services.
b) *As an offering to be provided to users.* It is essential to make a decision about the service models eCloud will deliver to its users.

2.1.1. Infrastructure as a Service (IaaS)

The (virtual) servers of an *elastic infrastructure* are offered to different users that are isolated from each other on a pay-per-use basis.

To deploy applications, cloud users typically install and maintain operating-system images and their application software on the cloud infrastructure.

Example IaaS components eCloud could rely on: IaaS storage, such as Amazon S3 or Amazon Glacier or compute servers, such as Amazon EC2.

Example IaaS offerings eCloud plans to provide: Metadata and content storage services

2.1.2. Platform as a Service (PaaS)

A middleware platform is offered to different users that are isolated from each other on a pay-per-use basis. An API allows users to deploy software components to a *Platform as a Service* offering, register and configure other platform services for communication, such as message queues, storage and routing.

In the PaaS model, cloud providers deliver a computing platform, typically including operating system, programming language execution environment, database, and web server. Application developers can develop and run their software solutions on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers. With some PaaS offers, the underlying computer and storage resources scale automatically to match application demand so that the cloud user does not have to allocate resources manually.

Example PaaS components eCloud could rely on: Google Web Services

Example offerings eCloud plans to provide as PaaS: N/A

2.1.3. Software as a Service (SaaS)

Software is offered to different users that are isolated from each other on a pay-per-use basis.

Users are provided access to application software and databases. Cloud providers manage the infrastructure and platforms that run the applications. SaaS is sometimes referred to as "on-demand software" and is usually priced on a pay-per-use basis. SaaS providers generally price applications using a subscription fee. In the SaaS model, cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients. Cloud users do not manage the cloud infrastructure and platform where the application runs.

Example SaaS components eCloud could rely on: Hosting of indexing services (hosted SOLR, ElasticSearch).

Example offerings eCloud plans to provide as SaaS: The Europeana Research platform.

2.1.4. Community as a Service (CaaS)

Different provider supplied services are offered to users that are isolated from each other on a pay-per-use basis. These users are enabled to create individual compositions of the provider supplied services to meet their functional and service level requirements. Users may create custom compositions of provider supplied services residing on the software, platform, or infrastructure layer. These customizations are again hosted by the provider.

Example CaaS components eCloud could rely on: N/A

Example offerings eCloud plans to provide as CaaS: For example, eCloud could allow each organisation to develop and share new services within the community. The eCloud infrastructure could then provide a layer that would allow users to define workflows/easily and create compositions of these services. This approach would allow more computational flexibility in the cloud for the partner organisations than offered by SaaS.

2.2. Cloud Types

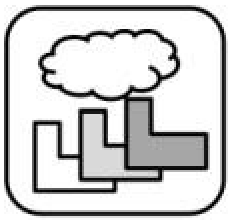
To decide how eCloud will be operationalized, the consortium should select one (or a combination) of the following cloud types. This decision has a major impact on both the organisation and implementation of the eCloud services and therefore each of the scenarios resulting from adoption of one model should be considered.

2.2.1. Private Cloud

Private Cloud is established in a dedicated data centre by an organisation itself or by an external provider. The services offered by this cloud are only accessible by one company.

Scenario of adoption by the Europeana Network: Europeana partners would decide to form an organisation/consortium that would build and run the newly developed private cloud infrastructure. Hardware would be owned by the consortium and would be hosted in an agreed data centre (possibly of a third party provider).

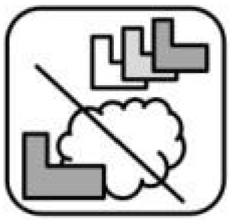
2.2.2. Public Cloud



Public cloud is established by leveraging economies of scale of a third party provider to allow an organisation to use resources dynamically while levelling the utilization of their static physical data centres. Additional security mechanisms are implemented to isolate organisations from each other.

Scenario of adoption by the Europeana Network: Europeana partners would deploy the services to be developed in the eCloud project in the public infrastructure of existing providers. No existing storage or computational resources of Europeana partners would be utilised to run the cloud infrastructure.

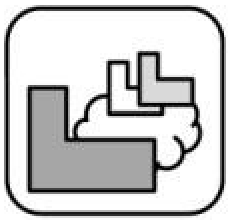
2.2.3. Hybrid Cloud



Private and *public clouds* can be integrated to form a *hybrid cloud*. Management functionality ensures that the disadvantages of private clouds (less elasticity) and public cloud (less privacy, security, and trust) are reduced by migrating workload between the environments.

Scenario of adoption by the Europeana Network: The offered solution would balance the benefits of both the public and the private cloud. It would be possible to decide which services and when run from the public component and vice versa.

2.2.4. Community Cloud



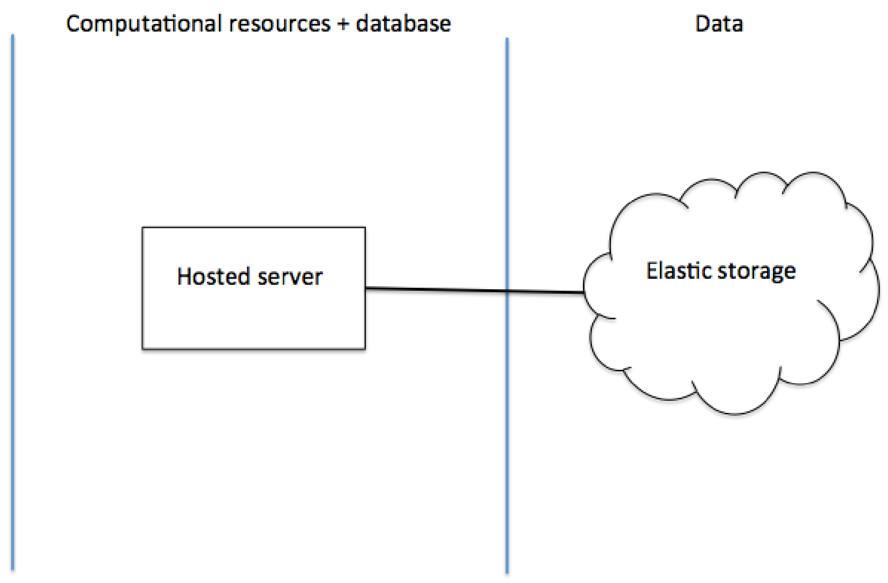
Community cloud is established containing computing resources that can be accessed by all business partners. Community cloud is typically used when a single or a combination of *aaS offerings shall be shared between multiple companies that trust each other to reduce costs of the offerings and enable a dynamic resource usage and sharing.

Scenario of adoption by the Europeana Network: To build the shared cloud infrastructure, Europeana partners would utilise the resources of a number of trusted partners. Each partner could be (a) delivering the same offerings (services are replicated) or (b) delivering a set of unique offerings together forming the community cloud infrastructure.

2.3. Deployment Options

This section aims to provide a set of possible scenarios that should be considered in the design stage of eCloud. The pros and cons of each option are discussed with respect to the characteristics described in Section 1. In reality, the final solution can possibly be the result of combining multiple options.

2.3.1. Option 1 - Hosted server with IaaS storage ("storage only cloud")



Key points:

- Utilise an existing elastic storage of an infrastructure provider (e.g. Amazon S3)
- Make use of a dedicated application server hosting solution (on or off premise).

Advantages:

- Simple to set up, run and maintain.
- Low risk in terms of implementation
- Satisfies the eCloud DoW requirements

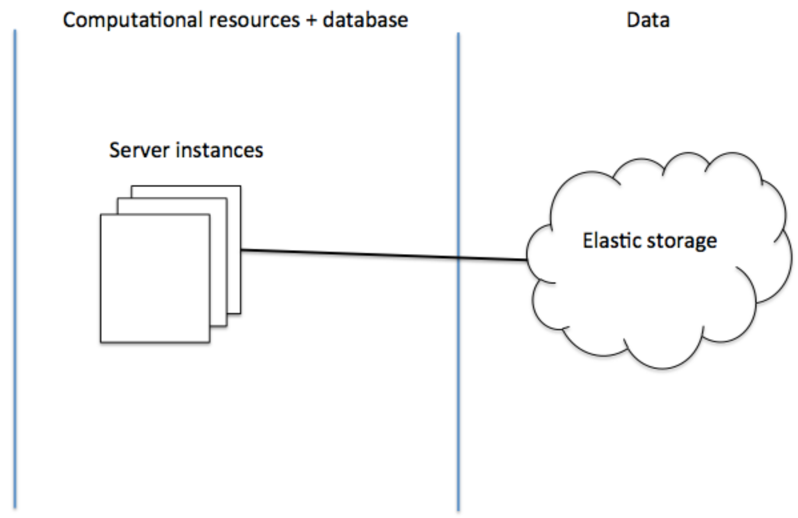
Disadvantages:

- Dependent on the availability of the host's infrastructure (i.e. no replication of computational resources)
- Might not scale in terms of computational resources to eCloud needs (no horizontal scalability and inherent elasticity of computational resources).

Example of application in the context of eCloud:

One of the eCloud partners (or the consortium) would host a server or a set of servers in their server room or would pay for hosting of a server to a third party company. These computational resources would run the software managing the cloud storage and the Europeana Research platform. There would be no inherent provisions for horizontal scalability of computational resources. However, the cloud would be elastic in terms of storage capacity.

2.3.2. Option 2 - Dedicated IaaS ("IaaS cloud")



Key points:

- Make use of (on-demand) server instances of an infrastructure service provider (such as Amazon EC2). Typically involve multiple front-end instances with a single or multiple database servers (for example, using the Amazon Block Storage - EBS).
- Backup or store additional data using EBS or S3.

Advantages:

- Easy to move existing architecture in the cloud and out of the cloud.
- Can be auto-scaled to take into account the computational needs.

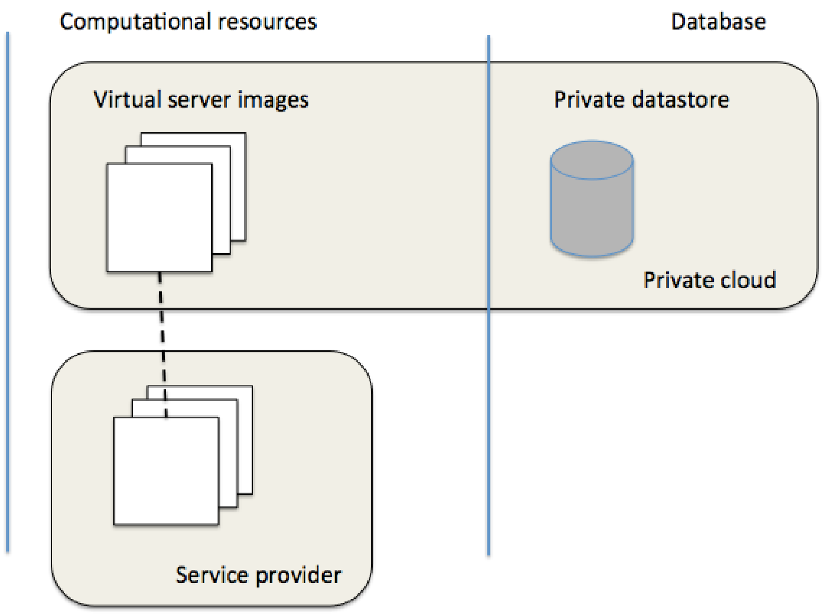
Disadvantages:

- Auto-scaling is more complicated from the perspective of the programmer than with PaaS and is also less responsive.

Example of application in the context of eCloud

eCloud would benefit from the inherent storage elasticity of an infrastructure provider. The software running the Europeana Research platform as well as the services for managing the cloud storage would be running in an environment hosted by an infrastructure provider. In this way, the computational resources would be extensible in an on-demand fashion and the eCloud software would be implemented to support horizontal scalability. Although the time in which more computational resources can be allocated is typically higher with IaaS than with PaaS, there is no indication that eCloud really needs such a high level of computational scalability. This solution might therefore be sufficient.

2.3.3. Option 3 - Private application with IaaS extension ("IaaS Hybrid option")



Key points:

- Satisfy part of the needs on premise, but tap into the cloud to leverage external resources once internal capacity has been exhausted. Can be, for example, achieved by building a private virtualisation infrastructure (VMWare vSphere). Public providers (such as Terremark) provide compatible offerings that can be utilised at peak times, for recovery or improved redundancy. An alternative solution is to utilise the free open source OpenStack technology to develop the IaaS solution that can run on the commodity hardware.

Advantages:

- Can utilise the available private infrastructure, dips into cloud only when needed.
- Can be implemented to be completely independent from the offerings of commercial service providers (if realised using software, such as OpenStack).
- Allows seamless reallocation of computational and storage resources between private and public cloud and migration between different infrastructure providers.

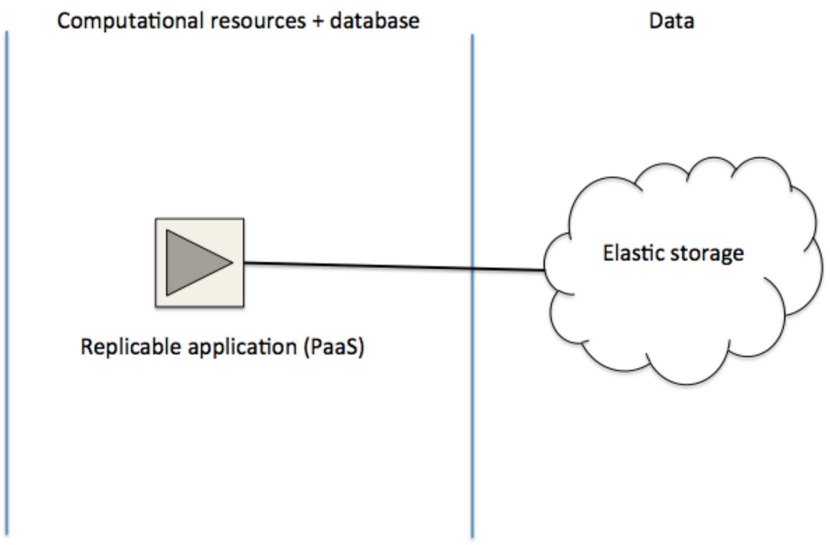
Disadvantages:

- There might be loss of performance in the private cloud due to virtualization, if realised using WMWare, (in comparison to a non-virtualised solution).
- More challenging in terms of implementation than the previous options.
- Higher software maintenance costs than with the previous option.
- If VMWare used, dependent on commercial software that can be quite expensive.

Example of application in the context of eCloud

This solution would build on an existing virtualisation (VMWare) or cloud computing framework (OpenStack). For example, it can be built with OpenStack that would provide the software to run an IaaS cloud even in the on premise environment. The eCloud software would allow the management of the cloud resources and migration between on premise and off premise environments. It would be possible to run eCloud on commodity hardware regardless of the hardware's geographical location. This option would balance the advantages and disadvantages of private and public infrastructure. The underlying infrastructure would be separated from the software product, which would allow migration of the software between infrastructure providers at any time in the future.

2.3.4. Option 4 - PaaS only


Computational resources + database
Data
Elastic storage
Replicable application (PaaS)

Key points:

- Fully based on a cloud-based platform (such as Google Web Services - GWS) and all computation performed there.

Advantages:

- Extremely quick response time due to inherent scalability and elasticity of the platform
- Easy to implement (would only require implementation of the envisaged cloud services and Europeana Research platform and not of the infrastructure itself).
- Very low maintenance costs

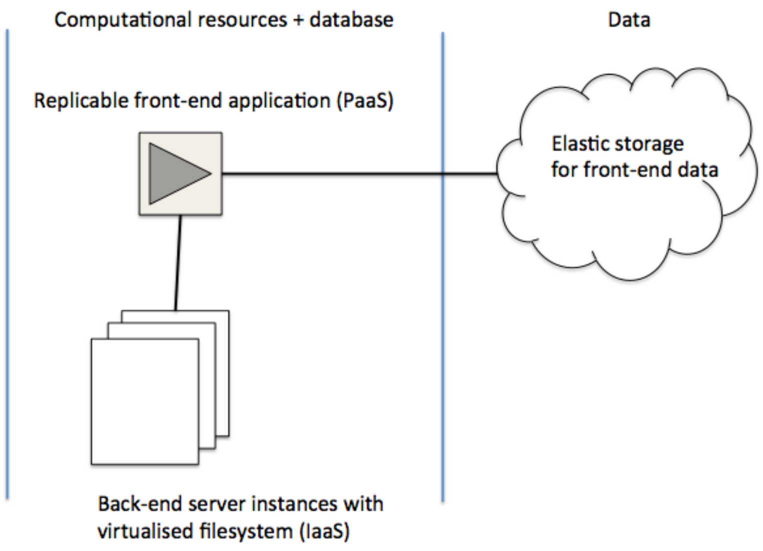
Disadvantages:

- Might be difficult to avoid vendor lock-in
- Less flexibility for back-end operations, which might be a challenge for a developer (currently, Google App Engine requires all processes to complete in 30s).

Example of application in the context of eCloud

The consortium would develop software designed to run in the infrastructure of a specific PaaS provider, such as Google. Storage elasticity would be dependent on the offerings of an IaaS provider.

2.3.5. Option 5 - PaaS application with IaaS storage and computation


Computational resources + database
Data
Replicable front-end application (PaaS)
Elastic storage for front-end data
Back-end server instances with virtualised filesystem (IaaS)

Key points:
- Utilise PaaS for scaling the end-user facing application
- Utilise elastic storage for end-user data
- Utilise IaaS for backend computation
- Backend server instances have a distributed shared file system (such as using Apache Hadoop - HDFS)

Advantages:
- Combines the advantages of PaaS and IaaS
- Extremely high front-end availability and scalability
- Good flexibility from a developer's perspective

Disadvantages:
- Most challenging to implement
- Might be difficult to avoid vendor lock-in for front-end facing applications

Example of application in the context of eCloud
The consortium would develop software to support the functionalities required by aggregators (as described in D2.2) exactly as in Option 2, i.e. would deploy it on off-premise elastic IaaS infrastructure. Alternatively, this option can also be combined with Option 3, in which case the IaaS would be a virtualised infrastructure or a cloud infrastructure developed by the consortium. However, all user interfaces, i.e. primarily the Europeana Research platform, would be designed to run in the infrastructure of a selected PaaS provider. This would provide very high end-user scalability yet good flexibility in development. The risk of vendor lock-in would be minimised only to thin end-user facing clients.

3. Assessing the Options

In order to assess the options outlined in Section 2, we have compiled a questionnaire and carried out interviews with stakeholders. These stakeholders included all the aggregators who are part of the eCloud project (TEL, Europeana and PSNC) as well as potential customers of eCloud outside of the project consortium, i.e. aggregators and libraries that could benefit from the eCloud technology once it is available. The questions were designed to address each of the identified cloud computing benefits identified in (Section 1). The questionnaire is available in the Annex. The goal of the questionnaire was to better understand the needs of these groups and their non-functional requirements on eCloud, so that the adoption of the services once eCloud is developed is smoother. The collected answers serve as a critical input for the decision making about how the eCloud architecture should be instantiated and deployed as each of the options listed in Section 2 have slightly different non-functional properties. The surveyed organisations included all the partner aggregators, namely:

1. TEL
2. Europeana
3. PSNC

plus organisations running aggregation systems that are outside of the eCloud consortium (potential customers):

1. The Open University - The CORE (COnnecting REpositories) Open Access aggregator (CORE) od research papers
2. National Library of Israel (NLI)

3.1. High Availability

The availability of systems and the requirements differ significantly between the surveyed organisations. Only few organisations indicated they are keeping accurate records of availability statistics and have policies in place to guarantee high availability.

The achieved availability among the organisations also differs substantially from unknown to two nines (i.e. 99%). On the high availability side, PSNC reported they might be able to soon extend to four nines (i.e. 99.99%). However, all organisations indicated the availability they currently do provide is sufficient for running of their services. NLI, Europeana and CORE categorise services into several categories with different levels of availability requirements and deploy them accordingly. Consequently, and quite surprisingly, providing high availability seems not to be in most cases the main incentive for aggregators and libraries to move towards the cloud environment. However, CORE mentioned that the availability increase that could be achieved by external replication of computational and storage resource to protect from university network downtime would still be extremely valuable. The minimum requirement from the surveyed organisations to consider using eCloud was between 99% and 99.99%. All participants thought that the service must demonstrate value for money and show convincing evidence of availability guarantees if organisations are expected to pay extra for even higher availability.

3.2. Backup, Disaster Recovery

Both backup and disaster recovery were mentioned by most surveyed institutions as the areas were eCloud could provide real benefits and savings. The identified benefits might be in increased effectiveness, data security and cost reduction. The survey results suggest that organisations often feel they can improve their backup and disaster recovery facilities and drive down their cost. Backup and disaster recovery also seem as good incentives for aggregators to join eCloud. On the other hand, the questionnaire also revealed that some organisations have policies that restrict them from moving their data outside of their country, essentially prohibiting them to use eCloud storage. However, this might not necessarily mean a complete barrier for adoption of eCloud. For example, the National Library of Israel, which has a policy that restricts the geographical location where data are stored, indicated that if data are replicated across different systems, such that there is never a full copy of an object at a single location, this might be a solution to overcome this issue. Europeana is in the position to adopt cloud technologies, but requires sufficient control and transparency over the solution and all parties providing technical infrastructure to be trusted. This suggests that if eCloud should be able to utilise existing storage and computational resources of organisations that join it, according to the community cloud model, there must be set clear criteria on providers. Providers must satisfy certain rules before they can join the cloud and also while staying as partner. The survey also suggested that to satisfy user requirements, eCloud should allow the transfer/backup of content to the cloud on a daily basis. This provides some hints on the throughput requirements.

3.3. Storage & Computational Elasticity and the Risk of Under-provisioning

In terms of storage needs, all surveyed organisations are in most cases able to predict their storage requirements for the upcoming months. This is particularly true, if new storage is required as a consequence of new projects. In these cases, the surveyed organisations are able to plan their needs several months in advance. However, Europeana, for example, mentioned that unplanned storage extension, which typically does not account to very large capacity, is currently very painful as it can take several days until that storage is allocated. Overall, storage elasticity does not seem to be the main incentive for organisations to join eCloud. Yet, the surveyed organisations require eCloud to be able to allocate new storage within 24h (for unplanned extensions up to a specific amount of space) and within days to a maximum of one month (for planned extensions with requirements of over a specific limit).

In terms of computational capacity, a few of the surveyed organisations said they have occasionally problems with under-provisioning, however this has typically no major consequences as it happens on backend systems or there are already plans being implemented to resolve these issues. Europeana said that under-provisioning on their frontend could potentially be harmful to the good reputation of the service and therefore this is an important issue. CORE mentioned that improved and guaranteed scalability of the frontend would be useful as an additional safety measure for the service. The average utilisation of the services was to the surveyed organisations either unknown or was somewhere between 20%-50%, suggesting a reasonably good opportunity for providing cost and computational capacity benefits by adopting cloud computing.

3.4. Security

The ability to enforce good security practices and the models in which they are applied varies widely between the surveyed organisations. However, all organisations are aware of existing security risks and feel they should work towards improving their current situation. Out of the surveyed organisations, TEL and Europeana rely already today on 3rd party providers to perform security scans and installation of latest patches. The remaining organisations either employ onsite experts or their organisations have larger independent expert teams. Regardless of how and what level of security is provided by each institution, the level of security is given by the security of the weakest link, which the attacker can exploit. Therefore, if eCloud is supposed to be realised as a community cloud, there must be an agreed level of guaranteed security that is professionally enforced across all components of the cloud. The survey also suggested that eCloud must work hard on building trust within the stakeholder community to attract new organisations.

3.5. Flexibility of Service and Extensibility

Flexibility of access to the data in the collection and their maintenance is important for all the surveyed organisations. The suggested route to provide this was by offering Application Interfaces (APIs) conforming to open standards. Since the level of integration of eCloud into the workflow of the surveyed organisations is yet unclear, it was difficult for organisations to comment on the user interfaces support that should be offered. Therefore, the development team should be prepared to revisit the requirements at a later stage. As the workflow of organisations can differ significantly, it might be useful to consider allowing users of eCloud to create their own compositions of eCloud services running within the eCloud infrastructure, as expected by the CaaS cloud service model.

3.6. Helpdesk

All surveyed organisations said that helpdesk is essential to support the operation of the eCloud service. PSNC suggested a two tier system: (a) a public community supported forum with a free online documentation and (b) a paid for dedicated helpdesk support with a short response guarantees. Europeana and TEL feel that for more critical issues a within 24h help should be available while for planned issues a longer response time can be sufficient. While most organisations feel they would be willing to pay for a responsive helpdesk support, Europeana adds that they would not be able to compromise on the level of emergency support, which must be available all the time as part of the package.

3.7. Maintenance

Most organisations feel they can benefit from cloud technologies by decreased maintenance. On the other hand, CORE and PSNC mentioned they already have existing teams that are independent of the development teams. Since it might be organisationally difficult to refocus or replace these teams, the need to shift the responsibilities and refocus staff might be a significant organisational barrier to the adoption of eCloud services. As a result, it is vital that the eCloud consortium is able to clearly articulate the efficiency and effectiveness benefits eCloud can provide. The eCloud business and exploitation plan should therefore address the issues of how organisations in this sector can refocus their existing IT maintenance teams to overcome these barriers to adoption.

3.8. Transparent Pay Model

While most of the surveyed organisations do not have a problem with utility based payments, CORE and PSNC indicated that utility based payment model might not be the best option for project based work, where infrastructure funding beyond the project period would be expected or where it would be difficult to estimate the maximum costs that can be incurred in this way. On the other hand, utility based payments are not considered as an obstacle by Europeana, TEL and NLI as they expect to have lower budget fluctuations in the future and feel this might give them an opportunity to operate more cost-effectively. In terms of cost transparency, organisations feel that cost estimations in the range of +-20% would be sufficient in most cases. No organisation required a completely transparent cost policy with respect to all participating users. This means that costs could potentially be adjusted with respect to the market position of each user, not only their usage.

4. The List of Notable Cloud Technologies

It would be out of the scope of this report to evaluate and compare all existing cloud technologies. There has been a considerable number of books and a plenitude of online articles already published on this topic. Therefore, rather than repeating existing work, we identified a set of leading cloud technologies, which can be used to support the development and deployment of infrastructures described in Section 2. In this section, we introduce them and comment on their suitability in the context of the eCloud project.

4.1. Public Cloud Services

4.1.1. Compute services

Amazon Elastic Compute Cloud (**EC2**) is a solution that enables users to rent virtual servers and deploy on them their own services. Amazon provides with EC2 a set of services that make it possible for the user to manage their virtual servers (instances) within the Amazon cloud. EC2 makes it easy to upscale or downscale at any time, paying only for the resources and duration instances are running, and provides support for deploying highly available solutions with redundancy. Amazon is with EC2 a clear leader in the IaaS field for computing resources. Other important players in this field are mentioned, for example, in the following list: http://www.clouds360.com/iaas.php.
EC2 is clearly one of the public IaaS offerings to be tested should eCloud rely on public cloud infrastructures. There are no financial and vendor lock-in risks associated with deploying the solution in Amazon. However, other vendors provide similar offerings, often comparing themselves to Amazon who became “the one to beat” in this sector. Therefore, a reasonable option for the eCloud project seems to be to deploy software on the Amazon infrastructure, but ensure it can be easily migrated to infrastructures of other vendors if needed. The deployment costs of these providers can then be monitored on an on-going basis to ensure the eCloud project is receiving good value for money.

Google App Engine is a PaaS offering for developing and hosting web applications. App Engine allows developers to deploy applications in Java, Python, PHP and GO. Applications can rely on the high-performance BigTable key-value store, which used to be the only database option, but now Google offers also an SQL database. A key feature of App Engine is that it automatically scales applications based on the number of incoming requests. This provides extremely good performance and inherent application scalability without the need of a developer to allocate servers or think about load balancing. The disadvantage of App Engine is that applications are typically not portable to other platforms. There are some projects at various degrees of maturity, such as AppScale or TyphoonAE, that try to overcome this issue, but none of them are at the point where installing and running an App Engine application is as simple as it is on Google's service. Other PaaS providers are listed in the following list: http://www.clouds360.com/paas.php. A notable one is **AWS Elastic Beanstalk,** which largely avoids the vendor lock-in issues of Google App Engine. Unfortunately, this technology is still in beta only.
While PaaS platforms provide many benefits over IaaS, it seems that for the purposes of the eCloud project, the vendor lock-in issue is logically a substantial barrier to their adoption. Besides that, the flexibility of PaaS is often quite limited, which might make the adoption of some technologies more difficult and is also more risky in terms of potential future needs. The one application to watch in the PaaS space is the Amazon Elastic Beanstalk, which might be potentially a suitable technology for the development of the eCloud front-end systems once it makes it to the production stage.

Windows Azure started as a PaaS platform for developing and hosting applications primarily developed using Microsoft technologies, such as ASP.NET, or relying on Microsoft’s infrastructure, such as Microsoft SQL Server. However, the platform also supports programming languages, such as PHP. Recently, Windows Azure started offering also IaaS where it is in direct competition with

Amazon EC2. Windows Azure is one of the three mighty cloud computing platforms provided by Amazon, Google and Microsoft.
Windows Azure seems to be positioned to attract mostly customers used to Microsoft technologies, which Europeana and its partners clearly are not. However, one of the Microsoft technologies to watch is the IaaS Windows Azure Virtual Machines infrastructure.

4.1.2. Storage services

Amazon provides three notable public cloud storage services designed for different purposes: EBS, S3 and Glacier.

Amazon Elastic Block Storage (EBS) is designed for the lowest latency and can be directly mounted to EC2 instances. EBS is expected to be used for storing data or running databases on EC2 instances. EBS can create storage volumes of up to 1TB.

Amazon Simple Storage Service (S3) is an elastic web store the data of which are available through HTTP/HTTPS. It has a slightly higher latency than EBS and is intended for storing data of any size.

Amazon Glacier is a low-cost storage that provides secure and durable storage for data archiving and backup. It is intended for situations when data is primarily written and less being read. It is claimed to be up to 90% cheaper than S3.
The list of other important vendors providing cloud storage is available here: http://www.clouds360.com/storage.php. However, Amazon is again in this domain a clear market leader.

DuraCloud is a very interesting service in the domain of digital libraries designed for storage and preservation with the aim to achieve maximum availability and durability through true redundancy. This means that data are not only distributed into multiple geographical locations managed by one vendor, but also to infrastructures of different vendors. DuraCloud uses Amazon S3, Amazon Glacier and Rackspace as their underlying storage services.

All of the mentioned services seem to have the potential to be used at some point in eCloud. For example, EBS can be used to support the database systems, Amazon S3 as a file storage for frequently accessible content and Amazon Glacier for the purposes of backup. DuraCloud can be a viable option should extremely high durability would be expected. This option can be actually very much appreciated by cultural heritage organisations.

4.1.3. Other services

In addition to compute and storage services, there is a number of other cloud services on the market that are potentially useful to eCloud. These include databases in the cloud and search in the cloud.
In terms of cloud databases, it seems logical to consider the solutions provided by the particular vendor of the compute cloud, i.e. SimpleDB or Relational Database Service (RDS) in the case of Amazon or BigTable in the case of Google. While in the past, these vendors used to offer mainly NoSQL options, there seems to be a trend now to offer both SQL and NoSQL.

In terms of search in the cloud, there are some interesting offerings on the market including ElasticSearch hosting, for example by Bonsai (https://addons.heroku.com/bonsai), Found (http://www.found.no/) or SearchBox (https://searchbox.io/plans_and_pricing), or the Amazon CloudSearch (http://aws.amazon.com/cloudsearch/pricing/) service. Reviewing these options, search as a service is not cheap, though vendors often guarantee indexes to be stored in main memory (RAM), which is in the case of search very important for achieving optimum performance. However, the biggest issue of the current search as a service offerings is that many of the available solutions do not provide an index storage that would satisfy the requirements of Europeana partners. For example, the best (ultimate) solution from Bonsai costs $900 per month and allows 20GB of indexed data. We know today that the indexes of TEL and OU (CORE) already require each over three times as much space. In this respect, Amazon CloudSearch seems to be a possibly cheaper option suitable also for large indexes. CloudSearch costs are calculated based on data out (retrieved items), data in are free. This makes CloudSearch potentially attractive for large indexes with relatively low amount of retrieved items, which is a description that can fit cultural heritage institutions.

Overall, it should be taken into account that the use of these additional services (both databases and search) might be in some cases slightly counterproductive. For example, it might make it more difficult to migrate the infrastructure to a different provider and can also constitute in some cases higher running costs, though possibly lower maintenance costs, in comparison to installing and running this technology on IaaS by eCloud staff.

4.2. Technologies for Building Private, Hybrid or Community Cloud Infrastructures

4.2.1. Frameworks & software for building cloud infrastructures

There is a plenitude of tools supporting in one way or another the development of cloud infrastructures. While we cannot list all of them, a few of them perhaps dominate the market more than others. In the domain of frameworks for building private or hybrid clouds two projects should certainly be mentioned:

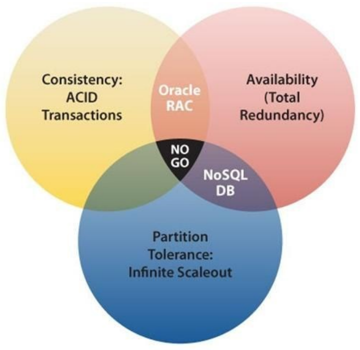
OpenStack – is a free open-source project managed by the OpenStack foundation, a not-for profit entity. The members of this foundation consist of companies, such as Cisco, IBM, NEC, Dell, VMWare and Yahoo! The technology consists of a series of projects that control pools of processing, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering its users to provision resources through a web interface. OpenStack technologies, such as OpenStack Swift (storage) or OpenStack Horizon (dashboard), seem to currently receive a lot of interest by the cloud computing community. One of the winning features of OpenStack is that the software can be out of the box deployed to infrastructures of multiple providers including Amazon EC2, Rackspace and GoGrid. The majority of the competition does not provide this yet (http://en.wikipedia.org/wiki/Cloud_computing_comparison).

VMWare vSphere (vCloud) allows on demand migration of services from internal cloud of cooperating VMware virtual machines to a remote cloud. This is ideal for organisations interested in flexibly utilising the cloud infrastructure of a third party vendor, in addition to on premise hardware resources, to ensure high uptime or overcome peak periods. The goal of the vCloud initiative is to provide the power of cloud computing with the flexibility allowed by virtualization.
Both OpenStack and vCloud are good candidates for technologies on which eCloud could build and both are very suitable for building not only private, but also hybrid or community clouds. However, they both follow quite different strategies. While vCloud builds on the experience of VMWare in virtualisation, OpenStack uses a philosophy of starting from scratch and adhering more elegantly to cloud concepts. For example, VMWare does not offer elastic storage or computation, but sells software, such as the Virtual machine recovery manager. VMWare software has been developed to satisfy acute needs of large companies, while OpenStack software has been developed with the cloud concept in mind. While vCloud is certainly a much larger player than OpenStack, vSphere is a commercial activity while OpenStack is free open-source. This might be an important advantage of OpenStack in the context of the eCloud community, which is very open.

Other well-known tools and frameworks in this filed include **AKKA**, a toolkit and runtime for building highly concurrent, distributed, and fault tolerant event-driven applications in Java, **EUKALYPTUS**, an open source software for building AWS-compatible private and hybrid clouds, or **OpenNebula**, an open-source cloud computing toolkit for managing distributed data center infrastructures to build private, public and hybrid implementations of IaaS.

4.2.2. Distributed databases

One of the critical limitations of distributed databases has been best described by the CAP Theorem. The CAP theorem (CAP standing for *Consistency*, *Availability* and *Partition Tolerance*) states that it is impossible for a distributed system to guarantee all three at the same time[9]:

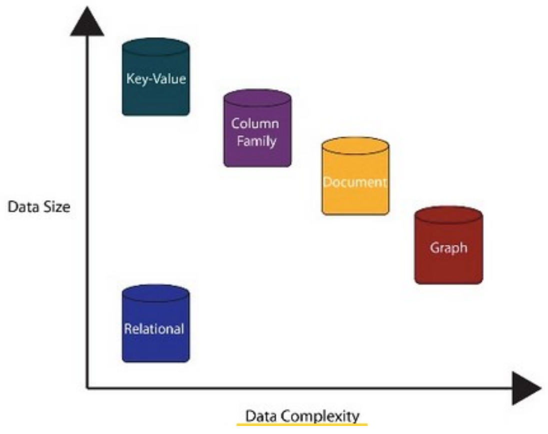

Consistency: ACID Transactions
Oracle RAC
Availability (Total Redundancy)
NO GO
NoSQL DB
Partition Tolerance: Infinite Scaleout

The CAP Theorem

- *Consistency* (all nodes see the same data at the same time)
- *Availability* (a guarantee that every request receives a response about whether it was successful or failed)
- *Partition tolerance* (the system continues to operate despite arbitrary message loss or failure of part of the system)

The introduction of the theorem at the beginning of this century. This led to a huge innovation in the area of distributed databases and the wide uptake of the so-called NoSQL solutions that sacrifice some level of consistency to achieve availability and partition tolerance.

The world of distributed databases provides currently so many good products that it would be for a book to compare them. We will therefore rather mention the main families of these databases and their leading members. As depicted in the following Figure, the key-value solutions offer the highest levels of scalability in terms of data size, but are less suitable for expressing complex data. On the other hand, graph databases, such as triple stores, can express more complex data relationships, but are less scalable.



We believe that suitable options for the development of eCloud can come from the Key-Value, Column family and Document databases. Members of these families with good reputation are:

- Key-value: Redis, Cassandra
- Column: HBASE, Cassandra
- Document: CouchBase, MongoDB

A slightly more difficult to characterise in terms of the figure, but quite a mature solution is MySQL Cluster. MySQL Cluster is a distributed ACID compliant architecture with no single point of failure. Contrary to its name, it offers both SQL and NoSQL. The ability to support both SQL and NoSQL, while having partition tolerance and availability, might be an attractive option especially in cases where migration from existing SQL databases is necessary.

[9] In 2012, it has been shown that reasonable trade-offs between all three are possible, thus it is not necessary to completely sacrifice one out of the three, as it has been often argued by the NoSQL community: http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed

One of the aspects to mention with distributed databases is that most of them typically require operating a several (typically at least 3 to 4) database nodes in each data centre. More precisely, it is not recommended to operate them in a mode where a cluster is formed of several servers each in a different geographical location as this could lead to performance issues (the local network is much faster) and security issues (data sent for node synchronization might not be encrypted). However, this requirement should not be confused with the ability of these databases to replicate to multiple geographical locations, which they mostly support.

Since all the mentioned options are widely used by their communities and it is not clear whether eCloud needs scalability at the level of key-value stores, such as provided by Redis, we recommend to select the database engine to run eCloud based on two criteria: (1) solution maturity (2) preference of the development team.

4.2.3. Storage software

There are two exciting projects providing software for building distributed file systems on the market.

Hadoop Distributed File System (HDFS) – is part of the Apache Hadoop software framework that has been developed for large scale data analytics across clusters of computers. Apache Hadoop is extremely popular in the data mining community as it contains also the well-known MapReduce tool that enables parallel processing of large quantities of data in Hadoop clusters. HDFS has been developed as the underlying distributed storage system for Hadoop. Rather than relying on hardware, the HDFS is capable of recovering from failures at the application layer to achieve high availability. It has also been designed with the aim to achieve very high throughput and was inspired by the proprietary Google File System (GFS), used in the Google search engine. As HDFS cannot be mounted directly by an operating system, the access to the file system is achieved through a Java API.

OpenStack Swift/Cinder – OpenStack provides two types of storage, Cinder is a block storage to be used by OpenStack compute instances, i.e. is conceptually similar to Amazon EBS. Swift is a scalable redundant storage system that scales horizontally across commodity hardware by adding new servers. Data are replicated automatically at a software level like in Hadoop.

The differences between OpenStack and Hadoop are in their communities and usage. While Hadoop caters for the data mining and analytics communities, OpenStack is specifically designed for storage administrators with an aim to cut storage costs, thus OpenStack is popular in the so called *cheap-and-deep* tier storage. Though HDFS can also be potentially used here, it might actually make more sense to deploy Hadoop on top of OpenStack to let OpenStack handle the storage and to enable Hadoop as a tool for data analytics on this storage. This might be a viable solution for eCloud that aims to combine the benefits of cheap storage with the ability to mine the data provided by aggregators.

5. Recommendations

The aim of this section is to summarise the results of the analysis and recommend a suitable strategy for the instantiation of eCloud, including its computational and storage infrastructure. Consequently, this section (together with Section 4) also provides the necessary information for achieving *Milestone 7 - The decision on the use of underlying cloud storage system*. The recommendations take into account, in addition to the analysis carried out as part of Task 2.1.2, the requirements documented in the Architectural Design Document (D2.2), the High Level Principles document (Task 5.1), the discussions that took place during WP2 virtual meetings and the existing experience and expertise of the WP2 team.

5.1. What Cloud Service Models Should Be eCloud Offering?

The eCloud offerings should be delivered at two levels, SaaS and IaaS:

- SaaS services that are important to aggregators are documented in the Architectural Document (D2.2). Additional SaaS offerings focused on the development of research tools will be realised in WP3 later in the project by the development of the Europeana Research platform.
- IaaS services (in the form of the Metadata and Content Service – D2.2) to provide reliable durable storage and potentially assist in disaster recovery to cultural heritage institutions (Section 3). The possibility of offering computational services in eCloud for (a) heavy backend processes on top of the eCloud data and (b) as a prevention against under-provisioning on front-end services should also be considered, but with a lower level of priority and criticality (Section 3). This is not part of the eCloud DoW, though might be useful.

At a later stage in the project, eCloud could also further consider providing the opportunity to its users to create and run composite services in the cloud (CaaS), but this should also not be the priority at this stage.

5.2. How is eCloud Delivering the Cloud Technology?

One of the fundamental, yet difficult to answer, questions is the type of cloud technology the eCloud project is developing. The discussions about the cloud types (Section 2.2) and the deployment options (Section 2.3) as well as the cloud computing questionnaire results revealed a set of preferences the consortium has in terms of the solution. Analysing the most effective ways to satisfy these preferences, we can see there is no option in terms of the cloud type or the deployment option that would be a clear winner. This is not surprising given the fact how quickly evolving and chaotic the current cloud computing market is. Many cloud computing offerings today are extremely difficult (or even impossible) to compare as they represent often ideologically different approaches.
However, to bring some light to the decision making process, we provide a set of recommendations. The aims of these recommendations are to suggest a deployment route that in the current market seems viable, does not inherently carry security or vendor lock-in risks and have the highest potential to be successfully implemented by the consortium and adopted by cultural heritage institutions across Europe.

5.2.1. Cloud type - public ("off premise") vs. private/community ("on premise")

In this section, we will discuss whether eCloud should be deployed on premise, i.e. utilizing its own hardware resources or off-premise utilising public cloud services. For clarity, even solutions that build distributed systems deployed in partners' data centres are in this context classified as on premise, i.e. managed by the consortium.

Among the main advantages of building the eCloud infrastructure on top of an existing public cloud infrastructure are:

- *Simplicity and robustness.* The simplicity and robustness of the solution that could be provided by relying on an already established and large scale provider.
- *Functionality guarantees.* It is less complicated to guarantee the provision of functionality beyond the end of the project as less effort needed to maintain the solution. Therefore, it might also be easier to recruit new customers.
- *Avoiding sunk costs.* Avoiding all sunk costs and thus decreasing the risk of acquiring unsuitable hardware, which is at this stage of the project very high due to the implicit unpredictability of data and computational needs of eCloud. At the same time, the surveyed organisations (Section 3) currently do not have sufficiently large on premise infrastructure that could support a project of this scale, if eCloud is adopted by many European organisations. The current practices of buying hardware in these organisations indicate an understandable preference for vertical rather than horizontal scaling.
- *Business exploitation and cost planning.* Easier to develop sustainable and realistic cost models for selling eCloud services based on usage. Very simple and responsive scalability.
- *Maintenance and security.* Little maintenance costs and professional off premise security management.
- *Organisational focus.* Ability of the development team to focus on the specific services required by cloud aggregators instead of on building server infrastructure.
- *Partner expertise.* Most of the eCloud project partners are experts in digital library technologies and associated areas, such as the semantic web, but there is little expertise in the consortium in building and running large scale data centres with hundreds of computer nodes.
- *Running in a highly reliable infrastructure*. The infrastructure offerings of the main public cloud computing players can provide significantly higher availability than the infrastructure of any of the project partners. However, most of the surveyed organisations are currently happy with their availability (Section 3).

Among the main drawbacks of running on a public cloud infrastructure are:

- *Complete reliance on a commercial provider.* Storing and running public sector services in an infrastructure of a commercial provider is even seen by some as potentially dangerous.
- *Organizational problems.* Many of the partners or potential customers have already some technical infrastructure (even though perhaps not sufficient) and teams responsible for its maintenance. There is some level of resistance to changing this model which would require refocusing of these teams to achieve cost reduction.
- *Security issues.* Some organizations have policies restricting them from storing content at the premises of a commercial provider, however it seems there might often be ways how to overcome these problems for both private and public cloud offerings if the providers are trusted (Section 3).

Among the advantages of private cloud are:

- *In-house security guarantees.* Although public commercial providers can potentially provide very high security, the principle that data are stored in an infrastructure of a third party provider can be perceived as a security threat. After it has been revealed that large cloud service providers were knowingly leaking data to government, this perception of insecurity and lack of privacy of public cloud is today possibly stronger than it was in the past. While this might seem less of a problem to

cultural heritage institutions in terms of the content, the use of cloud resources for storing user data and logs in these infrastructures might raise issues.
- *High degree of flexibility.* There are literally no limits to the development of the cloud in a private infrastructure.
- *Self-reliance and independence.* Being able to rely up to some extent on premise hardware is still important for organizations, though not necessarily most efficient.

Among the main disadvantages of private cloud are:
- *Little expertise and infrastructure.* None of the current partners have currently sufficient hardware infrastructure to run eCloud in their own data center and the necessary expertise to support the infrastructure beyond the project lifetime. On the other hand, there seems to be a high dedication in the consortium to develop this expertise and there are some financial resources available to the project that can be used to acquire a part of the necessary hardware to support such infrastructure.
- *High cost risks of acquiring hardware to run private cloud.* The types and amount of hardware resources to be needed are difficult to predict now and this is likely to only be clearer after the lifetime of the project.

Based on the advantages and disadvantages listed above, it can be seen that while using public cloud seems as a very straightforward way to the realization of the project objectives, there might be a certain resistance to this solution. This resistance is mostly of a strategic rather than technical nature. Consequently, an implementation relying purely on public cloud offerings might constitute a significant barrier to adoption in the future. Therefore, there should be an effort to eliminate this resistance by balancing the solution to retain most of the benefits of both approaches. This can be achieved using the *hybrid cloud option*.

Apart from the properties described above, there is one important variable, which is at this point difficult to estimate - cost. It is yet to be determined whether running the infrastructure privately can generate significant cost benefits. This can only be determined when the human infrastructure needed to support eCloud beyond the project lifetime is established and all costs including staff salaries, electricity, estates, software licenses, cables and broadband are put together and compared to the public cloud offerings.

Recommendation 1: *The eCloud infrastructure should build on the hybrid cloud concept balancing the specific benefits and drawbacks of on premise and off premise resources to fit the needs of the consortium and potential customers. The software should be agnostic with respect to the infrastructure in which it is deployed.*

This recommendation pretty much rules out Deployment Options 4 and 5 (Section 2.3), which utilize PaaS. PaaS typically cannot be deployed in a private environment. On the other hand, the recommendation is compliant with the combination of Deployment Options 2 and 3 (Section 3). In principle, this recommendation also implies that eCloud should be realized at the IaaS level and should have reasonable spec requirements on the individual compute and storage nodes, i.e. must be capable of running on commodity hardware.

Recommendation 2: *eCloud should support the process of allocating and balancing storage and computational resources between public and private cloud environments.*

This suggests that eCloud should be able to decide to run in an on premise environment and dip into the public cloud when needed. By doing so, eCloud gains all benefits of cloud computing related to elasticity & scalability, performance and high availability. It also allows eCloud to flexibly move more or less resources between the off premise and on premise hardware depending on the cost situation, whenever needed. eCloud should develop sufficiently advanced supporting tools to manage such resource reallocation and data transfers.

5.2.2. Cloud type – community (“geographically decentralised”) vs. private (“geographically centralised”)

The Europeana Cloud project benefits from a strong community engagement. While the approach of Europeana related projects in the past was to build a shared centrally managed resource, the eCloud project aims to explore the possible efficiencies that can be achieved in a decentralized infrastructure jointly provided by partners - a community cloud[10].
Running a community cloud compares with a shared private cloud in the following way:

Advantages of private cloud:

- *Easier management and maintenance.*
- *More secure.*
- *Easier to implement.*

Advantages of community cloud:

- *Potential for higher efficiencies and savings.* Can potentially generate higher savings by being able to pool unused resources from multiple data centres.
- *More resistant to failure (higher availability).* Data centres in different geographical locations can provide more effective strategies for replication and recovery.
- *Can ease the cloud technology adoption.* Creates an open environment in which all organizations that join feel as equal partners rather than dependent on a third party.

Recommendation 3. *The on-premise (non-public) part of the eCloud infrastructure should be delivered using a community-based approach. Organisations should be able to install eCloud software in their data centres and allow eCloud to make use of their own computational and storage resources.*

Apart from sharing of resources between partners, this recommendation opens the possibilities for defining a business model that would allow organizations to offset their costs for using eCloud by offering to the community their own computational and storage resources. While the viability of this option will have to be further investigated, the survey documented in Section 3 indicates that such an approach could ease the adoption of eCloud. This is especially true in the case of organizations that have large data centres managed by established IT teams.

5.2.3. Trust

Recommendation 4. *Each partner organisation offering computational or storage resources for eCloud must comply with a set of availability and security requirements specified and regularly updated by the eCloud community.*

This recommendation articulates an essential requirement most organisations joining eCloud will have on the security and reliability of the eCloud service, as indicated by the survey in Section 3. Therefore, each organisation contributing storage and computational resources must satisfy a set of availability and security standards. These must be checked before joining and also periodically revisited to ensure system reliability and security.

[10] We are here referring to the on premise managed part of the hybrid cloud described in the previous section.

5.2.4. Open software & service principles

Recommendation 5. *Dependence of eCloud software on specific commercial services and software should be avoided where possible.*

Many of the eCloud partners endorse the principles of open source development. Building eCloud on top of these technologies creates betters incentives for these communities to contribute to the software development, build a better product and generate new opportunities in the sector. It should also offer more flexibility in terms of the usage of the final product, which can in the future contribute to keeping costs down.

5.2.5. The underlying storage system

Recommendation 6. *The community part of the underlying storage system should be able to utilize cheap commodity hardware and replicate across multiple data centres. To add an extra level of elasticity or improved durability, it should support the integration with cheap IaaS storage.*

In compliance with Recommendation 5, we suggest to use open technologies that are current leaders on the market, such as OpenStack Swift or Apache Hadoop (HDFS) or even their combination to provide the community part of the distributed file system. If more storage than available in the community is needed at any time, eCloud should utilise the storage of IaaS providers, such as Amazon. If the motivation for the storage use is increased durability, back-up or disaster recovery, technologies such as Amazon Glacier should be considered or preferred over standard offerings, such as Amazon S3. This is due to their extremely low costs when low latency is acceptable.
To set-up a reasonably realistic environment for development and testing the partners should seek to acquire cheap but sufficiently large amount of commodity hardware, with at least several machines in each data centre. Such hardware can ideally be resourced from existing sources, such as potentially underused computer study rooms in libraries.

5.2.6. Further investigation and monitoring

Recommendation 7. *Computational and storage costs should be monitored over the project duration.*

Cost of third party (public cloud) storage as well as maintenance of commodity storage to partner organisations must be further investigated and monitored over the duration of the project. As the market is currently very vibrant and unpredictable, the best strategy for acquiring cheap storage and computational resources might well change during the project duration. The consortium should be prepared for such a situation.

5.2.7. Business strategy & supporting services

Recommendation 8. *The business, sustainability and governance model will need to consider the need for establishing a Helpdesk with appropriate responsibilities and service guarantees and a procedure for requesting new eCloud features.*

This recommendation follows directly from the results of the survey in Section 3 where all participants were very clear about these needs. This is according to the DoW the responsibility of WP5, in particular Task 5.4.

6. Conclusions

This document analyzed the technological options the eCloud project has in terms of developing and deploying its cloud computing services. The process of identifying the cloud computing benefits (Section 1), describing the available deployment options (Section 2), the cloud computing survey (Section 3) and the evaluation of suitable cloud computing technologies (Section 4) culminated into the formulation of a set of eight Recommendations (Section 5).

The Recommendations suggest perhaps a challenging to implementation route, however a route with a low risk and high community adoption potential. The suggested solution guarantees, it will be always possible to run eCloud software at reasonable costs and provides virtually unlimited flexibility to the developers in extending the system. The success of this approach will not only depend on the technical implementation, but will be highly influenced by the adoption of appropriate business models and processes that will support the operation of eCloud. A very close collaboration between WP2 and WP5 is therefore essential.

References

John Rhoton. 2013. **Cloud Computing Explained: Implementation Handbook for Enterprises**

John Rhoton. 2011. **Cloud Computing Architected: Solution design Handbook**

Niraj Juneja. **A Walk in the Clouds: Comparing Google AppEngine, Amazon AWS and Sun Project** http://www.slideshare.net/njuneja/cloud-computing-presentation-644830

Renat Khasanshyn. (2012) **Benchmarking Couchbase Server** http://www.couchbase.com/presentations/benchmarking-couchbase

Andrew Oliver. (2012) **Navigating the NoSQL Landscape (Which freaking database should I use!)** http://www.couchbase.com/presentations/navigating-nosql-landscape

Kristof Kovacs. Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs HBase vs Couchbase vs Neo4j vs Hypertable vs ElasticSearch vs Accumulo vs VoltDB vs Scalaris comparison http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis

Akka 2.0: http://doc.akka.io/docs/akka/2.1.1/Akka.pdf

Clouds360.com. **Get Your Head In The Clouds. http://www.clouds360.com/index.php**

Amazon CloudSearch - http://aws.amazon.com/cloudsearch/pricing/

Cloud Computing Patterns. http://cloudcomputingpatterns.org/

ElasticSearch http://www.elasticsearch.org/

GoGrid. **Deploy public, private or dedicated cloud servers in minutes** www.**gogrid**.com/

OpenStack. **Open source software for building private and public clouds.** http://www.openstack.org/

Erik Mitchell. **Library services through cloud computing: Case studies on services, platforms and infrastructure** http://www.slideshare.net/mitcheet/cloud-computing-in-academic-libraries

Apache Hadoop. http://hadoop.apache.org/

Andrew Woods. (2013) **Hosting your services in the cloud - Lessons DuraSpace has learned.** http://or2013.net/sites/or2013.net/files/OR2013CloudPracticesProposal.pdf

Taregowda, P. Urgaonkar, B. and Giles, C. (2010) Cloud Computing: A Digital Libraries Perspective http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5558004

MySQL Cluster http://answers.oreilly.com/topic/1862-what-is-mysql-cluster/

Where would I use MySQL Cluster https://blogs.oracle.com/MySQL/entry/where_would_i_use_mysql

Ambrust et al. (2009) Above the Clouds: A Berkeley View of Cloud Computing http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

Calvo et al. (2010) On distributing load in cloud computing: A real application for very-large image datasets http://www.sciencedirect.com/science/article/pii/S1877050910003017#

Rupesh Sanchati, Gaurav Kulkarni**.** (2011) **Cloud Computing in Digital and University Libraries** http://globaljournals.org/GJCST_Volume11/6-Cloud-Computing-in-Digital-and-University.pdf

Open Nebula http://opennebula.org

Eucalyptus http://www.eucalyptus.com

A File Storage Service on a Cloud Computing Environment for Digital Libraries http://ejournals.bc.edu/ojs/index.php/ital/article/view/1844/pdf

Bold predictions on which NoSQL databases will survive http://www.xaprb.com/blog/2013/01/10/bold-predictions-on-which-nosql-databases-will-survive/

Targeting 200 Million Views A Day And Beyond With Redis http://highscalability.com/blog/2012/4/2/youporn-targeting-200-million-views-a-day-and-beyond.html

Amazon S3 vs EBS. http://www.cloudiquity.com/2009/03/differences-between-s3-and-ebs/

Benefits of Disaster Recovery in Cloud Computing http://www.onlinetech.com/resources/e-tips/disaster-recovery/benefits-of-disaster-recoverx_y-in-cloud-computing

Artem Livshits. (2012) **MySQL on S3: performance with storage located across the continent** http://www.oblaksoft.com/mysql-on-s3-performance-with-storage-across-continent/

Sebastian Stadil. (2013) **By the numbers: How Google Compute Engine stacks up to Amazon EC2** http://gigaom.com/2013/03/15/by-the-numbers-how-google-compute-engine-stacks-up-to-amazon-ec2/

Hadoop and multiple data centres - http://www.datastax.com/wp-content/uploads/2012/09/WP-DataStax-MultiDC.pdf

Amar Kapadia. (2012) **The Significance of Hadoop running on OpenStack Swift** http://www.buildcloudstorage.com/2012/07/significance-of-hadoop-running-on.html

couchbase evaluation http://www.couchbase.com/presentations/benchmarking-couchbase

A File Storage Service on a Cloud Computing Environment for Digital Libraries http://ejournals.bc.edu/ojs/index.php/ital/article/view/1844/pdf

Annex 1. The Cloud Computing Requirements Questionnaire

The aim of this questionnaire is to understand the needs and requirements of organisations and the specific conditions under which they would consider using eCloud services. The goal of this research is also to identify the types of cloud services that can provide the main benefits to the sector.

1. **High availability**
   - How important is for your organisation high-availability (i.e. that systems do not go down and are always available)?
   - Does your organisation have a specific policy or requirement on availability of service (i.e. how many nines, i.e. 99.9…9%)?
   - What level of service availability is your organisation theoretically able to provide and what level does it provide now?
   - If eCloud offered higher availability than your organisation could provide, would that be a reason to consider porting your infrastructure to eCloud?
   - What would be the minimum guaranteed availability for you to consider using eCloud?
   - Would you consider paying extra, if eCloud could offer even higher availability (e.g. five nines instead of three nines)?

2. **Backup, disaster recovery**
   - Would it be practical/useful for you, if eCloud could provide a reliable back-up and disaster recovery?
   - Does your organisation have any policies or restrictions regarding back-up and disaster recovery?
   - Under which conditions would you consider relying on eCloud for backup?
   - Would you be interested in using eCloud for disaster recovery?
   - Under which conditions would you consider relying on eCloud for disaster recovery?
   - Do you have specific requirements on eCloud in terms of backup facilities?
   - Do you think there is a potential for cost savings at your organisation that would be achieved by upgrading back-up and disaster recovery procedures?

3. **Elasticity**

   a. **Storage**
   - Does your organisation struggle with lack of data storage, its maintenance, decommissioning, etc.?
   - Would you consider a service that offers elastic, always available storage for your content useful?
   - How important would it be for you to make sure that eCloud can always accommodate your new data (i.e. there are no space limits or restrictions)
   - How do you feel about having to wait a few hours, days or months in case you need more storage? Is that a possibility or is immediate on-demand elasticity a requirement?
   - What would be your maximum time requirement on the allocation of new storage (under which you would still consider using it)?

   b. **Computational**
   - Does your organisation often or sometimes suffer from under-provisioning (i.e. your systems are unable to meet user demand during peak times or specific events)?
   - What would be the consequences for your organisation if you could not/cannot meet demand?
   - Are your systems always utilised 100%. How much redundancy do you have?

- How much extra hardware resources do you/would you need to have to ensure your systems do not suffer from under-provisioning (estimate in percent with respect to existing hardware resources).
- Do you think there is room for improvement of your services that could be achieved by better utilisation of computational resources when needed through cloud computing technologies?

4. The risk of under provisioning

- Would it be a strong incentive for you to move your data/services to eCloud if eCloud would remove from your organisation the risk of under-provisioning (i.e. not being able to meet demand)?

5. Security

- Does your organisation have any policies regarding where data must be stored and/or where computational resources must be located (such as, not in US, not outside of organisation, etc.)?
- Does your organisation has any policy on security and privacy?
- Do you think that your organisation is able to ensure high privacy and security at a reasonable cost?
- Does the IT team at your organisation monitors and installs security updates and patches shortly (in the matter of hours) after they are released?
- Does the IT team has the sufficient knowledge to ensure safety and privacy of data?
- If eCloud could provide higher security and privacy guarantees for your data than you can achieve at your organisation, would you consider using it?
- Do you see any organisational security related problem in uploading your private data to eCloud (e.g. copyrighted content)?

6. Flexibility of Service and Extensibility

- Would you be interested in building new services on top of eCloud (for example, using its API), or would you rather prefer using off-the-shelf eCloud services.
- Is it likely your organisation would have specific requirements on eCloud APIs before it would be able to start using eCloud?

7. Helpdesk

- In case your organisation put data into eCloud and/or relied on some eCloud services, what level of customer/helpline support would you expect?
- (24 hour helpdesk, respond within 24 hours, respond within a week, etc.)
- Would you be willing to pay more for a more responsive support?

8. Maintenance

- If your organisation did not have to carry out maintenance updates and run security patches:
  a) Would it help your IT team to focus on the tasks central to the organisation?
  b) Would that have the potential to save you some human resources?
  c) Would it positively impact the development of new services for your users (e.g. by decreasing the time to market for new services)?
- Would you consider moving to eCloud if eCloud could remove from you most of the maintenance issues and carry them out professionally at a lower or similar costs?

9. Transparent pay model

- Do you think your organisation would be willing to pay for eCloud services, if these series would demonstrate possible savings and/or improved services provided by your organisation?

- Which payment model do you think would better suit your organisation: pay as utility/set cost per period
- How important is for you payment transparency - all institutions pay the same depending on the usage of the services or all institutions pay the same
- How important is it for you to know exactly how much it will cost in the end. Would estimation be sufficient for you, if you knew how much your organisation pays depending on usage?
- How difficult would it be to change the process of paying from sunk costs to utility billing at your institution?