Final VRE Architecture and Interfaces Definition

Workpackage	**WP 5**	**VRE Infrastructure and Services Design and Development**	
Task	Task 5.1	VRE Architecture and Interfaces Definition	
Author (s)	Pedro Gonçalves		Terradue
	Serena Avolio		ACS
	Sergio De Gioia		ACS
	Raul Palma		PSNC
	Simone Mantovani		MEEO
	Sergio Ferraresi		MEEO
	José Manuel Gómez		ESI
Reviewer (s)	Fulvio Marelli		Terradue
	Ugo Di Giammatteo		ACS
	Simone Mantovani		MEEO
Approver (s)	Cristiano Silvagni		ESA
	Pedro Gonçalves		Terradue
Authorizer	Mirko Albani		ESA
Document Identifier	EVER-EST DEL WP5-D5.12		
Dissemination Level	Public		
Status	Approved by the EC		
Version	1.1		
Date of issue	14 December 2018		

Document Log

Date	Author	Changes	Version	Status
02/07/2017	Pedro Goncalves	Initial Version	0.1	Internal draft
21/07/2017	All technical partners	Specific contributions added	0.9	Internal draft
25/07/2017	Ugo Di Giammatteo	Peer review	0.9	Internal draft
26/07/2017	Simone Mantovani	Peer review	0.9	Internal review
28/07/2017	Simone Mantovani	Final review	1.0	Draft to be approved by the EC
14/12/2018	Cristiano Silvagni	Document status	1.1	Approved by the EC

Table of Contents

List of Figures

List of Tables

Definitions and Acronyms

Acronym	Description
ABAC	Attribute Based Access Control
ACL	Access Control List
AJAX	Asynchronous JavaScript and XML
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BAPI	Business Application Programming Interface
CMS	Content Management System
CRL	Certificate Revocation List
CRUD	Create read Update Delete

CSV	Comma-Separated Values
DOI	Digital Object Identifier
E-PDSC	Extended-Preservation Dataset Content
EAI	Enterprise Application Integration
EDA	Event Driven Architecture
EDI	Electronic Data Interchange
EO	Earth Observation
ERP	Enterprise Resource Planning
ES	Earth Science
ESA	European Space Agency
EVAT	Expert-user Visual Analysis Tool
EVER-EST	European Virtual Environment for Research - Earth Science Themes
FTP	File Transfer Protocol
FTPS	FTP over SSL
GIF	Graphics Interchange Format
GUI	Graphical User Interface
HTML	Hypertext Mark-up Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over TLS, HTTP over SSL, and HTTP Secure
IDE	Integrated Development Environment
IIOP	Internet Inter-ORB Protocol
IMAP	Internet Message Access Protocol
IO	Input Output
IS	Identity Server
ISO	International Organization for Standardization
IT	Information Technology
IWA	Integrated Windows Authentication
JIT	Just In Time
JMX	Java Management Extension
JPEG	Joint Photographic Experts Group
JSON	JS Object Notation
LDAP	Lightweight Directory Access Protocol
LGPL	Lesser General Public License
MEA	Multi-sensor Evolution Analysis
MLLP	Minimum Lower Layer Protocol
MOM	Message Oriented Middleware
MQTT	MQ Telemetry Transport
OAGIS	Open Applications Group Integration Specification

OASIS	Organization for the Advancement of Structured Information Standards
OCSP	Online Certificate Status Protocol
OGC	Open Geospatial Consortium
OPI	Operator Panel Interface
PBAC	Policy Based Access Control
PDSC	Preservation Dataset Content
PNG	Portable Network Graphics
POP	Point of presence
RBAC	Role Based Access Control
RDF	Resource Description Framework
REST	Representational State Transfer
RO	Research Objects
RODL	Research Objects Digital Library
RSS	Rich Site Summary
SFTP	Secure File Transfer Protocol
SLA	Service Level Agreement
SMTP	Simple Mail Transfer Protocol
SPARQL	Protocol and RDF Query Language
SQL	Structured Query Language
SSO	Single Sign-On
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VAC	Visual Analysis Client
VRC	Virtual Research Community
VRE	Virtual Research Environment
WCF	Windows Communication Foundation
WCPS	Web Coverage Processing Service
WCS	Web Catalogue Service
WebCGM	Web Computer Graphics Metafile
WFMS	Workflow Management Systems
WFS	Web Feature Service
WMS	Web Map Service
XML	EXtensible Mark-up Language

Applicable Documents

Document ID	Document Title
EVER-EST DEL WP3-D3.1	VRE Detailed Definition of Use Cases
EVER-EST DEL WP1-D1.1	D1.1 Project and Quality Management Plan
EVER-EST DEL WP5-D5.7	Technical Note on Digital Information and E-Collaboration Services - Final Version

Reference Documents

Document ID	Document Title
Wf4Ever D1.3v2	Wf4Ever Architecture – Phase II
Wf4Ever D2.2v2	Wf4Ever Design, implementation and deployment of workflow lifecycle management
Wf4Ever D3.2v2	Wf4Ever Design, implementation and deployment of Workflow Evolution, Sharing and Collaboration Components

1 Introduction

1.1 Purpose of the document

The purpose of this document is to provide the evidence of the final architecture of the EVER-EST infrastructure. Then, the document aims to show what are the main components of the virtual research environment and describe the API's and interfaces that allow each of them to interact with the others. It is, by definition, a final summary of D5.1 "EVER-EST Architecture", which also contained the description of each component and the approach that was going to be followed for its implementation and integration into the system.

1.2 Background

The current document was not in the original list of project deliverables. It has been formally requested by the EU Officer – Mr. Antonios Barbas – after the first informal interim review that was held in Brussels, June 2016.

1.3 Document structure

This introductory chapter has been thought mainly to provide key information to readers that do not belong to the EVER-EST technical team in order to provide further information concerning the document objectives and background.

The EVER-EST VRE architecture, in Chapter 2, provides a simplified high-level view of the architecture, describing from a very high-level perspective the goals and functions.

Chapter 3, the Research Object in Earth Science, provides an overview of the entire mechanism behind the adoption of Research Objects in the Earth Science domain: this goes from the functionalities that researchers can use from the VRC GUI (described further on in chapter 4), to the backend, to the different API's that the RO-HUB provides to the other components. This chapter is an evolution of the original version in D5.1 and focuses only on the RO's concepts and implementation. All the part related to the VRE and VRC portal and GUI's are described in chapter 4.

Chapter 4, Common Services, refers to the category of services that are vital to guarantee the correct functioning between the various infrastructure components, and between the infrastructure and the final users.

Chapter 5, E-Collaboration services and GUI design, provides details concerning the Content Management System (CMS) that shall ensure that the necessary collaboration tools are deployed which shall include means to integrate via the support of API's tools for (e.g.) video conferencing, chat and messaging. Additional information shall be provided about specific e-collaboration services that focus on Research Objects and that shall be implemented during the project. The initial part of the chapter provides an overview of both the VRE and VRC Graphical User interfaces, describing the design approach, the functionalities and the interfaces of both components.

Chapter 6, E-Research Services, describes the core technologies that guarantee users to perform their research activities. This mainly corresponds to data processing activities and therefore the EVER-EST solution, based on Sandbox Cloud Processing is widely described in all its functionalities and features. The second part of the chapter focuses on workflow management and provides information about the workflow engines that shall be adopted within the project, in connection to ROHub. The last part of the chapter provides a description of which technologies shall allow users' researches and findings to be preserved in the long term.

Chapter 7, E-Learning, describes the technology that shall be adopted to guarantee that e-learning capabilities are integrated in EVER-EST.

2 The EVER-EST VRE Architecture

2.1 The architecture in a nutshell

Figure 1 provides a high-level overview of the EVER-EST VRE architecture.

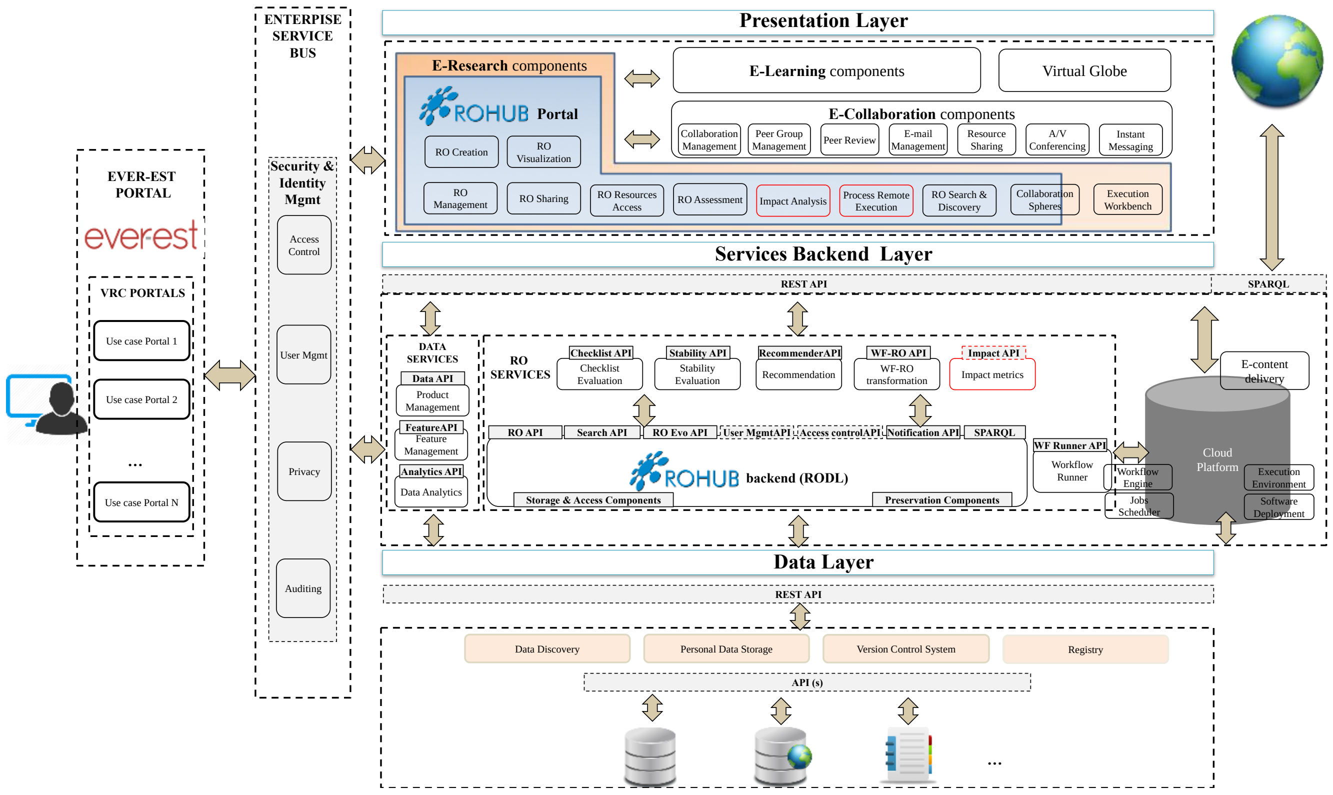


Figure 1 The EVER-EST VRE Architecture

The architecture reflects the organisation of functions across the various layers and with regards to the core elements of the design (on the right of the Enterprise Service Bus vertical component) the following can be identified:

1. **Presentation Layer** - on the top part of the architectural diagram **-** contains the core elements and technologies that shall guarantee the availability of those services and functions, which are directly used by the different communities. It includes the ROHub, the e-collaboration, e-learning and e-research services along with the mechanisms for Earth Science data discovery.

2. **Service Layer** - in the central part of the architectural diagram - provides both generic VRE services and Earth Science specific services. These components represent the reasoning engine of the e-infrastructure and actually orchestrate and manage the services available to the VRE final users.

3. **Data Layer** - bottom part of the design - references the data holdings made available to the VRCs: data is linked and proper means are provided, where feasible, to access it from the VRE. As a default setting, data will not be copied or duplicated, but will continue to reside on the provider's local servers unless it is directly retrieved by the user.

The two components on the left part of the core Architecture design include:

1. The VRE Gateway and the VRC GUI's. This is actually the main Gateway used to access the EVER-EST services, including public research objects, a deployment of the Jupiter Notebook, KPI measurements plus additional on-line training modules made available for guest users. The Gateway contains the shortcut to each VRC user interface. While some services and functions can be seen as common and transversal to all the VRCs, a set of features and functions are specific for each of the communities that shall require an ad-hoc design. The specific features regarding the user interface are not described in detail since they are evolving as the system is used and maintained.

2. The Enterprise Service Bus, with particular focus on the services for Security and Identity management, as the interaction of each user with the VRE, requires multiple levels of security and identity controls across the entire infrastructure (e.g. rights to access the VRE, or to execute a process, discover a catalogue, access data, etc.).

2.1.1 Presentation layer

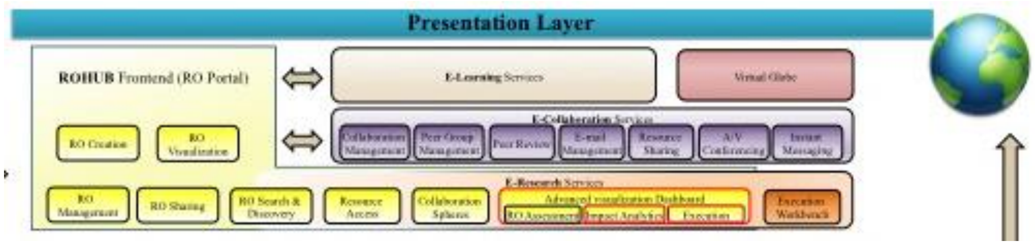


Figure 2 Presentation Layer

The EVER-EST VRE infrastructure is RO-centric: it takes advantage of the new concept and paradigm of Research Objects to provide features that are applied for the first time in Earth Science during the project. Users are able to share their findings in the shape of scientific workflows that can be analysed, executed and possibly modified by others - with full respect of Citation, Trust and IPR management issues. The technology can be also used to prepare workflows that are executed automatically (e.g. every week or each time new data is available) or involve different users within the same organisation.

The way the infrastructure has been designed allows the full or partial use of its services depending on what are the specific constraints of each organisation participating to the project. The following main modules, which belong to the Presentation Layer, have been identified:

1. The VRE Portal provides access to the user interfaces made available during the project lifecycle: from the Virtual Research Community portals fully customised on VRC needs (see Chapter 5), up to the ROHub portal whose functioning mechanisms and API's are widely described in Chapter 3.
2. E-Collaboration services. A consistent part of the standard services for e-collaboration is guaranteed to users by the integration and adaptation of existing COTS. In particular, user and group management, and the creation of specific user interfaces is guaranteed by the use of a Content Management System application, described in Chapter 5. In addition, the integration - via standard API's - of those specific services (e.g. chat, video conferencing, etc.) that emerges from WP3 use

cases. Specific RO centric e-collaboration services are available to final users, as in the case of Collaboration Spheres described in section 3.4.3.

3. User interface for Data Discovery, Access and Analysis. It is evident that a consistent part of the services available to final users must allow Earth Scientist to perform their everyday operations, which normally includes the Earth Science Data Discovery. The user interface component for Data Discovery and Analysis is guaranteed by the MEA platform that is described in section 4.4.
4. E-Learning services are accessible via the Jupiter Notebook technology as described in Chapter 7.
5. E-Research services are available - on the Presentation Layer - via the workflow management tools that shall allow users to create workflow centric Research Objects. Workflow managers are described in section 6.2.

2.1.2 Service layer

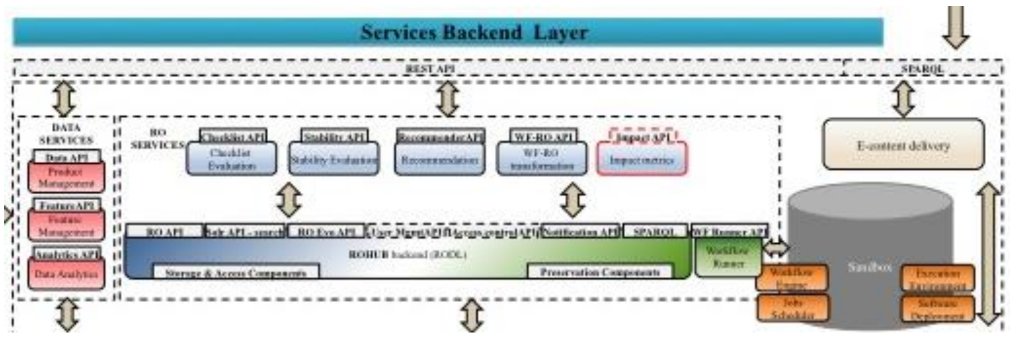


Figure 3 Service Layer

As previously described, the service layer is the reasoning engine of the infrastructure. In most cases it reflects the backend or the specific implementation of the services available on the presentation layer. With respect to the Service Layer the following have been identified:

1. Research Object Digital Library (RODL). It is the backend of the ROHub component and is described in section 3.4
2. Common Services. These services are almost totally hidden to the final users although they are essential to guarantee the infrastructure integration and user access rights management.
3. The Sandbox is the core element that shall guarantee VRE users the capability to perform Cloud based processing of Earth Science data. Connected to the workflows engine on the presentation layer, it is described in section 6.1.
4. Preservation services shall be integrated within the RODL. They take advantage of RO concepts but also leverage on findings in Earth Science preservation technologies as described in section 6.3.

2.1.3 Data layer

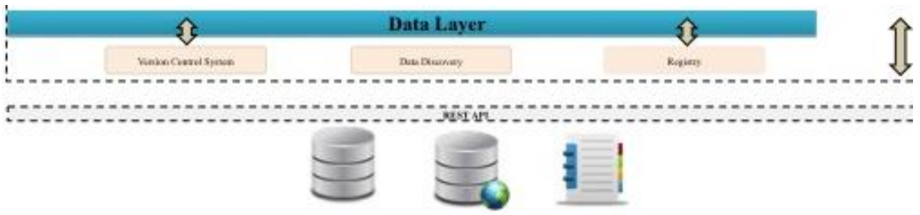


Figure 4 Data Layer

This layer manages the discovery and access functionalities to external Earth Science catalogues. This is mostly based on the connection to Earth Observation catalogues (as it is the case of FedEO[1]) and specific catalogues as CoCoNet[2]. Further catalogues based on Open Search Geo and Time Extensions[3] can be integrated in the future if the need arises in discussions with the user communities.

[1] http://wiki.services.eoportal.org/tiki-index.php?page=FEDEO

[2] http://coconetgis.ismar.cnr.it

[3] http://www.opengeospatial.org/standards/opensearchgeo

3 Research Objects in Earth Science

3.1 Overview

The VRE portal enables scientists in the Earth Science domain with the functionalities to discover, create, manage, share, (re) use and preserve the research artefacts related to their use cases or studies through the Research Object (RO) concept and tools. This supports their capability in the use of data (known as e-research functionalities). Such functionalities for RO management could be (loosely) compared to the document management functionalities provided by systems like Alfresco CMS (or other similar systems). The e-research functionalities provided by the VRE portal also enable users to manage their executable artefacts (application, jobs, workflows) and their execution environment.

The portal enables scientists to collaborate in different modes and to communicate throughout the research lifecycle (e-collaboration functionalities), and provides access to training materials and online courses (e-learning functionalities) to guide future data scientists to overcome the challenges of increasing data volumes and support their ability to validate, analyse, visualise, store, and curate the information.

A major component of the VRE portal is the ROHub system inherited from the Wf4Ever project[4], which integrates a set of RO services alongside a graphical user interface to provide a holistic solution for RO management. In the rest of this section the ROHub and other RO components are described, which together with the components described in Chapter 0 address the e-research functionalities, while the e-collaboration and e-learning components will be described in their respective services sections.

3.2 Research objects overview

Research Objects are semantically rich aggregations of resources that bring together data, methods and people in scientific investigations. Their goal is to create a class of artefacts that can encapsulate our digital knowledge and provide a mechanism for sharing and discovering assets of reusable research and scientific knowledge[5]. The RO concept and definition was developed as part of the Wf4Ever project. Below, Figure 5 depicts a high-level view of the Research Objects concept graphically.

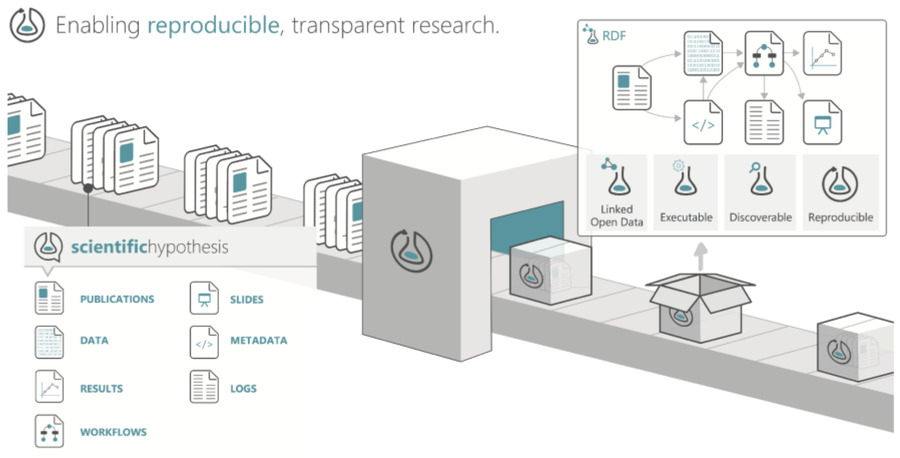


Figure 5 Research Objects specification

EVER-EST VRE is the first RO-centric native infrastructure leveraging the notion of Research Objects and their application in observational rather than experimental disciplines and particularly in Earth Science. The

[4] http://www.wf4ever-project.org/

[5] http://www.researchobject.org/

nature of ROs, which are portable units of scientific knowledge, enables the sharing, preservation, reuse, communication, etc. of the results in Earth Science domain.

3.3 Research objects management functionalities

Based on our experience with Research Objects, a number of RO functionalities that shall become part of the EVER-EST architecture definition are identified. Some of these functionalities are based on previous results, adapted to EVER-EST, plus some new functionality specifically addressing EVER-EST needs. The functionalities can be grouped by their nature into four main categories (as depicted in Figure 6):

- **RO Access & Usage**: these functionalities enable consumers (scientists, researchers, other end-users) to determine the existence, description, location and availability of stored RO resources; to request, access and receive (download) ROs and aggregated resources; to edit, create and share ROs resources; to visualise, inspect, annotate and comment ROs and aggregated resources; to assess and visualise RO characteristics (e.g., quality, impact, etc.).
- **RO Data Management & Analysis**: these functionalities generate, maintain and provide access to added value data derived from, or related to, RO resources (e.g., quality information, recommendations, metadata extraction, impact metrics, etc.).
- **RO Storage and Preservation**: these functionalities provide the underlying mechanisms for the storage, maintenance, search, index and retrieval of ROs and aggregated resources, ensuring appropriate levels of protection and the preservation of resources.
- **RO Lifecycle**: these functionalities support the dynamic nature of RO resources, enabling the execution of executable RO resources (e.g., workflows), the management of RO evolution, including versioning and curation activities, and the provision of monitoring and notification mechanisms to ensure the correct preservation of ROs through time.

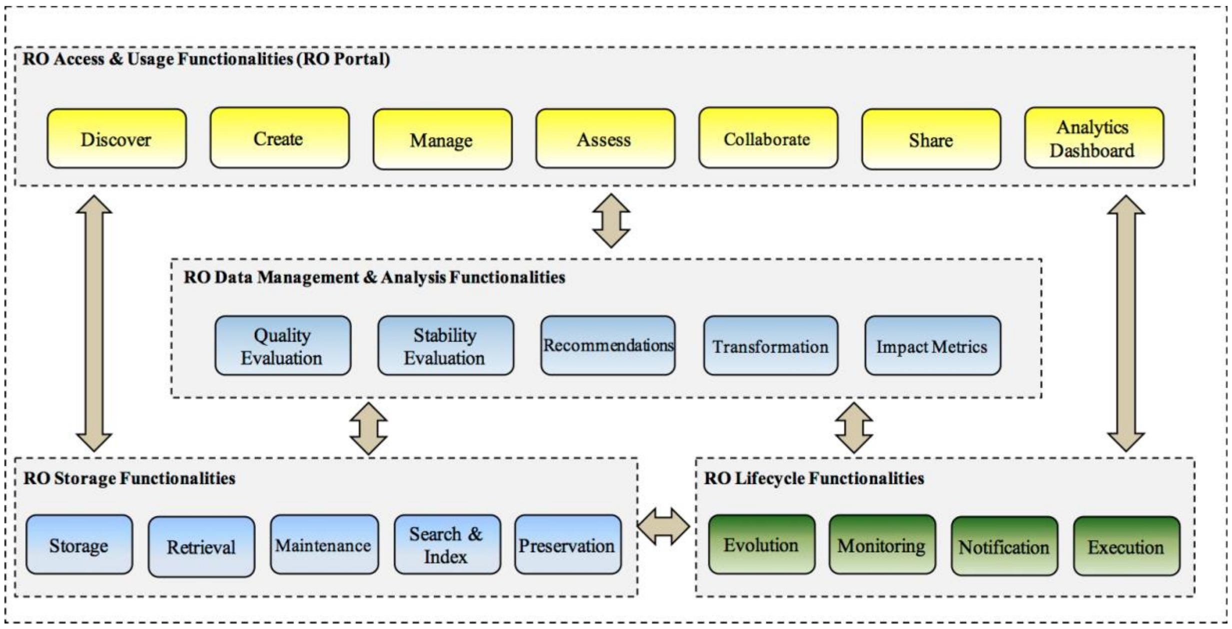


Figure 6 Management functionalities

3.4 RO components and APIs

RO functionalities are realised by a set of services exposing RESTful APIs alongside client applications. A service may implement one or several APIs depending on the focus or plurality of the functionality provided, and in turn the API is based upon the RO models it uses when storing and operating upon information within the service.

Most of these service components are existing prototypes and few others are developed during EVER-EST project. In Figure 7 the APIs implemented by the services are depicted, marking with red those components that are not existing at the EVER-EST project kick-off, and with dashed lines the APIs that may be replaced by more generic implementations in EVER-EST. The latter is, for example, the case of identity management and access control that are replaced by a specific component developed ad hoc for the project and described in Chapter 0.

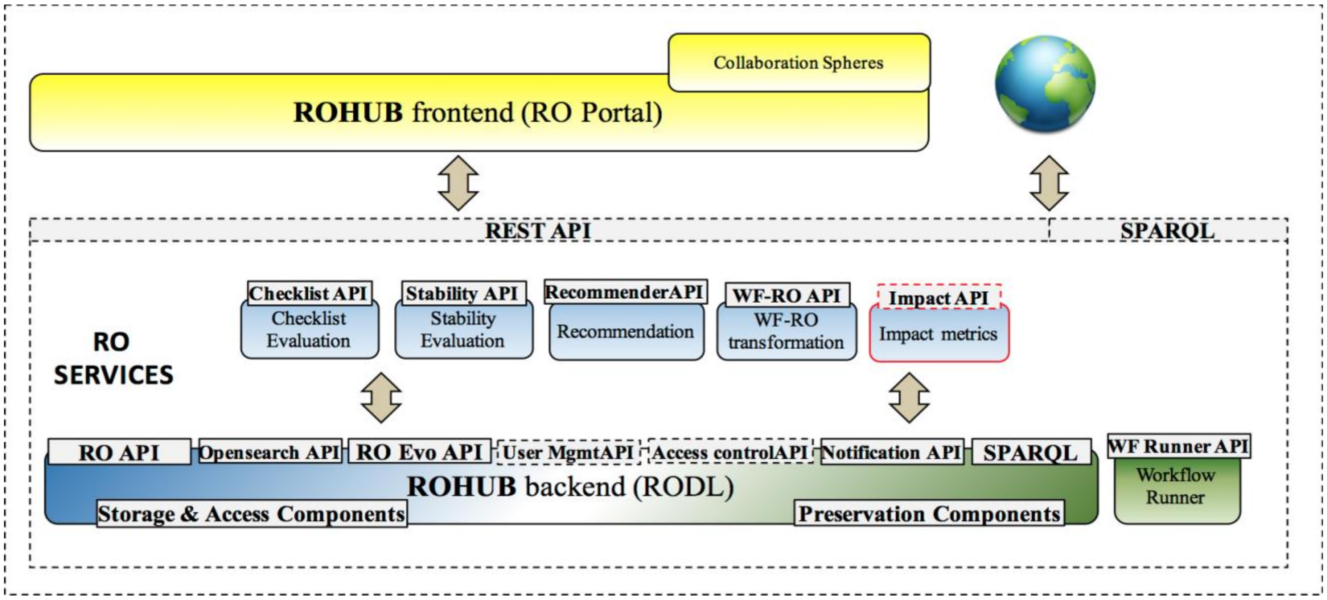


Figure 7 Components and API's

In the following the description of these components is provided, starting with ROHub the main system, then the collaboration spheres (a GUI for RO discovery), and then the rest of RO added-value services (most of them already integrated in ROHub).

3.4.1 ROHub

ROHub is a research object (RO) management platform supporting the storage, lifecycle management, sharing and preservation of research findings via ROs. It implements a set of features to help scientists throughout the research lifecycle to: (i) create and maintain high-quality ROs that can be interpreted and reproduced in the future; (ii) collaborate along this process; (iii) publish and search these objects and their associated metadata; (iv) manage their evolution; and (v) monitor and preserve them supporting their accessibility and reusability.

It can be characterised as follows:

- Holistic approach to scientific content preservation;
- Targets researchers, scientists, students, and enthusiasts, but also supports other stakeholders like industry, investors, and publishers;
- Online platform supporting multiple client applications, also supporting local installations;
- Modular and extensible to cover specific domains needs;

- Non workflow-specific.

The core component of ROHub is the backend service, RODL, which exposes a set of Restful APIs implementing the RO model[6] to support programmatically access to the provided functionalities (see Section 4.1.1). The backend can support multiple client applications. ROHub provides by default a reference Web application (ROHub portal) exposing all the research object functionalities to the end-users; however, other client applications, such as the EVER-EST portals, collaboration spheres, (alpha)myexperiment, and command line tool RO Managerrely on ROHub backend.

ROHub also integrates other RO added-value services, at the front and/or back end side, including notification, transformation of workflows into research objects, quality and stability assessment, metadata enrichment, and exploratory search (see Section 4.1.3).

In the context of the EVER-EST VRE, ROHub backend (RODL) is one of the central middleware components that supports (and interacts with) the other components in the architecture. However, the VRE provides two frontends to the end-users for using, accessing and managing research objects: the EVER-EST VRE portals (plus the specific VRCs portals), and the ROHUB portal. The EVER-EST portal enables scientists of the four VRCs and interested users (e.g., citizens) to discover, produce and manage their research work in Earth Observation (EO) through the Research Object (RO) and EO tools. This portal is the main entry point to the VRC portals, each of them providing a dedicated working environment interface customised to fit the VRC needs. Internally, these portals use research objects as the underlying mechanism to manage and preserve their work (interacting with RODL through the RO middleware API). However, they abstract the research object terminology and details from the user interface in order to enable tailored access to (some of) the research objects management capabilities, thus providing an operational oriented interface to research objects. ROHub portal, on the other hand, provides an advanced, lifecycle oriented, interface exposing the full set of research object management capabilities to the end-users. It is intended for more technical/advanced users that have been already familiarised with research objects, or who would like to analyse or manage the research object lifecycle in more detail. So, while in the ROHub portal, the user may need to perform multiple individual operations to build a research object (create, annotate, add resources, etc.), the EVER-EST portal may encapsulate all these operations in a single control. However, ROHub portal can also support inter-VRCs collaboration, for instance, by providing richer interfaces to discover research objects and resources based on cross-domain characteristics, or by enabling open discussions around them. Moreover, other EVER-EST VRE components like the cloud platform, Seafile and jupyter notebooks may interact with ROHub for carrying out some tasks (e.g., execute a workflow in a research object, store results in the personal file storage, etc.).

3.4.1.1 ROHub architecture

At a lower level, ROHub backend (RODL) itself has a modular structure that comprises access components, long-term preservation components and the controller that manages the flow of data (see Figure 8). The access components include a storage backend, a semantic metadata triplestore, a search & index server, and an internal relational database. The storage backend can use a built-in module for storing ROs directly in the file system (as the online platform), or it can be based on the digital library system dLibra[7], which provides file storage and retrieval functionalities, including file versioning and consistency checking.

[6] http://wf4ever.github.io/ro/

[7] http://dlibra.psnc.pl/

The semantic metadata are parsed and stored in a triplestore backed by Jena TDB[8]. The use of a triple-store offers a standard query mechanism for clients and provides a flexible mechanism for storing metadata about any component of a RO that is identifiable via a URI.

Additionally, the semantic metadata is indexed in a search & index server based on Solr, enabling the implementation of efficient search mechanisms and interfaces. Finally, internal information about the RODL state and interactions with other services is stored in a MySQL database.

The long-term preservation component stores ROs in a secure, replicable storage platform, including resources and annotations. Additionally, ROHub supports standard preservation tasks like fixity checking and file format decay alerts, but it also monitors the RO quality through time against a predefined set of requirements. If a change is detected, notifications are generated as Atom feeds.

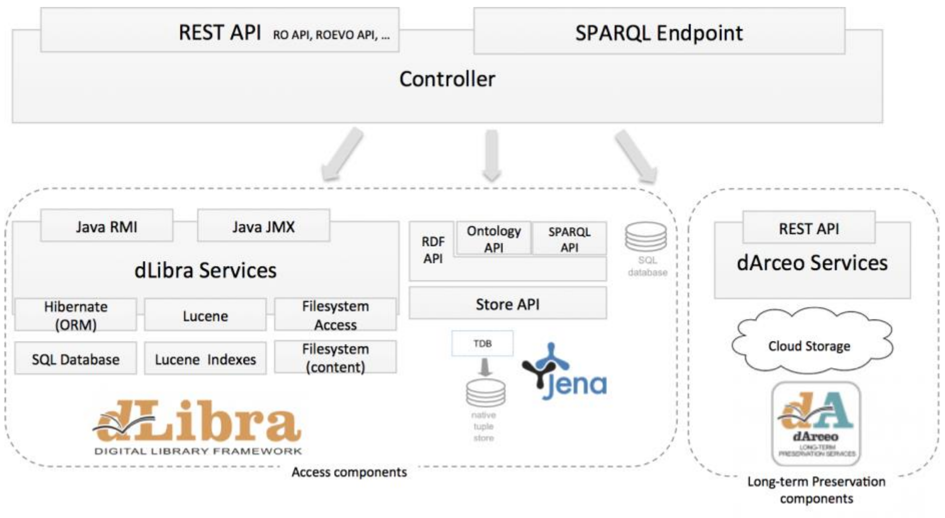


Figure 8 ROHub backend architecture

3.4.2 ROHub Interfaces

The architecture principles of Research Objects are the adoption of Linked Data and RESTful approaches. Accordingly, a set of APIs services has been built in order to manage ROs and its information following these principles. The focus of these APIs was interoperability and compatibility between different software components. The API suite includes several specifications for managing different aspects of ROs, including the core RO lifecycle and evolution, notifications, recommendations, quality information, workflow-centric information, etc. In the following, a brief introduction is provided for each of them, and the reader may refer to the full specifications that are publicly available[9] for additional information.

The two main APIs are:

- **RO API** enables the storage and retrieval of ROs and their aggregated resources, as well as annotating them. It defines the formats and links used to create and maintain ROs in accordance with the RO model, and to specify relations between different resources.

[8] https://jena.apache.org/documentation/tdb/

[9] https://github.com/wf4ever/apis/wiki/Wf4Ever-Services-and-APIs

- **RO evolution API** enables the management of the ROs lifecycle. It defines the formats and links used to change the RO stage according to the evolution model, as well as to retrieve the RO evolution provenance.

Besides the two core APIs the system exposes the following APIs to allow different useful operations:

- **OpenSearch API** enables the searching and retrieval of Research Objects in a standard and accessible format. The exposed API implements also the geospatial and time extensions of OpenSearch.
- **Notification API** generates notifications whenever there are events related to Research Objects. Notifications are available in an Atom format[10] (similar to RSS), and clients are able to poll them to check if there is anything new.
- **User Management API** provides methods for identity management to create/retrieve/update/delete users (based on SCIM[11]), and methods for authorisation management (based on OAUTH 2.0[12]) to create/retrieve/delete access tokens and to register/retrieve/update/delete client applications
- **Access control API** enables to grant permissions to the research object based on roles and modes. User roles can be set as owner, editor and reader, while RO modes can be set as private, public and open.

3.4.2.1 ROHub main functionalities

From the user point of view (i.e., frontend), ROHub provides the following key features:

- **Create, manage and share ROs**. There are different methods for creating ROs in ROHub: (i) from scratch, adding resources progressively; (ii) by importing a pack of resources from other systems (currently myExperiment[13] platform); (iii) from a ZIP file aggregating files and folders; (iv) by uploading local ROs from the command line using RO Manager Tool[14]. Resources can be added and annotated from the content panel that also shows the folder structure. ROHub provides different access modes to share the ROs: open, public or private. In the open mode, anyone with an account can visualise and edit the RO. In the public mode, everyone can visualise the RO, but only users with correct permissions can edit it. In private mode, only users with correct permissions can visualise and/or edit the RO.
- **Discover, explore and reuse ROs**. ROHub provides a keyword search box and a faceted search interface to find ROs, as well as a SPARQL endpoint interface to query RO metadata. Another option for RO discovery is provided by the Collaboration Spheres, accessible from ROHub, which provides an exploratory search interface as described in Section 4.2. It is worth mentioning that as part of EVER-EST initiative, RO search capabilities based on geo-referenced criteria has been implemented via the OpenSearch endpoint, and a user interface (e.g., map) is provided.
- **Assessing RO quality** Users can visualise a progress bar on the RO overview panel, which shows the quality evaluation based on set of predefined basic RO requirements. When clicked, users can visualise further information about the RO compliance. Users can also get more information about the quality of the RO from the Quality panel, where they can choose from different

[10] https://tools.ietf.org/html/rfc4287

[11] http://www.simplecloud.info/specs/draft-scim-api-00.html

[12] http://oauth.net/2/

[13] http://www.myexperiment.org/home

[14] https://github.com/wf4ever/ro-manager

templates to use as the basis for evaluating the RO. This functionality integrates the **checklist evaluation service** (see section 3.4.4). Additionally, users can open the RO monitor tool to visualise the RO stability and quality throughout time. This tool relies on the **stability evaluation service** (see section 3.4.4).

- **Managing RO evolution**. From the RO overview panel, users can create a snapshot (or release) of the current state of the (live) RO, at any point in time, for sharing the current outcomes with colleagues, get feedback, send it to review, or to cite them. Similarly, when the research has concluded, they can release and preserve the outcomes for future references. ROHub keeps the versioning history of these snapshots, and calculates the changes from the previous one. Users can visualise the evolution of the RO from the History panel, and navigate through the RO snapshots. Additionally, ROhub enables users to fork an existing RO to facilitate the reuse of ROs.
- **RO Inspection**. ROHUB provides a content navigation panel to traverse the RO content, showing information about resource size or number of folder entries. Additionally, from the content panel, the user can aggregate additional resources, and create folders. Once a resource is selected the metadata associated is displayed, and the users with permissions is able to modify (add/edit/remove) annotations, move the resource, or delete the resource itself.
- **Workflow annotation and transformation** When users select a workflow resource from the content panel, they are able to extract workflow metadata and resources. This allows to (i) generate the workflow description representation according to the RO model; (ii) retrieve and aggregate the workflow sub-components, such as scripts, nested workflows and web services, in the main RO; (iii) transform the workflow resource into a workflow bundle. This functionality relies on the **WF-RO transformation service** (see section 3.4.4) and therefore at the moment is compatible only with Taverna[15] workflows.
- **Nested ROs**. Scientists can aggregate any type of resources, including links to external resources and other ROs. The latter allows aggregating RO bundles, which are structured ZIP files representing self-contained ROs that facilitate their transfer and integration with 3rd party tools. Taverna, for example, can export provenance of workflow runs as RO Bundles. In ROHub, bundles are unpacked into nested ROs, exposing their full content and annotations. Hence, scientists can navigate through the inputs, outputs and intermediate values of the run, something potentially useful for future reproducibility.
- **Preserve and monitor ROs**. ROHub includes long-term preservation component that stores objects (in a secure, replicable storage platform) and includes monitoring features, such as fixity checking and RO quality, which generate notifications when changes are detected. This can help to detect and prevent, for instance, workflow decay, occurring when an external resource or service used by a workflow becomes unavailable or is behaving differently. Users can visualise changes in the RO, regarding the content and quality monitoring in the notification panel and they can subscribe to the atom feed to get automatic notifications. This functionality integrates the **stability evaluation service** (see section 3.4.4).
- **Semantic enrichment**. An RO is enriched automatically (or on demand) with structured metadata extracted from its textual content, including the main concepts, domains, lemmas and named entities, in order to facilitate its discovery via the faceted/keyword search interfaces. Such metadata complements the metadata provided explicitly by scientists, offering a richer, machine-readable description of the RO, exploited in the keyword and faceted search interfaces. This functionality integrates the **RO enrichment service** (see section 3.4.4).
- **DOI and citation**. Now a DataCite (www.datacite.org) DOI allocator, ROHub can assign a DOI to the released ROs, enabling citation and stimulating scholarly communication and sharing before

[15] http://www.taverna.org.uk/

actual paper publication. DOI assignment follows RO release after automatically checking that the RO follows DataCite's policies.

A running instance of ROHub is publicly available[16] for testing, and a demo video showcasing its functionalities is available in YouTube[17].

3.4.3 RO discovery interface: Collaboration Spheres

The Collaboration Spheres provides an alternative user-oriented interface for RO discovery targeted to scientists interested in finding Research Objects based on similarity aspects. Collaboration Spheres is a web user interface for the visualisation of correlation between similar objects (e.g., users, Research Objects) based on collaborative filtering and versatile keyword content-based recommendations. It implements a visual metaphor based on spheres, which uses concentric circles, where the similarity is represented using the distance from the centre of the sphere to the place where the object is shown, and separated circles, where different type of information or ranges can be displayed by representing different separated circles. In particular, the interface associated to the Collaboration Spheres contains four main spheres:

- **The User**. It displays the active user.
- **The Inner Sphere** (context of interest). It represents the context of interest. This circle contains the users and ROs that are selected or pre-defined by the active user. In order to create a context of interest, the inner sphere is populated by drag-and-dropping relevant ROs and users from lists ranked by relevance.
- **The Intermediate Sphere** (model-based recommendation). It represents information associated to the recommendation obtained by using the context of interest as input for the recommendation techniques. The ROs and members of social network displayed in this sphere correspond to suitable items according to the context of interest defined in the previous sphere. It follows an inside-outside criterion where the inner part flows towards the outside. At this stage the recommendation obtained by this layer uses a content-based recommendation approach based on tags and annotations. This is a social-oriented approach and it is based on the fact that people are many times more reliable than search engines (*"I trust my friends more than I trust strangers."*).
- **The Outer Sphere** (user-based recommendation). It contains recommended ROs and users based on past user actions rather than in the information explicitly defined in the context of interest. In order to populate this sphere, it also relies on predictive models that provide the user with new possible interests.

Figure 9 depicts the functionality of this tool. This tool relies on the recommendation service (see Section 3.4.3), and thus also on user profiles and social networks around the resources such as the ones provided by myExperiment[18] platform, the largest public repository of scientific workflows, which also includes social features.

The collaboration spheres interface is agnostic from the different similarities calculated by the underlying recommenders. Currently it supports the following types of similarities:

[16] http://www.rohub.org/

[17] https://youtu.be/TxW2wvreyoQ

[18] http://www.myexperiment.org/

- **RO vs. RO**. It represents ROs that share common properties. This relation is evaluated by analysing the similar or identical tags described in the ROs, and thus, it is classified as content-based type.
- **User vs. User**. It represents users that share common interests, and is aimed towards supporting information exchange about the domain and the activities being carried out. This relation is evaluated by analysing the similarity between tags contained in the user profiles and associated information, and thus, it is classified as content-based.
- **RO vs. User**. It represents the functionalities that are shared with user interest or vice versa. This relation is evaluated by analysing the similar or identical tags described in the user profile and RO tags (content-based), and by analysing historical/related use of ROs by a user and predicting new possible RO's likes (collaborative filtering type).

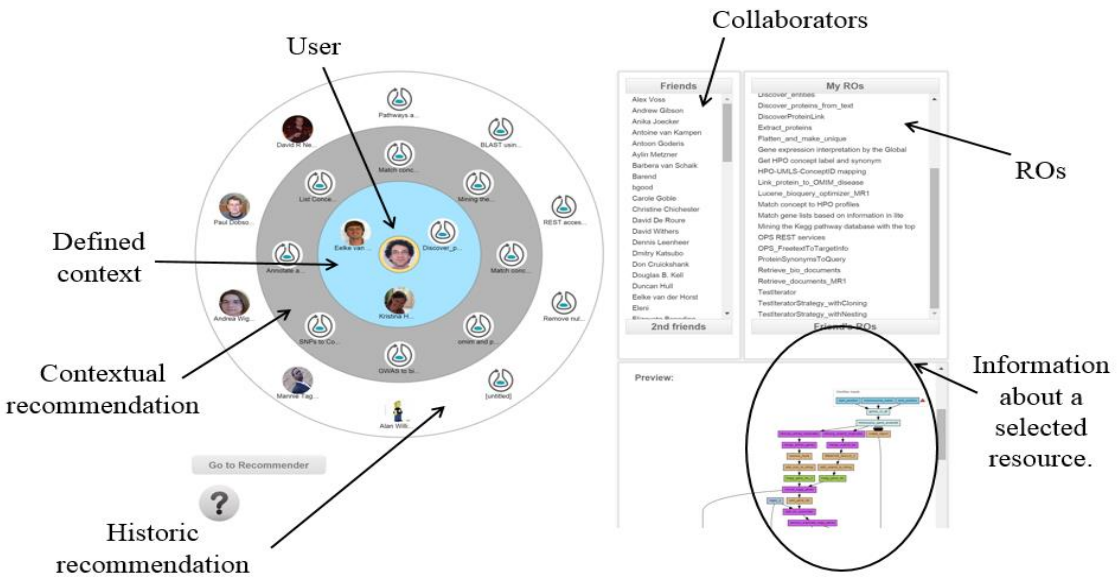


Figure 9 Collaboration Spheres interface

A running instance of the collaboration spheres is publicly available[19] for testing.

3.4.4 RO added-value services

The other RO added-value services that generate, maintain and provide access to added value data derived from, or related to, RO resources are herein described. Note that these services are also publicly available[20] for testing.

- **RO checklist service[21]** provides remote access to the minim-based evaluation of Research Objects, used to test for completeness, runnable or repeatability.
- **RO stability service[22]** is based on the evaluation of the RO along time. The service captures concrete values provided by the evaluation of the RO in different moments of its evolution or by subjecting

[19] http://sandbox.wf4ever-project.org/collab2/index.html
[20] http://sandbox.wf4ever-project.org/
[21] http://sandbox.wf4ever-project.org/roevaluate/
[22] http://sandbox.rohub.org/decayMonitoring/rest/getStability

the snapshots of a RO to the checklist evaluation. It allows testing the ability of a Research Object to achieve its original purpose after being subject of changes on its resources. The stability service also includes a **decay monitoring visualisation tool**. This interface allows visually monitoring and keep track of track of the status of external datasets and web services required for workflow execution.

- **RO enrichment service**[23] generates automatically semantic annotations based on the (textual) resources aggregated in order to improve the discoverability and searchability of research objects. The annotations are written according to the content description vocabulary (**https://w3id.org/contentdesc**). These annotations are leveraged by ROHub search engine to produce more accurate results and to provide new facets to explore the research object collection. In addition, the recommender service uses these structured metadata to suggest research objects of interest to the users.
- **RO recommender service**[24] suggests research objects that might be of interest according to user's research interests. The recommender system follows a content-based approach in the sense that it compares the research object content with the user interest to draw the list of recommended items. This comparison is based on the annotations added by the semantic enrichment process. The user interests are identified from the top concepts in the user's research objects. These concepts are then compared with the concepts that annotate the research objects in the whole collection. The user interest can be increased by i) adding specific research objects from other users or ii) adding a different scientist. In the former case the main concepts of the research object are added to the user's interests and in the latter case the scientist interests are added to the user's interests.
- **RO aggregation tool**[25] allows researchers to create a new research object with the content of other existing research objects or to establish relations with these research objects without copying their content. The motivation to generate this tool comes from the fact that the VRCs wanted to group bibliographic resources according to event dates and topics
- **Scholarly communication services**[26] harvest research object citation information from Google Scholar and Microsoft Academic, which are specialised sites that gather citation information of research works in scholarly communications
- **Workflow transformation (WF-RO) service**[27] offers a service that transforms workflows into Research Objects. Workflows are often complex data structures that embed data such as their sub-resources, annotations and provenance. The service described by this API creates a research object that exposes these data according to the RO model.

In the previous version of this document, the workflow runner service was introduced. This service provides an abstraction layer on top of workflow management systems that support the execution of workflows remotely or programmatically, in order to generate research objects that encapsulate the workflow execution provenance. Yet in the current implementation the service was only abstracting Taverna server, but during the testing it was found that it was not updated with respect to the latest Taverna server version, and making such update would require significant effort. In the meantime, it was found out that Taverna server already provides the functionality for generating research objects with the workflows execution provenance. Hence, it was decided to resign from the workflow runner service and instead use directly

[23] http://everest.expertsystemlab.com/ro/enrichment

[24] http://everest.expertsystemlab.com/EverEstSpheres/services/jsonservices/api

[25] http://everest.expertsystemlab.com/rocreation/

[26] http://everest.expertsystemlab.com/scholar/gscholars/write-title-here & http://everest.expertsystemlab.com/scholar/msacademics/write-title-here

[27] http://sandbox.rohub.org/wf-ro/

Taverna server for the EVER-EST VRE. Taverna server implements a lightweight REST API that is also publicly available[28].

Some of these services & tools are already integrated or accessible from ROHub portal (checklist, stability, enrichment, wf-ro), and some other will be available in the near future (recommender, aggregation, and scholarly).

3.4.5 RO and ROHub extensions for Earth Science

As part of EVER-EST project, a gap analysis has been conducted to identify the necessary updates to be implemented in the RO model. This analysis revealed five main areas where the gap between the coverage provided by the RO model that far and the needs of the earth scientists were significant: Geographic data, time, intellectual property rights, data access policies, and general-purpose information. In some cases, such information was not covered at all by the previous version of the research object model (geographic, time, data access policies), and in other cases it was not covered with sufficient detail as required by the earth scientists (intellectual property rights). The main additions to the model are summarised below:

- Geographical, the coordinates of the region relevant for the research object and the observation it represents.
- Time, indicating the time span covered in the observation.
- Intellectual Property Rights, including copyright holder, copyright starting year, type of license and attribution.
- Data Access Policy, i.e. the access level and policies under which the research object can be accessed.
- General Metadata, including the main scientific discipline of the research object, the size of the resources aggregated by the research object, the submission date when the research object was released, its digital object identifier (DOI), the status according to the research object life cycle, and the main target community.

The computational resources covered by the model have also been extended to cover not only scientific workflows but also other infrastructure, e.g. virtual machines or dedicated software, frequently used in Earth Sciences. Furthermore, earth scientist demanded new types of research objects according to the kind of the aggregated resources, and hence the research object types has been extended, to characterise not only workflow-centric ones, but also data-centric research objects, service-centric research objects, and documentation and bibliographic research objects, and the associated checklists that assess their quality has been developed. Finally, the research object lifecycle was extended with a new status (forked), which characterises a new branch of the research object derived from the main one.

With respect to the implementation, ROHub already implements the extensions in the model and checklists; however, some functionality is still needed to enforce the new metadata if available (e.g., detailed access policies). Additionally, ROHub already implements the extended lifecycle model supporting forks, the possibility to generate DOIs for snapshots/releases. Moreover, several new RO added value services have been implemented, including the RO enrichment service that is also integrated with ROHub, the scholarly communication services, the RO aggregation tool, and a new recommender service integrated in the new version of the collaboration spheres.

Finally, a brand new RO portal has been implemented as part of EVER-EST project (see Section 4.3).

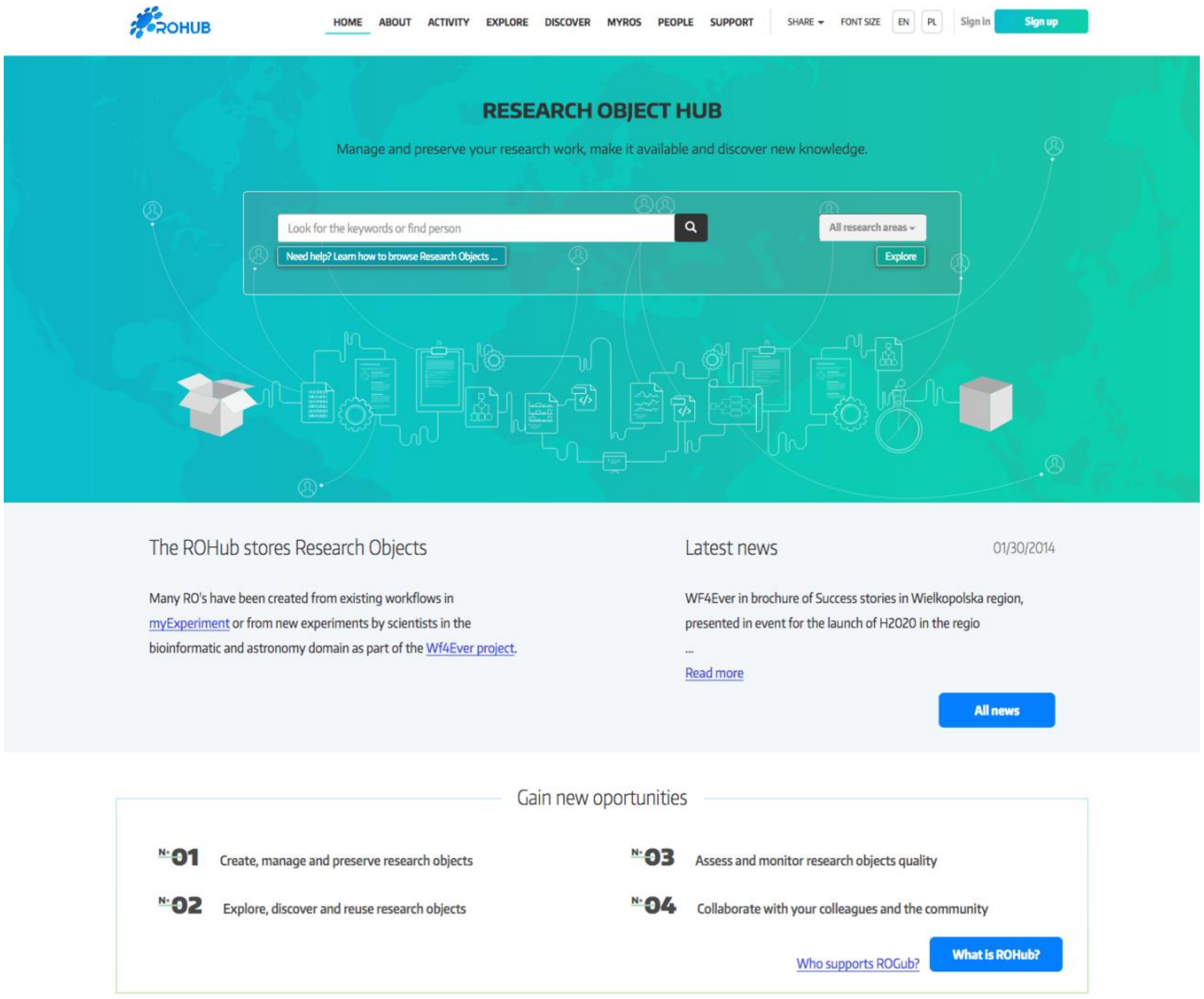
[28] http://dev.mygrid.org.uk/wiki/display/tav250/REST+API

3.4.6 The new RO-HUB Portal

ROHub portal is a web client application providing a comprehensive user interface for the management and preservation of research objects (ROs).

The new portal is based on Play framework providing a lightweight, stateless, web-friendly architecture, and angularJS, a structural framework for dynamic web apps. The combined usage of these technologies enables creating a modular web application with a set of visual components that can be easily reused in other applications.

The development of the portal was conducted in two parallel threads. The first thread, still ongoing, focussed on the implementation of a set of visual components required to implement the existing functionalities available in previous portal plus the extensions required for the Earth Science domain. The second thread focused on implementing a new portal design based on a usability analysis.



RESEARCH OBJECT HUB

Manage and preserve your research work, make it available and discover new knowledge.

The ROHub stores Research Objects

Many RO's have been created from existing workflows in myExperiment or from new experiments by scientists in the bioinformatic and astronomy domain as part of the Wf4Ever project.

Latest news 01/30/2014

WF4Ever in brochure of Success stories in Wielkopolska region, presented in event for the launch of H2020 in the regio

...

Gain new oportunities

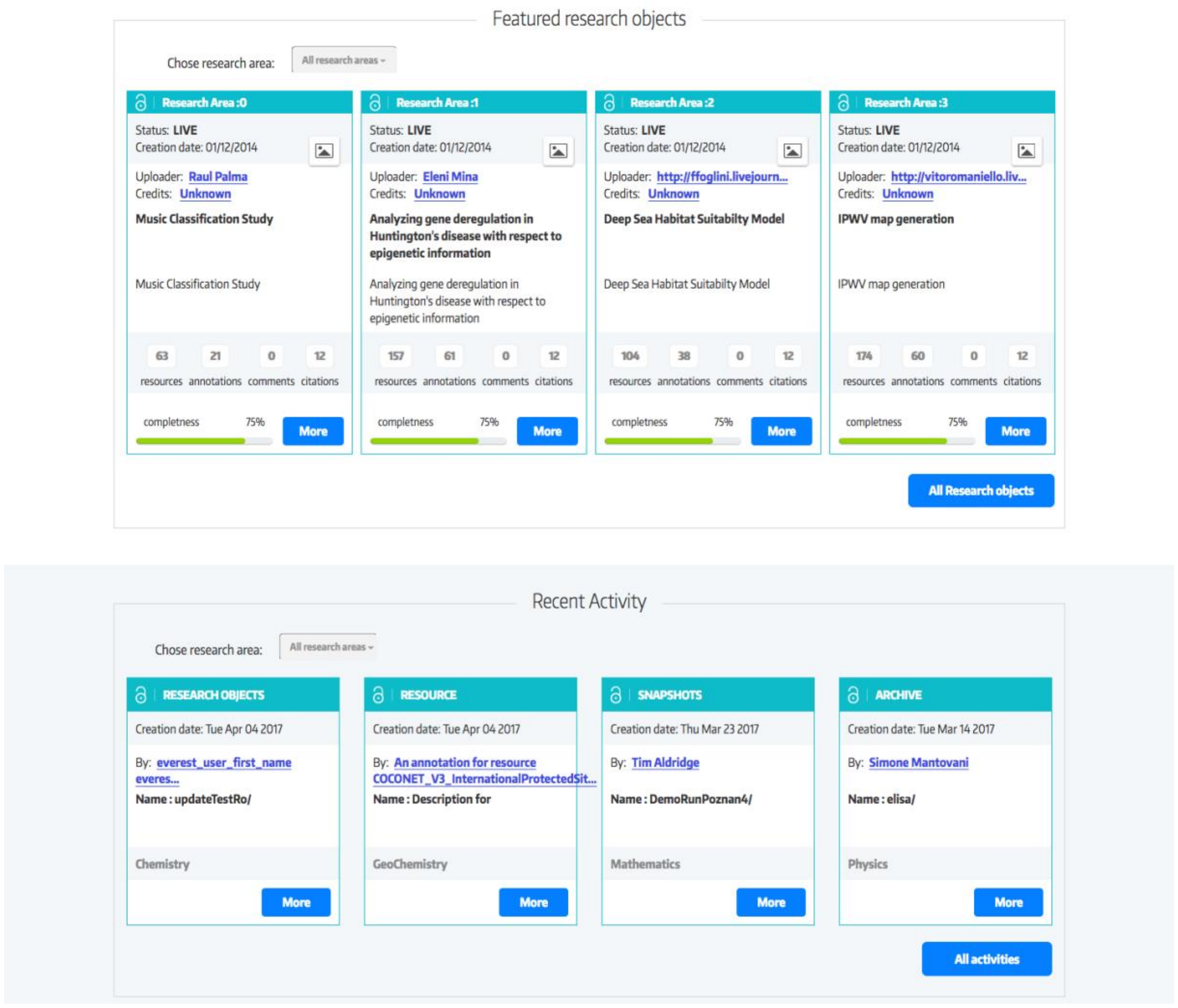
N° 01 Create, manage and preserve research objects

N° 02 Explore, discover and reuse research objects

N° 03 Assess and monitor research objects quality

N° 04 Collaborate with your colleagues and the community

Figure 3-6 ROHub portal main page (1/3)



Featured research objects

Chose research area: All research areas

Research Area :0
Status: LIVE
Creation date: 01/12/2014
Uploader: Raul Palma
Credits: Unknown
Music Classification Study
Music Classification Study
63 resources, 21 annotations, 0 comments, 12 citations
completness 75%
More

Research Area :1
Status: LIVE
Creation date: 01/12/2014
Uploader: Eleni Mina
Credits: Unknown
Analyzing gene deregulation in Huntington's disease with respect to epigenetic information
Analyzing gene deregulation in Huntington's disease with respect to epigenetic information
157 resources, 61 annotations, 0 comments, 12 citations
completness 75%
More

Research Area :2
Status: LIVE
Creation date: 01/12/2014
Uploader: http://ffoglini.livejourn...
Credits: Unknown
Deep Sea Habitat Suitabilty Model
Deep Sea Habitat Suitabilty Model
104 resources, 38 annotations, 0 comments, 12 citations
completness 75%
More

Research Area :3
Status: LIVE
Creation date: 01/12/2014
Uploader: http://vitoromaniello.liv...
Credits: Unknown
IPWV map generation
IPWV map generation
174 resources, 60 annotations, 0 comments, 12 citations
completness 75%
More

All Research objects

Recent Activity

Chose research area: All research areas

RESEARCH OBJECTS
Creation date: Tue Apr 04 2017
By: everest_user_first_name everes...
Name : updateTestRo/
Chemistry
More

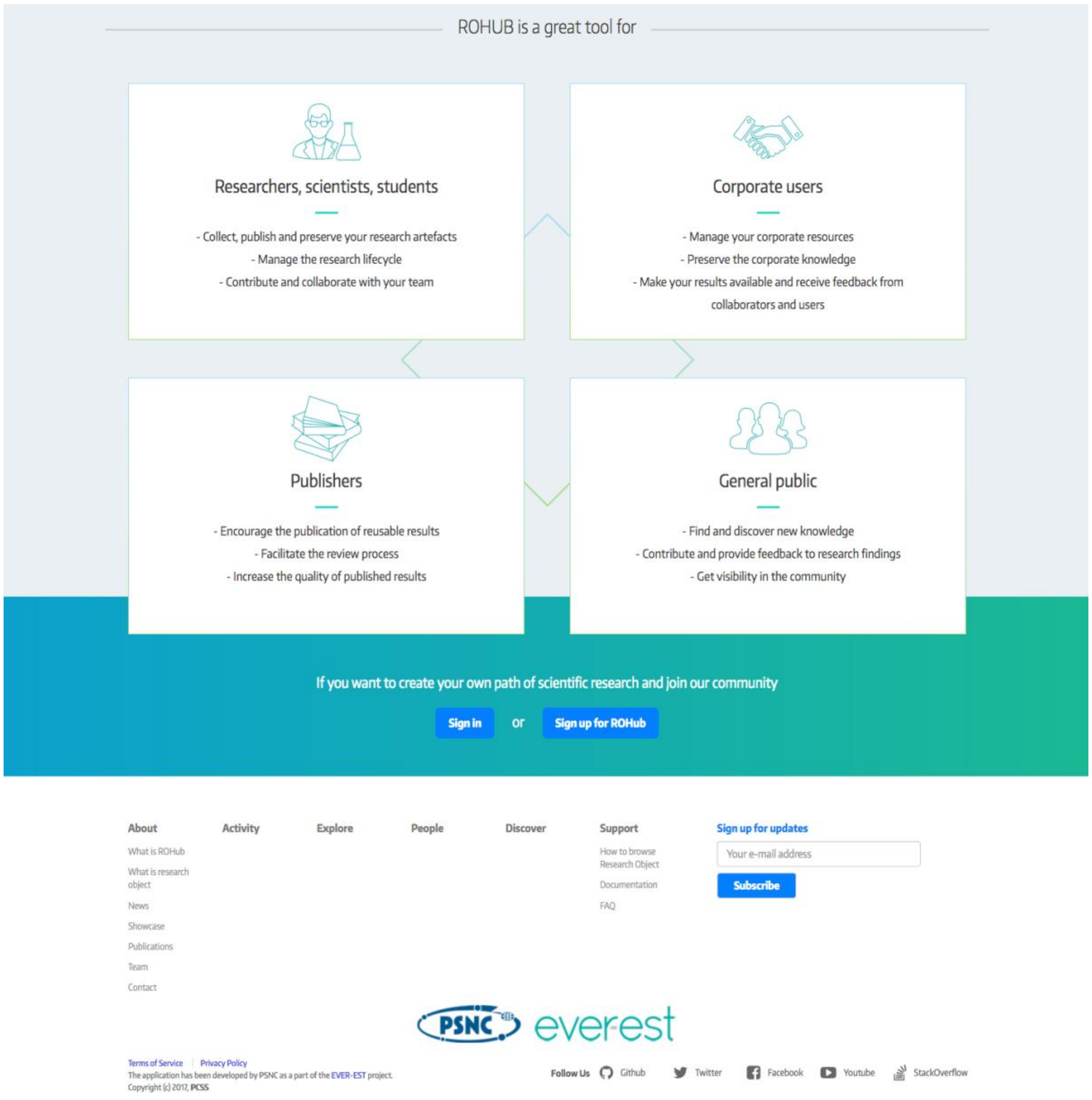
RESOURCE
Creation date: Tue Apr 04 2017
By: An annotation for resource COCONET_V3_InternationalProtectedSit...
Name : Description for
GeoChemistry
More

SNAPSHOTS
Creation date: Thu Mar 23 2017
By: Tim Aldridge
Name : DemoRunPoznan4/
Mathematics
More

ARCHIVE
Creation date: Tue Mar 14 2017
By: Simone Mantovani
Name : elisa/
Physics
More

All activities

Figure 3-7 ROHub portal main page (2/3)



ROHUB is a great tool for

Researchers, scientists, students

- Collect, publish and preserve your research artefacts
- Manage the research lifecycle
- Contribute and collaborate with your team

Corporate users

- Manage your corporate resources
- Preserve the corporate knowledge
- Make your results available and receive feedback from collaborators and users

Publishers

- Encourage the publication of reusable results
- Facilitate the review process
- Increase the quality of published results

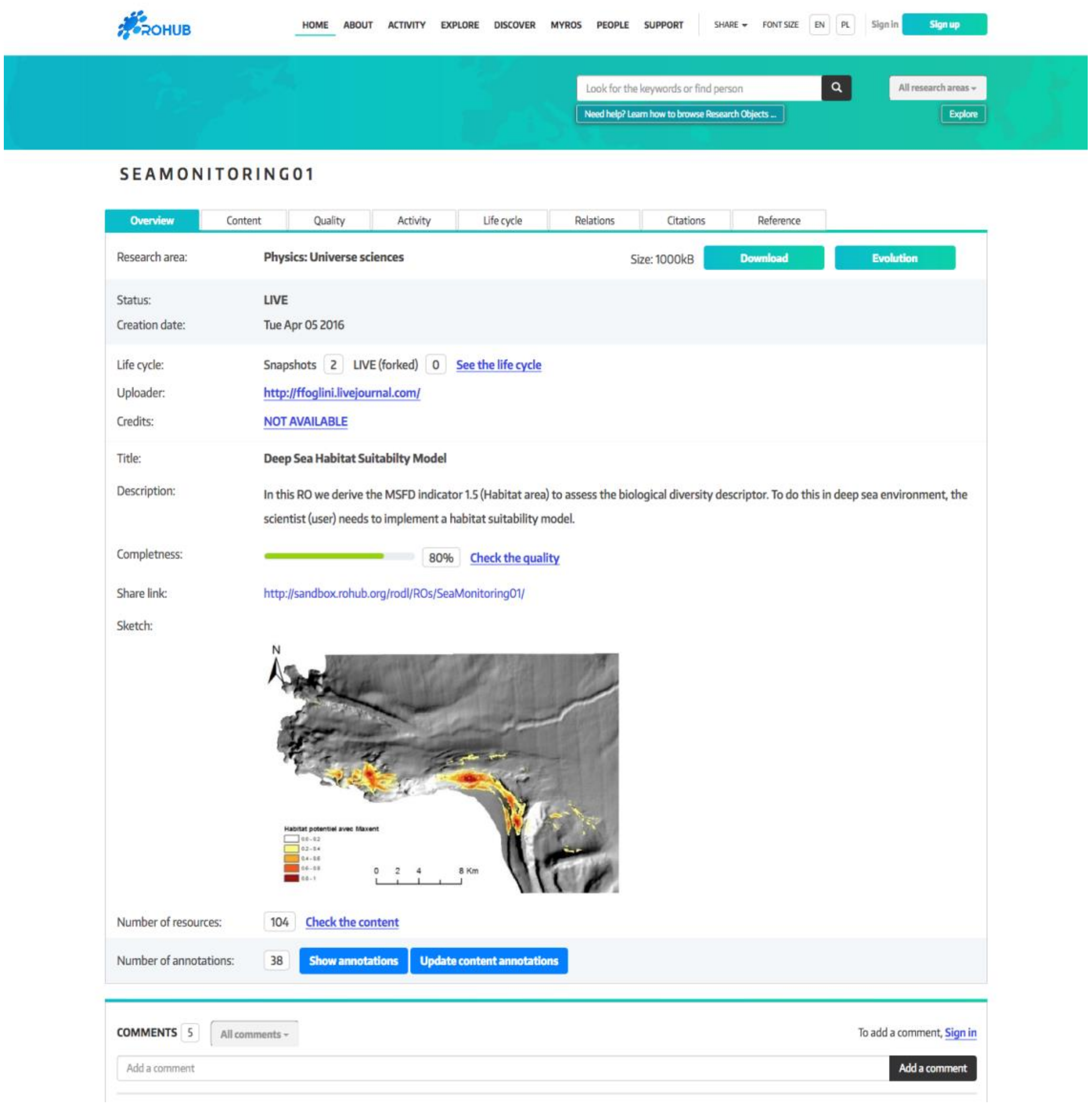
General public

- Find and discover new knowledge
- Contribute and provide feedback to research findings
- Get visibility in the community

If you want to create your own path of scientific research and join our community

The application has been developed by PSNC as a part of the EVER-EST project.
Copyright (c) 2017, PCSS

Figure 3-8 ROHub portal main page (3/3)



SEAMONITORING01

Research area:	Physics: Universe sciences — Size: 1000kB
Status:	LIVE
Creation date:	Tue Apr 05 2016
Life cycle:	Snapshots 2 LIVE (forked) 0 See the life cycle
Uploader:	http://ffoglini.livejournal.com/
Credits:	NOT AVAILABLE
Title:	Deep Sea Habitat Suitabilty Model
Description:	In this RO we derive the MSFD indicator 1.5 (Habitat area) to assess the biological diversity descriptor. To do this in deep sea environment, the scientist (user) needs to implement a habitat suitability model.
Completness:	80% Check the quality
Share link:	http://sandbox.rohub.org/rodl/ROs/SeaMonitoring01/
Sketch:	
Number of resources:	104
Number of annotations:	38

COMMENTS 5

To add a comment, Sign in

Figure 3-9 ROHub portal RO main page

3.4.7 Technological comparison between V1 and V2 versions of ROHub

Table 1 comparison of technologies between ROHub portal v1 and portal v2

ROHub Portal v1	**ROHub Portal v2**
Apache Wicket 6.9.0	Play Framework 2.5 + AngularJS v2
OpenID Authentication 2.0, updated to support OpenID Connect 1.0	OpenID Connect 1.0
Research Object Bundle 2013, updated to Bundle 1.0	Research Object Bundle 1.0
Solr 4.2, update to support OpenSearch 1.1	Solr 6 + OpenSearch 1.1

Jena 2.11	RDF4J 2.0

Table 2 comparison of capabilities between ROHUB portal v1 and portal v2

ROHUB Portal v1	**ROHUB Portal v2**
No modular design	Modular design
No visual components	Visual components
No UX design	UX design
No API	REST API

4 Common Services

The Common Services category represents the family of services, which are used by different elements of the EVER-EST infrastructure. It includes, by definition, a variety of services, from remote resources access services to management of complex semantic information. The implemented web-based services is loosely integrated into portals to constitute an inter-disciplinary infrastructure that can be easily extended beyond the initial Earth Science communities to other scientific domains.

The following common functionalities have been identified to ease the integration of new/existing components:

- Identity management;
- Auditing;
- Enterprise Service Bus.

4.1 Identity management

In computing, the identity management refers to the management of individual principals, their authentication and authorisation

Identity management can involve four basic functions:

- **User provisioning/removing**: User provisioning or account provisioning technology creates, modifies, disables and deletes user accounts and their profiles across IT infrastructure and business applications. Provisioning tools use approaches such as cloning, roles and business rules so businesses can automate on boarding, off-boarding and other administration workforce processes (for example, new hires, transfers, promotions and terminations). Provisioning tools also automatically aggregate and correlate identity data from HR, CRM, email systems and other "identity stores" Fulfilment is initiated via self-service, management request or HR system changes[29].
- **Authentication**: Authentication is the process of confirming the correctness of the claimed identity[30].
- **Authorisation**: Authorisation is the approval, permission, or empowerment for someone or something to do something[31].
- **Federated identity management**: Many companies want to share resources with their partners, but how can they do this when each business is a separate security realm with independent directory services, security, and authentication? One answer is federated identity. Federated identity helps overcome some of the problems that arise when two or more separate security realms use a single application. It allows employees to use their local corporate credentials to log on to external networks that have trust relationships with their company[32].

Each of the above functions presents many challenges when applied on distributed and heterogeneous systems.

[29] http://www.gartner.com/it-glossary/user-provisioning

[30] https://www.sans.org/security-resources/glossary-of-terms/

[31] https://www.sans.org/security-resources/glossary-of-terms/

[32] https://msdn.microsoft.com/en-us/library/ff359110.aspx?f=255&MSPPError=-2147217396

4.1.1 Authentication and federated identity management

The integration of systems often presents the problem on how the authentication is performed in each of the components. Usually systems own a proprietary user database. Sometimes the user has to create an identity on each system, using different password. The single-sign-on started the work on standardisation. The major protocols for authentication have been evaluated during the first deployment phase of the project, OpenID Connect and SAML2.0 have been chosen, criteria for choosing will be illustrated further.

4.1.2 Authorisation

There are different authorisation models, which are implementing different authorisation features with different granularity:

- **Access Control List (ACL):** the oldest and the simplest. Each resource on a system to which access should be controlled, referred to as an object, has its own associated list of mappings between the set of entities requesting access to the resource and the set of actions that each entity can take on the resource.
- **Role Based Access Control (RBAC):** the most used. Access to a resource is determined based on the relationship between the requester and the organisation or owner in control of the resource; in other words, the requester's role or function determines whether access is granted or denied.
- **Attribute Based Access Control (ABAC):** the access control decisions are made based on a set of characteristics, or attributes, associated with the requester, the environment, and/or the resource itself. Each attribute is a discrete, distinct field that a policy decision point can compare against a set of values to determine whether or not to allow or deny access.
- **Policy Based Access Control (PBAC):** The way to externalise authorisation. There is a central repository where the access policies are contained, and governed. The policies and the model are usually following the XACML (*eXtensible Access Control Markup Language*) OASIS standard.

4.1.3 Provisioning/removing

In a centralised/integrated system, the user creation/deletion/modification must be propagated to every subsystem. Modern integrated systems are providing self-service registration, self-reset password features.

4.2 A service oriented architecture

There are some basic principles behind Service Oriented Architecture, and how the common services help to meet the following smart objective: "*Deploy a VRE based on a Service Oriented Architecture (SOA) tailored for Earth Science.*" will be addressed within this chapter.

4.2.1 Service oriented architecture main principles

Thomas Erl, in *Service-Oriented Architecture: Concepts, Technology, and Design*, defines the Service-Orientation design principles as follows[33]:

- **Standardised Service Contract**: Services within the same service inventory are in compliance with the same contract design standards.
- **Service Loose Coupling**: Service contracts impose low consumer coupling requirements and are themselves decoupled from their surrounding environment.

[33] http://serviceorientation.com/serviceorientation/index

- **Service Abstraction**: Service contracts only contain essential information, and information about services is limited to what is published in service contracts.
- **Service Reusability**: Services contain and express agnostic logic and can be positioned as reusable enterprise resources.
- **Service Autonomy**: Services exercise a high level of control over their underlying runtime execution environment."
- **Service Statelessness**: Services minimise resource consumption by deferring the management of state information when necessary.
- **Service Discoverability**: Services are supplemented with communicative metadata by which they can be effectively discovered and interpreted.
- **Service Composability**: Services are effective composition participants, regardless of the size and complexity of the composition.

Some of these principles are implemented to the design best practices e.g.:

- **Standardised Service Contract**: the web services will adhere to consolidate interfaces, like OGC standards for the catalogue and data processing (WMS, WCS, WPS), the catalogues will adhere to OpenSearch specifications.
- **Service Reusability**: The main idea behind the ROHub design is that objects are meant to be re-usable.
- **Service Abstraction**: Usage of standards like the WPS (OGC Web Processing Service)
- **Service Statelessness**: Processing services are not saving any state. The workflow will carry the service output through the entire process.
- **Service Composability**: This is the idea behind the ROHub workflows. Services can be orchestrated and composed together to build new workflows.
- **Service Discoverability**: It will be analysed if a Service Registry will be used in the project.
- **Service Autonomy**: Services can be deployed in sandbox[34].

4.2.2 The WSO2 Identity Service

WSO2 Identity Server provides secure identity management for enterprise web applications, services, and APIs by managing identity and entitlements of the users securely and efficiently.

The Identity Server enables to create, maintain and terminate user accounts along with user identities across multiple systems including Cloud applications. When there are multiple applications that require authentication, users should be able to log in at one place and still have seamless access to all the other applications.

Additionally, the Identity Server brings about a new and improved approach to federation. There is a centralised Identity as a Service Provider. It is still an overall n-to-n relationship. There is a 1-to-n relationship from a federation partner to consumer services (where multiple consumer services rely on a single centralised federated Identity Provider for security) and a 1-to-n relationship from consumer service to federation partners (where a single consumer service can rely on multiple Identity providers for security). This model ensures greater efficiency.

Some of the features that the Identity Server provides are enlisted here:

Inbound Authenticators

[34] https://en.wikipedia.org/wiki/Service_autonomy_principle

The responsibility of inbound authenticators is to identify and parse all the incoming authentication requests and then build the corresponding response. A given inbound authenticator has two parts.

1. Request Processor;
2. Response Builder.

For each protocol supported by WSO2 IS, there should be an inbound authenticator. This architecture component includes inbound authenticators for SAML 2.0, OpenID, OpenID Connect, OAuth 2.0, and WS-Federation (passive). In other words, the responsibility of the SAML 2.0 request processor is to accept a SAML request from a service provider, validate the SAML request and then build a common object model understood by the authentication framework and handover the request to it. The responsibility of the SAML response builder is to accept a common object model from the authentication framework and build a SAML response out of it. Both the request processors and the response builders are protocol aware, while the authentication framework is not coupled to any protocol.

Single Sign On

SSO enables users to provide their credentials once and obtain access to multiple applications. In SSO, the users are not prompted for their credentials when accessing each application until their session is terminated.

Local Authenticators

The responsibility of the local authenticators is to authenticate the user with locally available credentials. This can be either username/password or even IWA (Integrated Windows Authentication). Local authenticators are decoupled from the Inbound Authenticators. Once the initial request is handed over to the authentication framework from an inbound authenticator, the authentication framework talks to the service provider configuration component to find the set of local authenticators registered with the service provider corresponding to the current authentication request.

Once the local authentication is successfully completed, the local authenticator will notify the framework. The framework will now decide no more authentications is needed and hand over the control to the corresponding response builder of the inbound authenticator.

Local custom authenticators can be developed and plugged into the Identity Server.

Outbound/Federated Authenticators

The responsibility of the federated authenticators is to authenticate the user with an external system. This can be with Facebook, Google, Yahoo, LinkedIn, Twitter, Salesforce or any other identity provider. Federated authenticators are decoupled from the Inbound Authenticators. Once the initial request is handed over to the authentication framework from an inbound authenticator, the authentication framework talks to the service provider configuration component to find the set of federated authenticators registered with the service provider corresponding to the current authentication request.

A federated authenticator has no value unless it is associated with an identity provider. The Identity Server out-of-the-box supports SAML 2.0, OpenID, OpenID Connect, OAuth 2.0 and WS-Federation (passive). The SAML 2 .0 federated authenticator itself has no value. It has to be associated with an Identity Provider. Google Apps can be an identity provider - with the SAML 2.0 federated authenticator. This federated authenticator knows how to generate a SAML request to the Google Apps and process a SAML response from it.

Identity Provisioning

Identity provisioning plays a key role in propagating user identities across different SaaS providers. Provisioning is the process of coordinating the creation of user accounts, e-mail authorisations in the form of rules and roles, and other tasks such as provisioning of resources associated with enabling new users.

SCIM - System for Cross-domain Identity Management

The System for Cross-domain Identity Management (SCIM) specification is designed to make managing user identities in the WSO2 Identity Server easier. SCIM is an emerging open standard, which defines a comprehensive REST API along with a platform neutral schema and a SAML binding to facilitate the user management operations across SaaS applications; placing specific emphasis on simplicity and interoperability as well.

Inbound provisioning

Inbound provisioning focuses on how to provide users to the Identity Server. Out-of-the-box, the Identity Server supports inbound provisioning via a SOAP-based API as well as the SCIM 1.1 API. Both the APIs support HTTP Basic Authentication. If you invoke the provisioning API with Basic Authentication credentials, then where to provision the user (to which user store) will be decided based on the inbound provisioning configuration of the resident service provider.

The SCIM API also supports OAuth 2.0. If the user authenticates to the SCIM API with OAuth credentials, then the system will load the configuration corresponding to the service provider who owns the OAuth client id. If you plan to invoke the SCIM API via a web application or a mobile application, it is highly recommended to use OAuth instead of Basic Authentication. You simply need to register your application as a service provider in Identity Server and then generate OAuth keys.

Outbound Provisioning

Outbound provisioning talks about provisioning users to external systems. This can be initiated by any of the following:

- Inbound provisioning request (initiated by a service provider or the resident service provider);
- JIT provisioning (initiated by a service provider);
- Adding a user via the management console (initiated by the the resident service provider);
- Assigning a user to a provisioning role (initiated by the the resident service provider).

WSO2 Identity Server supports outbound provisioning with the following connectors. You need to configure one or more outbound provisioning connectors with a given identity provider, and associate the identity provider with a service provider. All the provisioning requests must be initiated by a service provider - and will be provisioned to all the identity providers configured in the outbound provisioning configuration of the corresponding service provider.

- SCIM;
- SPML;
- SOAP;
- Google Apps provisioning API;
- Salesforce provisioning API.

4.2.2.1 The WSO2 identity server APIs

The WSO2 Identity Server provides several APIs that can be used:

- **Identity Application Management APIs**[35]: The identity application management API can be used to add and configure service providers in the WSO2 Identity Server.
- **Identity Provider Management API**[36]: The identity provider management service API can be used to add and configure identity providers in the WSO2 Identity Server.
- **SCIM APIs**[37]: The SCIM API can be called in order to perform various tasks in the WSO2 Identity Server. For simplicity, cURL commands are used in this example to send CRUD requests to the REST endpoints of Identity Server.
- **User Role Management APIs**[38]: WSO2 Identity Server provides a Web service API by the name called RemoteUserStoreManagerService for user role management. If your application needs a user role management function, you can directly integrate with Identity Server instead of dealing with the user store.
- **User Information Recovery Service**[39]: those are service methods to recover information about the users, like get all challenge questions or get captcha code.

Product pages can be consulted for additional information.

4.2.2.2 The WSO2 identity server in EVER-EST

WSO2 Identity Server provides the Identity Management functionality to the EVER-EST Web Application, services and APIs. User Provisioning and Authentication functionalities have been identified as key functionalities in EVER-EST as well as the Single Sign On feature that plays a fundamental role in enabling interoperability among the wide range of application and services offered by EVER-EST. All the Identity Management functionalities are centralized to support the coordinated usage of the different EVER-EST components and allow their interoperability.

Centralisation has considerable benefits, such as easier administration and potentially faster development cycles for new apps.

User Provisioning

The user provisioning functionality in EVER-EST is driven by the Self Sign Up functionalities offered by the WSO2 Identity Server.

An unregistered user who tries to login has the possibility to perform a self-sign up in the EVER-EST Identity Server. A user can register himself as a Citizen or as a member of one of the VRCs.

The users belonging to an EVER-EST Virtual Research Community need to be approved by a VRC administrator.

The users not belonging to a VRC can register himself as Citizen. In both upon the successful filling of the registration form, the user receives an email to the email address provided during registration for verifying the account and get it activated. The self-sign up process creates the user and locks the user account until the user confirmation is received.

[35] https://docs.wso2.com/display/IS510/Identity+Application+Management+API

[36] https://docs.wso2.com/display/IS510/Identity+Provider+Management+API

[37] https://docs.wso2.com/display/IS510/SCIM+APIs

[38] https://docs.wso2.com/display/IS510/Managing+Users+and+Roles+with+APIs

[39] https://docs.wso2.com/display/IS510/User+Information+Recovery+Service

Once the user is created in the Identity Server the EVER-EST applications can implement an own Just in Time (JIT) user provisioning to complete, for their necessity, the user information exposed for them by the IS.

Authentication

Among the variety of authentication protocols OpenID Connect and SAML2.0 have been chosen for EVER-EST.

OpenID Connect has been selected as it is the new emerging standard for single sign-on and identity provision on the internet.

OpenID Connect is a lightweight identity verification protocol built on top of modern web standards (OAuth 2.0, REST and JSON). It uses simple JSON-based identity tokens (JWT), delivered via the OAuth 2.0 protocol to suit web, browser-based and native / mobile apps.

Along with OpenID Connect the use of SAML2.0 protocol has become necessary for the integration of Seafile COTS that represents the EVER-EST Personal File Storage. Seafile can authenticates users using Shibboleth. Shibboleth software implements the SAML2.0 identity standards.

The Identity Server takes care of transformation of protocols and tokens.

Authorisation

Authorisation tasks in EVER-EST are mainly managed by the single application and not centralised also if there is the necessity to protect via OAuth2.0 authorisation protocol some of the resources. The OAuth protocol provides clients with a ‘secure delegated access’ to server resources following a specific authorisation flow.

Exploiting the User Role Management API offered by the WSO2 Identity Server the EVER-EST applications that need to validate the access to their resources can be provided of credentials for using the APIs offered by the validation web service provided by the WSO2 IS.

The ROHUB APIs are the resources that need protection. When an API consumer, in our concern the VRE Portal, want to access these resources the API owner, in our concern the ROHUB, following the OAuth2.0 specification and using the exposed API can validate the access to the resources.

Following the OAuth2.0 specification, the API owner (resource owner) owns the responsibility to prevent unauthorised users from accessing an API (or a resource in general). For that to be possible the specification prescribes that API consumers send a valid OAuth2.0 access token along with every API call. Such access token must be sent as the HTTPS request parameter. Once the API Server gets the call it needs to validate the access token against the OAuth2.0 Authorisation server, before processing the request and replying to the consumer API.

The following figure provides an overview of the process; it contains the authentication process, steps 1-8, and the protected API call, steps 9-12.

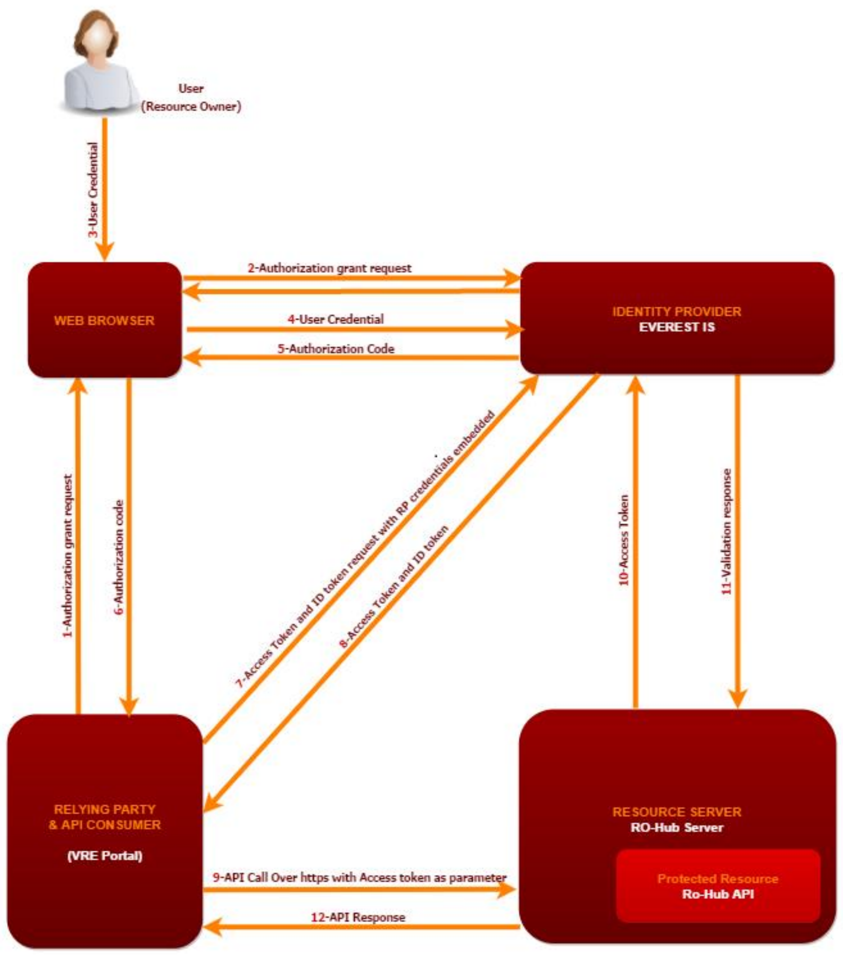


Figure 14 Authentication Process and API protection

Single Sign On

In EVER-EST there are multiple applications that require authentication, users should be able to log in at one place and still have seamless access to all the other applications. The WSO2 Identity Server takes care of transformation of protocols and tokens exploiting the Inbound, Outbound and Local Authenticator features.

Once the user provides his/her credentials it's automatically authenticated on all application enabling a Single Sign On scenario between multiple heterogeneous authentication protocols.

The EVER-EST applications and services have to be highly interconnected to allow a seamlessly data exchange among disparate IT systems. The SSO functionality allows users from the communities to perform their duties in a fraction of the time accelerating collaboration.

SSO in EVER-EST has also the advantage to ease auditing of the applications access.

The SSO solution strengthens security, streamline application access, and improves user experience.

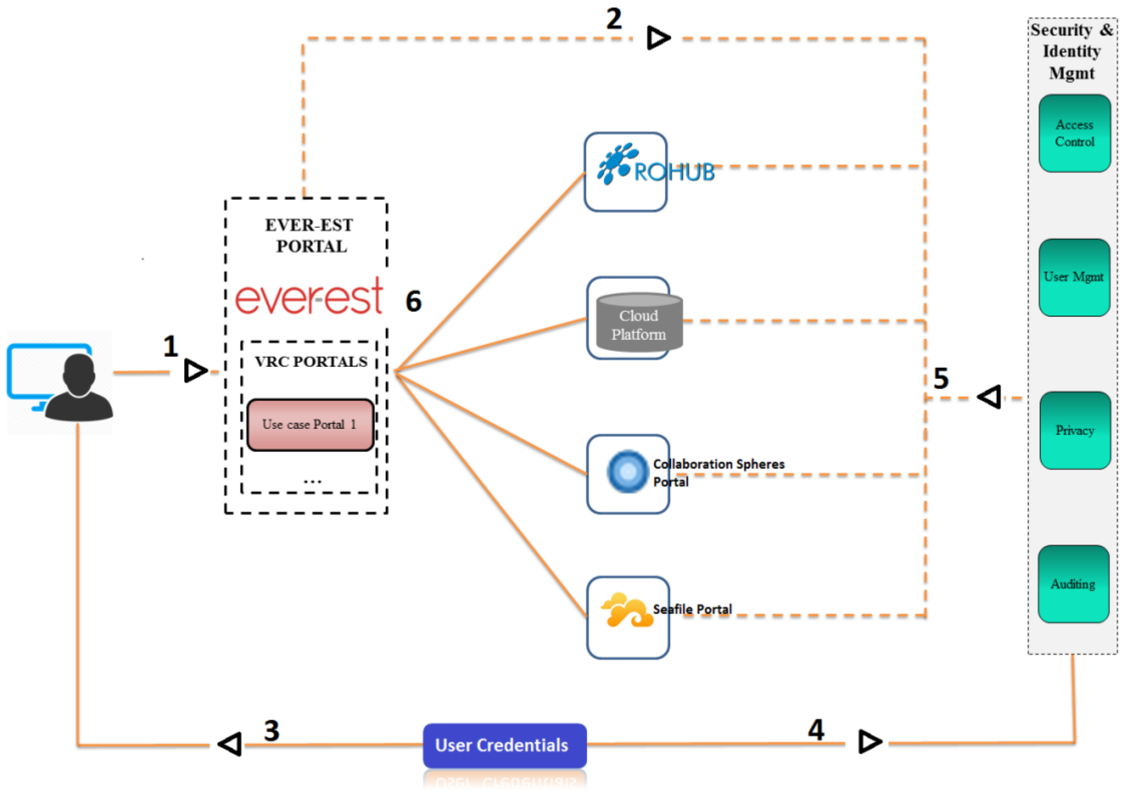


Figure 15 SSO flow scheme

The figure above represents a simplified version of a typical SSO flow in EVER-EST architecture.

1 - User connects to the EVER-EST portal;

2 - EVER-EST portal transmits an authentication request to the Security and Identity Management;

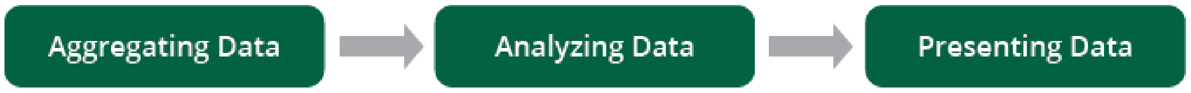
3 - User is redirected to the Security and Identity Management for authentication;

4 - User submit credentials to the Security and Identity Management, who in turn authenticates the user;

5 - User is redirected back to the EVER-EST Portal accompanied by a token confirming positive authentication and bearing user information and access right; 6 - User starts using the services being already authenticated.

4.2.3 WSO2 data analytics server

WSO2 Data Analytics Server (DAS) is a lean, fully open source, complete solution for aggregating, analysing data and presenting information about business activities. It provides real-time visibility into distributed complex systems, including service-oriented architecture (SOA) processes, transactions and workflows.



It is designed to treat millions events per second, and is therefore capable to handle Big Data volumes. The capabilities of the Data Analytic server, integrated with the Enterprise Service bus, can provide EVER-EST with the functionalities to analyse in real time and batch service usage and statistics.

Data analytics refer to aggregating, analysing and presenting information about business activities. Aggregation refers to the collection of data, analysis refers to the manipulation of data to extract information, and presentation refers to representing this data visually or in other ways such as alerts. WSO2 DAS is structured around a workflow model consisting of three main phases: collecting, analyzing and decision making and finally communicating.

Data, which needs to be monitored or processed sequentially, shall go through these modules. The WSO2 DAS architecture reflects this natural flow in its very design as illustrated by

Figure **16** below.

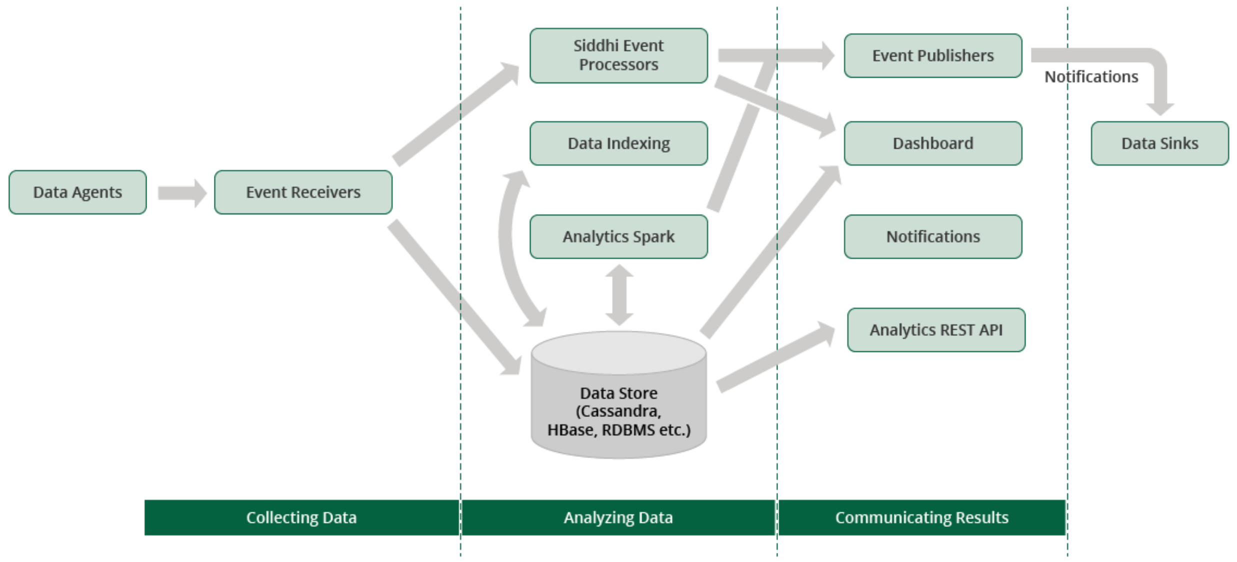


Figure 16 The WSO2 DAS dataflow

In addition to the Event Receivers, WSO2 DAS facilitates REST API based data publishing. The REST API can be utilised with web applications and web services. For more information on REST API, see Analytics REST API Guide[40].

4.2.4 The enterprise service bus

The role of the Enterprise Service Bus (ESB) in the EVER-EST VRE's Service oriented architecture is that of an enabler of communications among systems and components both currently involved in the VRE and those which may be deployed during an evolution phase of the VRE. Such enablement consists in leveraging extensive connectivity technologies supported by WSO2 ESB thus making it possible to integrate very heterogeneous components. At the same time the use of ESB within the VRE has been inspired by a low-intrusion philosophy,

[40] https://docs.wso2.com/display/DAS300/Analytics+REST+API+Guide

whereby components endowed with a mature level of communications technology are allowed to interoperate directly. This is the key point for an architecture that performs fast, avoids unnecessary complexities, and evolves easily. From this perspective the service oriented model intrinsic in ROHub, Terradue Cloud Platform, and Seafile are exposed to VRC Portals for a direct interaction with no intermediary service bus.

The detailed analysis of existing components carried out during the design phase has thus reduced the need for ESB as a communication enabler.

That said, WSO2 ESB is still needed on the VRE as a backbone where systems, which may be needed in VRE evolutions, must be plugged in in order for them to enter the VRE ecosystem if their functionalities do not conform to the Service Oriented Architecture model. The mediation of the ESB, through the implementation and deployment on it of wrapper services, provides for adaptation of such unconventional systems thus shielding VRE's service oriented architecture from being corrupted every time evolutions happen. Conversely, WSO2 ESB is used as a data collector in the preparatory phase of data analytics. The usage pattern in this scenario is as follows. Data for KPI measurement, generated by the systems composing VRE, get collected by the ESB mainly through listening on log files and data repositories while such data is produced or as a second option (for systems that do not have a data repository shareable for the purpose) through receiving real-time data feeds via HTTP directly from the systems. The ESB once receives the data sends such collected information to WSO2 Data Analytics Server where data analysis and KPI monitoring can take place. The principal reason for this approach, where ESB is an intermediary component between data sources and WSO2 Data Analytics Server, lies in helping to diminish the burden that systems would have if they were needed to directly feed KPI information in addition to their main job of consuming or providing business services. An additional reason comes from the opportunity to have at disposal the steadily growing family of WSO2 ESB connectors able to talk to disparate range of systems and applications. This can be taken advantage of in future evolution of VRE in cases where KPI data need to be augmented with related information available on external systems. In this light, ESB plays a key role in assessing the successful exploitation of the EVER-EST VRE allowing to continually evaluate the project's achievements and to re-direct activities as the need arises.

4.3 Data discovery and access

One of the core functionalities that EVER-EST must provide its users and communities with is the capability to search and retrieve data from within the Virtual Research Communities portals. As for most of the Earth Science data catalogues, functionalities must be provided to allow each VRC to interact with a 2D/3D map of the world, specify a region of interest and time constraints, to search and retrieve relevant datasets they process and possibly share with their community of interest, together with their results. Several EVER-EST related projects have made significant and recognised contributions to the design and implementation of a multi-disciplinary platform where data discovery and access where a major challenge. As an example, the platform developed by the ENVRI project provided discovery capabilities as well as fast and easy access for the heterogeneous data sets scattered across the ESFRI communities. The ENVRI approach, based on the OpenSearch paradigm, is used and re-adapted by EVER-EST to facilitate the process of resource discovery and access eventually leveraging on networks of cross-linked information.

On the client side, the user interface has been re-adapted from the pre-existing work made in the frame of the EarthServer project on the MEA platform[41] to provide a user-friendly mechanism for data search, discovery, access and interaction. On the server side, the system will progressively point to an ensemble of heterogeneous Earth Science catalogues, which will be made accessible from the platform during the project lifetime. These will include since the early stages the FedEO catalogue, which is currently the main entry

[41] http://eodataservice.org

point for Earth Observation data coming from the several Space Agencies, including ESA, DLR, NASA and JAXA.

By providing access catalogues of both on going and legacy Earth Observation missions, the VRE significantly addresses the needs of the VRCs participating to the project. On top of that, each of the communities will need tailored discovery capabilities on specific data catalogues. For instance, specific work will be carried out to guarantee the VRE access to the CoCoNet catalogue[2] – used by the Sea Marine VRC. During the project lifetime, and in particular during the VRE data population phase, additional catalogues will be integrated into the search and retrieval module to include, for instance, specific catalogues for in situ data. Catalogues, related technologies and user requirements will be elicited by the continuous interaction between the technical team in charge of the VRE implementation and the single Virtual Research Communities.

As the core of the VRE infrastructure is represented by ROHub, the concept of Research Object will be the paradigm allowing VRE users to describe, save and share their observations, and an ad-hoc development will guarantee search and retrieval capabilities for Research Objects within this module. Therefore, in addition to RO search capabilities directly provided by ROHub (see section 3.4.3), the Research Objects will be discoverable via geo-referenced searching criteria based on Open Search Geo and Time Extensions. It shall also be noted that EVER-EST does not move or copy data or resources to create and maintain the catalogue entries. It will simply hold the metadata description for the resources and a link for direct access to the original location of the resources. Authorised users will then be able to download the resources according to the access policy of each data provider.

Data discovery approach is based on an INSPIRE compliant, two-step discovery process supported by two levels of catalogue:

1) A main catalogue, or aggregator node, containing high-level metadata information on the data collections available in the different centres;

2) Second level catalogue(s), which may be hosted at the data centre, containing detailed information about the individual products.

This complexity is completely hidden to the user. This approach makes the system scalable and allows the number of resources available for discovery to be increased without significantly affecting the response time of the system. The VRE metadata model will make use of the relevant standards (e.g. ISO 19115, Dublin Core, DIF) and is flexible and extendable to meet the needs of all the communities involved in the use cases in order to add and store new metadata identified as during the project lifetime.

4.3.1 The OpenSearch protocol for data discovery

Catalogues integration is performed via OpenSearch interface[42], although some specific adaptation modules have been developed to guarantee interoperability with non-OpenSearch based catalogues, where needed. EVER-EST combines the data discovery and access approach with the preservation framework from the SCIDIP-ES project to create a knowledge layer that is based on bridge ontology and web services. More information about these features can be found in the Preservation section of the document.

Starting in 2008, the OpenSearch protocol was pushed towards standardisation in the geospatial context by specifications of the Open Geospatial Consortium (OGC). This interface was recognised and adopted last year by the OGC as the preferred baseline for the catalogue services. ESA selected this interface for the implementation of the Agency's Next Generation User Services for Earth Observation (ngEO), NASA applied

[42] http://www.opensearch.org

it to the newly deployed system of Earth Observing System (EOS) Clearing House (ECHO) and their interface to the Federation of Earth Science Information Partners (ESIP) that enables the discovery and access to the totality of NASA archives. The CEOS WGISS Integrated Catalog (CWIC) recently established a common interoperability best practice of OpenSearch in order to allow for standardised and harmonised access to metadata and data of CEOS agencies.

OpenSearch provides a standard way of describing search engine capabilities so that they can be leveraged by search client applications. The OpenSearch Geo, Time and Earth Observation Extensions adds a simple way to configure search engines for spatial and temporal queries over distributed repositories of contents having geographic and time properties, and for syndication of these search results in one large index.

4.3.1.1 CEOS best practices

Using the best practices for search services with OpenSearch with Geo, Time and EO extensions as defined by the CEOS (Committee on Earth Observation Satellites) allows a standardised and harmonised access to metadata and data of world's satellite Earth Observation data providers. It also provides a framework to ease the discovery (i.e. search) and retrieval (i.e. access) of EO data (remote-sensing and in situ). This solution facilitates the aggregation of results between disparate Earth Data providers via a common standard by allowing search engine discovery (using OpenSearch Description Documents) and, mostly importantly, facilitates smooth integration between different server implementations.

CEOS best practices constitute the reference document to manage the discussion between the technical team and the single VRCs when it comes to the integration via OpenSearch of additional catalogues into the VRE. Born to promote the adoption of the OpenSearch paradigm among Earth Data providers, the document is used as the main reference to define requirements, remove ambiguities, and share a common terminology between the stakeholders. The document provides an overview of the OpenSearch Description Document (OSDD), which is the central element to allow clients access to search engines with no a priori knowledge of the interface. The agreement and adoption of the OSDD by single catalogues is the mechanism guaranteeing catalogue visibility and discoverability via OpenSearch from the VRE. In addition, the Best Practices define with a high level of detail the set of Required/ Recommended/ Optional features that each data provider shall implement to adhere to OpenSearch servers implementation standards. Those roles will be discussed for potential adoption, during the project lifetime, between EVER-EST consortium partners.

4.3.2 Discovering Earth Science resources

This chapter provides an overview of the main catalogues that will be made accessible from within the VRE. As the core of the EVER-EST infrastructure is based on ROHub and the elements shared by scientists will have the shape of Research Objects, the capability to search and discover RO's via geo/time parameters is described hereafter.

4.3.2.1 Discovering Research Objects

One key feature of EVER-EST will be the possibility to discover and directly access the different Research Objects through Geographical and Timing criteria. It is worth mentioning that not all Research Objects may refer to geo-referenced elements: i.e. the RO describing an algorithm. So this feature is applicable only when the Research Object has precise spatial and temporal elements (e.g. study of the maritime pollution in 2015 over the Mediterranean Sea).

The ROHub will expose an OpenSearch interface for searching and publishing search results in a standard and accessible format, and in line with the Earth Science domain specific needs, ROHub will also implement the OGC OpenSearch Geo and Time Extensions to support objects having a geographical position and point in time.

Those two attributes are the ones that are common to all EO data, and are mandatory in order to render the objects in a 2D map. Such information are captured and aggregated in the Research Object as annotations based on the RO model, and are associated to the RO itself and/or to the individual resources (e.g., dataset).

Since the ROHUB index is built on Solr, the SpatialRecursivePrefixTreeFieldType (RPT) is used in order to implement the spatial searches. Solr offers the following functionalities:

- Query by polygons and other complex shapes, in addition to circles & rectangles;
- Multi-valued indexed fields;
- Ability to index non-point shapes (e.g. polygons) as well as point shapes;
- Rectangles with user-specified corners that can cross the dateline;
- Multi-value distance sort and score boosting;
- Well-Known-Text (WKT) shape syntax (required for specifying polygons & other complex shapes);
- Heatmap grid faceting capability.

4.3.2.2 Discovering Earth Observation data - FedEO

All Virtual Research Communities participating to the EVER-EST initiative will make use of Earth Observation data within their Use Cases. By providing direct access to FedEO and implementing feature for the Sentinel Hub discovery, the VRE will guarantee full discoverability of the main Earth Observation catalogues which are available, including ESA, EUMETSAT and DLR.

The FedEO Clearinghouse provides a unique access point for discovering, ordering and accessing Earth Observation (EO) dataset series and datasets. The Federated Earth Observation Missions (FedEO) initiative was initially an OGC pilot activity aimed to refine OGC specifications relevant to Earth Observation.

The same OGC specifications have subsequently been implemented in support of the Copernicus programme, in particular to provide interoperable access to catalogues from European Earth Observation Missions and support the subsequent ordering of data in support of the European Commission's Copernicus Service Projects. The resulting interoperable catalogue, the FedEO Clearinghouse, is providing discovery, ordering and online data access services for space based Earth Observation missions from over 25 years, as a potential contribution to GEO GEOSS.

The FedEO Clearinghouse brings together the catalogues from ESA, e-GEOSS, EUMETSAT, DLR, DMC, DMI, MDA, SPOT and VITO, as well as the catalogue of data contributed to the International Disaster for Space and Major Disasters and access to Alaska Satellite Facility and the CEOS WGISS Integrated Catalogue (Figure 17).

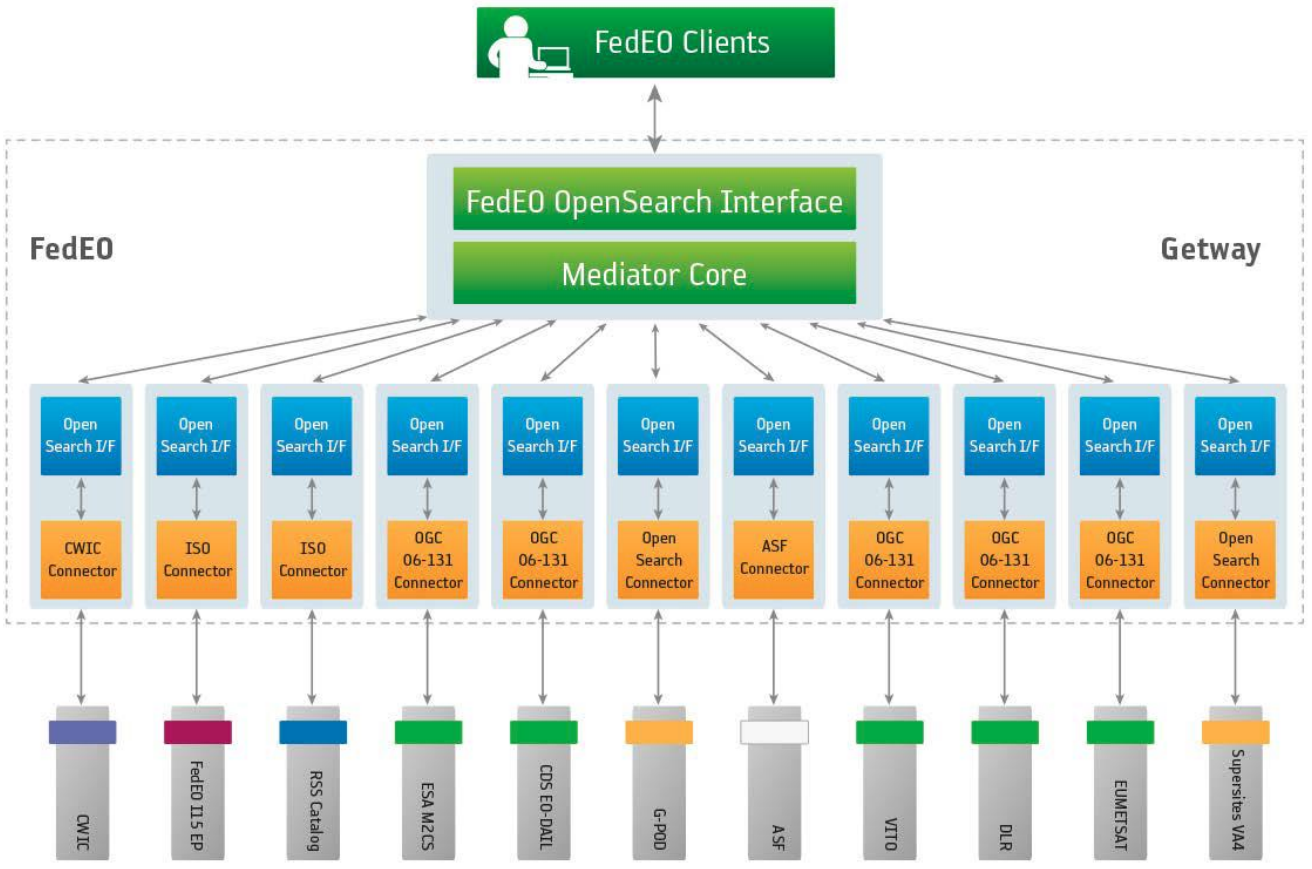


Figure 17 FedEO schema

FedEO is built around the concept of dataset series. It helps to consider each dataset series as a set of metadata records. One such metadata record can either describe another dataset series or an individual dataset (i.e. product). Each dataset series is identified by an identifier, which allows addressing search requests to a particular dataset series and receiving metadata records as part of the search response.

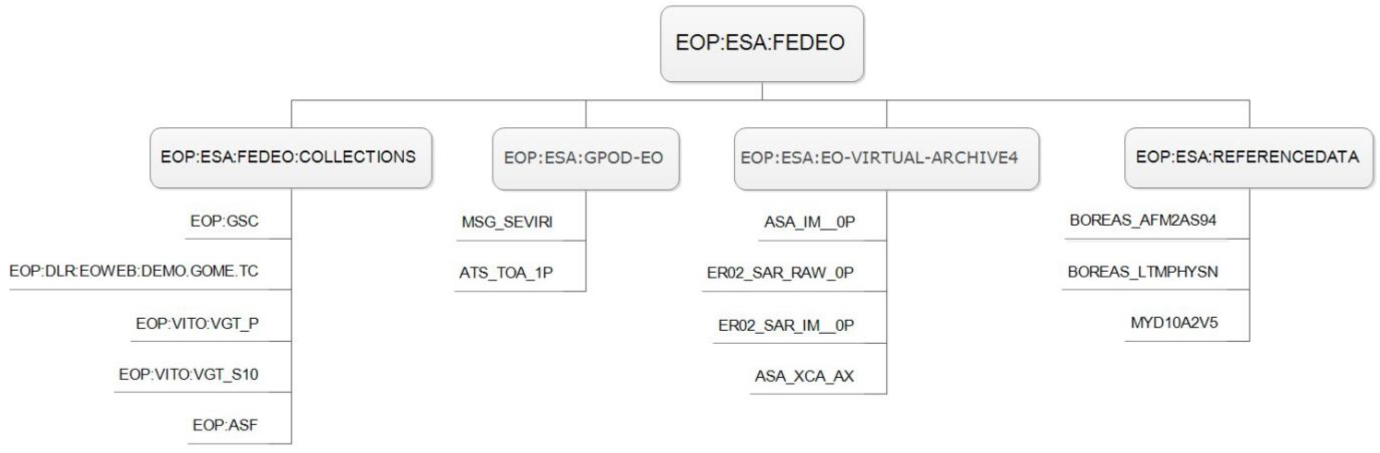


Figure 18 FedEO Data Hierarchies

Not all dataset series share the same list of search parameters. If a search parameter is not supported by a dataset series, it will in most cases by silently ignored. FedEO provides an OpenSearch interface aligned where possible with the SearchRetrieve operation for retrieving dataset series or dataset metadata. The range of parameters accepted by this operation and the media types returned by the operation are described in the FedEO OpenSearch Description Document (OSDD), which can be found at: http://fedeo.esa.int/opensearch/description.xml.

The OSDD is applicable to all FedEO dataset series. The complete client user guide that is used to access FEDEO from the VRE is available at the following link:

https://wiki.services.eoportal.org/tiki-download_wiki_attachment.php?attId=3699

The Sentinels Scientific Data Hub (SCI-HUB) guarantee free and open access to the Rolling Archive of Sentinel-1 and Sentinel-2 user products, and will be the mean to actually access the data discovered via FedEO. Sentinel-1 products will provide for download via HTTP in the .ZIP archive file format. The SCI-HUB allows a maximum of 2 concurrent downloads per user to ensure a download capacity for all users.

The Data Hub also exposes two dedicated Application Program Interfaces (API) for accessing the EO data stored in the rolling archive. The APIs are:

- Open Data Protocol;
- OpenSearch.

The Open Data Protocol (OData) interface allows accessing the EO data stored on the rolling archive. This protocol is based on top of the well-supported HTTPS/ REST transfer protocol that can be handled by a large set of client tools as simple as common Web browsers, download-managers or computer programs such as *cURL* or *wget*. The Odata protocol provides easy access to the Data Hub and can be used for building URI for performing search queries and product downloads offering to the users the capability to remotely run scripts in batch mode.

The OpenSearch has been described in section 4.3.1 and is complementary to the OData. In fact, OpenSearch can be used to complementary serve as the query aspect of OData, which provides a way to access, identified or located results and download them. The API Hub may be accessed through the URL https://scihub.copernicus.eu/apihub/. This implies that the OpenSearch API is published at https://scihub.copernicus.eu/apihub/search and the OpenData API is published at https://scihub.copernicus.eu/apihub/odata/v1. The API Hub is managed with the same quota restrictions, ie. a limit of two parallel downloads per user. Discover the list of the products stored in the archive.

4.3.2.3 The CoCoNet catalogue

In addition to the Earth Observation data provided by FedEO or the Sentinel Hub (described in sections 4.2 and 4.3), the Sea Monitoring Use Case makes wide use of the CoCoNet and EMODnet catalogues.

CoCoNet is a European project that produces guidelines to design, manage and monitor network of MPAs and Ocean Wind Farms. The project covers a high number of countries and involves researchers covering a vast array of subjects, developing a timely holistic approach and integrating the Mediterranean and Black Seas scientific communities through intense collective activities and a strong communication line with stakeholders and the public at large. The aim of this project is to provide a common framework for marine data management and final synthesis of the outcomes of different scientific topics from heterogeneous sources. An integrated Geodatabase and a WebGIS system will be the linking tool for all partners, regions and thematic research. It will involve the entire consortium at different levels in topics such as data provision and integration, GIS products, GIS interpretation, data archiving and data exchange.

The European Marine Observation and Data Network (EMODnet) is a long-term marine data initiative from the European Commission Directorate-General for Maritime Affairs and Fisheries (DG MARE) underpinning its Marine Knowledge 2020 strategy. EMODnet is a consortium of organisations assembling European marine data, data products and metadata from diverse sources in a uniform way. The main purpose of EMODnet is to unlock fragmented and hidden marine data resources and to make these available to individuals and organisations (public and private), and to facilitate investment in sustainable coastal and offshore activities through improved access to quality-assured, standardised and harmonised marine data which are interoperable and free of restrictions on use.

In the frame of the EVER-EST project the operational CoCoNet catalogue is managed directly by CNR-ISMAR: the Open Search Geo and Time Extensions interface has been enabled to guarantee the interoperability within the Virtual Research Environment and to provide access to the datasets in EMODnet that are considered most relevant for the use cases deployed in EVER-EST.

CoCoNet makes data available freely and without restriction. CoCoNet makes data available in a timely and easy way to users through the WebGIS platform, but CoCoNet data remains dependent on data contributions.

4.4 Visual components for data analysis

The user interface for Data Analysis is provided by specific visual components that provide web tools to discover, access, explore, analyse and download geospatial data directly in the Virtual Globe, the natural context for Earth Scientists to perform their daily job.

The Data Analysis visual component is based on the Multi-sensor Evolution Analysis (MEA) system, an Earth Observation and geospatial data analysis tool empowered with OGC standard interfaces. The original Graphic User Interface (GUI) has been conceived to exploit satellite data products via OGC standard interfaces (e.g. access via Web Coverage Service (WCS), processing via Web Coverage Processing Service (WCPS), etc.). In the frame of the EVER-EST project a critical design review of the web application has been performed to drive the implementation of the service functionalities at the proper granularity to support the implementation and customization of the EVER-EST VRC portals.

A short description of the Data Analysis visual component is provided in the following sections; details can be found in [EVER-EST DEL WP5-D5.7].

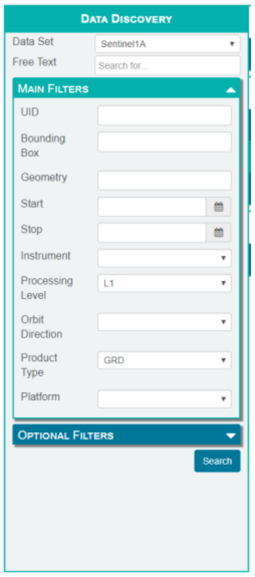
4.4.1.1 Data discovery and access

The Discovery and Access visual component of the Data Analysis Service provides the client interface to discover EO OpenSearch-enabled metadata. It is composed by the *Data Discovery* interface (see Figure 19) and the *Data Discovery Results* interface (see Figure 20) where the in the list of returned entries are presented.

The *Data Discovery* interface is composed of three sections:

1. Common section: allows the user to select the data set to inquiry and to insert some free text to start the search;
2. Required Filters section: contains the list of filters considered most important by the VRCs;

3. Optional Filters section (optional): contains the list of filters considered less important by the VRCs.


Data Discovery

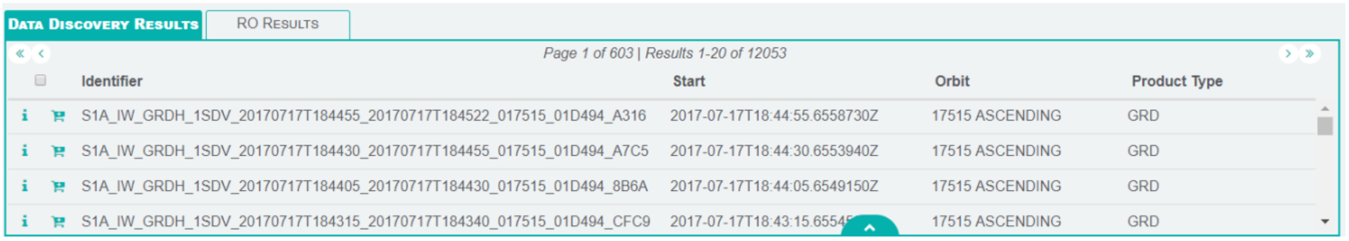
Data Set	Sentinel1A
Free Text	Search for...

Main Filters

UID	
Bounding Box	
Geometry	
Start	
Stop	
Instrument	
Processing Level	L1
Orbit Direction	
Product Type	GRD
Platform	

Figure 19 Data Discovery visual component

The *Data Discovery Results* interface shows the results in a list and allows to move among pages, see products details and add/remove products to/from the discovery basket.



Page 1 of 603 | Results 1-20 of 12053

Identifier	Start	Orbit	Product Type
S1A_IW_GRDH_1SDV_20170717T184455_20170717T184522_017515_01D494_A316	2017-07-17T18:44:55.6558730Z	17515 ASCENDING	GRD
S1A_IW_GRDH_1SDV_20170717T184430_20170717T184455_017515_01D494_A7C5	2017-07-17T18:44:30.6553940Z	17515 ASCENDING	GRD
S1A_IW_GRDH_1SDV_20170717T184405_20170717T184430_017515_01D494_8B6A	2017-07-17T18:44:05.6549150Z	17515 ASCENDING	GRD
S1A_IW_GRDH_1SDV_20170717T184315_20170717T184340_017515_01D494_CFC9	2017-07-17T18:43:15.6554	17515 ASCENDING	GRD

Figure 20 Data Discovery: results page

The Visual Component is tailored on VRC needs:

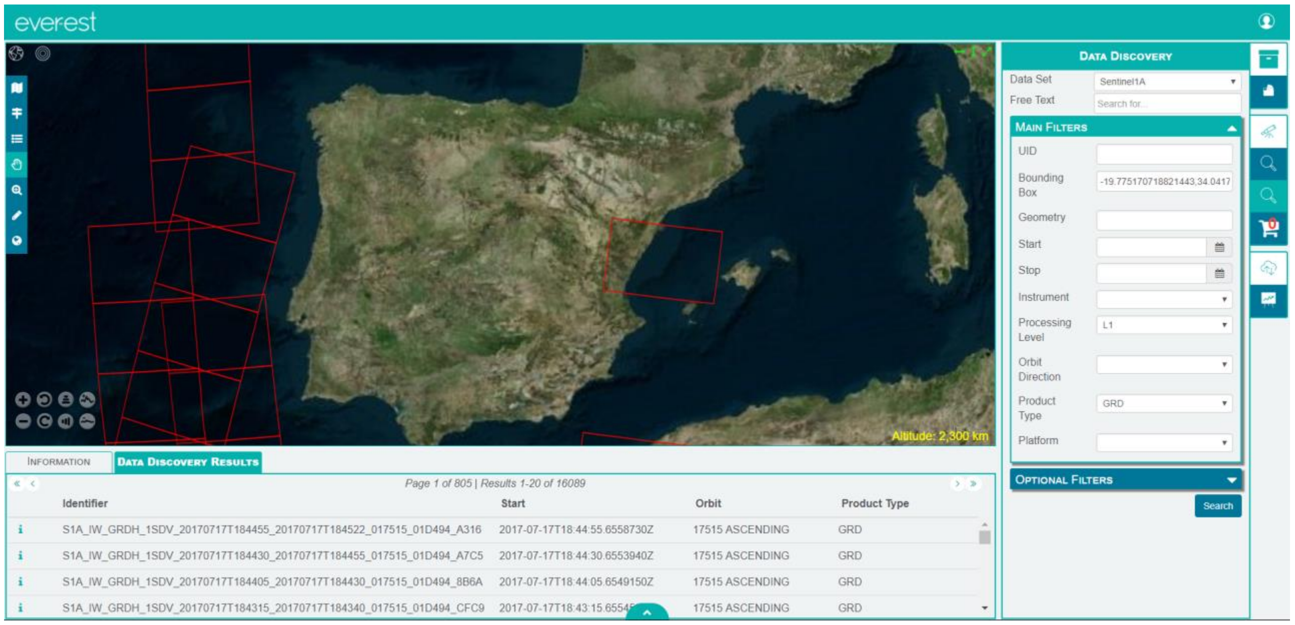
- the filters are automatically gathered from the VRCs OpenSearch repositories and grouped into the previous sections based on VRC requirements
- the list columns can be customized according to the VRC requirements to present only the parameters of interest (e.g. identifier, start date, …)

4.4.2 Visualisation on the Virtual Globe Visual Component

Once the EO data/metadata is selected, the Visualisation component of the Data Analysis Service retrieves the EO data and shows it on the Virtual Globe.

The Virtual Globe VC is based on Web World Wind (WWW), a free, open-source 3D virtual globe using web technologies such as JavaScript. It was developed by NASA and ESA and allows interactive visualisation of geographic information on an interactive 3D globe or 2D map. It provides an API that enables JavaScript programs to control every detail of visualisation and interaction, and runs on all major operating systems, desktop and mobile devices, and web browsers.

Figure 21 shows an example of visualisation on the Virtual Globe: the users can interact with the Virtual Globe using standard GIS toolbox (zoom in, zoom out, area selection, change base layer, etc.).



Data Set	Sentinel1A
Free Text	Search for...
UID	
Bounding Box	-19.775170718821443,34.0417
Geometry	
Start	
Stop	
Instrument	
Processing Level	L1
Orbit Direction	
Product Type	GRD
Platform	

Page 1 of 805 | Results 1-20 of 16089

Identifier	Start	Orbit	Product Type
S1A_IW_GRDH_1SDV_20170717T184455_20170717T184522_017515_01D494_A316	2017-07-17T18:44:55.6558730Z	17515 ASCENDING	GRD
S1A_IW_GRDH_1SDV_20170717T184430_20170717T184455_017515_01D494_A7C5	2017-07-17T18:44:30.6553940Z	17515 ASCENDING	GRD
S1A_IW_GRDH_1SDV_20170717T184405_20170717T184430_017515_01D494_8B6A	2017-07-17T18:44:05.6549150Z	17515 ASCENDING	GRD
S1A_IW_GRDH_1SDV_20170717T184315_20170717T184340_017515_01D494_CFC9	2017-07-17T18:43:15.6554	17515 ASCENDING	GRD

Figure 21 Visualisation component in the 3D virtual globe

The Virtual Globe allow inspection of geo-referenced resources (shape, GeoTiff, KML, KMZ, and PNG + WLD, …). It also allows visualisation of data from Web Map Service (WMS), Web Feature Service (WFS) and Web Map Tile Service (WMTS) services.

4.4.2.1 Web Coverage Service

The OGC Web Coverage Service (WCS)[43] supports electronic retrieval of geospatial data as "coverages", namely the digital geospatial information representing space/time-varying phenomena.

Every implementation of a WCS shall adhere to the WCS standard. The WCS core standard defines only basic requirements, while the WCS extensions define expansions to meet additional requirements, such as the response encoding.

WCS provides access to coverage data in forms that are useful for client-side rendering, as input into scientific models, and for other clients. The WCS may be compared to the OGC Web Feature Service (WFS) and the Web Map Service (WMS). As WMS and WFS service instances, a WCS allows clients to choose portions of a server's information holdings based on spatial constraints and other query criteria.

Unlike WMS, which returns spatial data to be portrayed as static maps (rendered as pictures by the server), the WCS provides available data together with their detailed descriptions; defines a rich syntax for requests against these data; and returns data with its original semantics (instead of pictures) which may be interpreted, extrapolated, etc., and not just portrayed. Unlike WFS, which returns discrete geospatial features, the WCS returns coverages representing space/time-varying phenomena that relate a spatio-temporal domain to a (possibly multidimensional) range of properties. As such, WCS focuses on coverages as a specialised class of features and, correspondingly, defines streamlined functionality. WCS 2.0 uses the coverage model of the GML Application Schema for Coverages [OGC 09-146r1], which has been developed

[43] See http://www.opengeospatial.org/standards/wcs

with the goal that coverages handled by a WCS, can be more easily interchanged with other OGC services. WCS 2.0 supports all coverage types supported by said Application Schema; it is not constrained to quadrilateral grid coverages like previous WCS versions.

The International Standard defines three operations:

- GetCapabilities (discovery operation);
- DescribeCoverage (discovery operation);
- GetCoverage (query operation).

The OGC WCS Range Subsetting Extension Standard[44] adds subsetting capabilities to the WCS operations, while the several encoding capabilities are offered by specific profiles (e.g. OGC WCS GeoTIFF Coverage Encoding Profile[45], the OGC WCS JPEG2000 Coverage Encoding Extension[46] and the OGC WCS GML Coverage Application Schema[47]).

4.4.2.2 Web Map Service

A Web Map Service (WMS)[48] produces maps of spatially referenced data dynamically from geographic information. The International Standard defines a “map” to be a portrayal of geographic information as a digital image file suitable for display on a computer screen. A map is not the data itself. WMS-made maps are generally rendered in a pictorial format such as PNG, GIF or JPEG, or occasionally as vector-based graphical elements in Scalable Vector Graphics (SVG) or Web Computer Graphics Metafile (WebCGM) format.

The International Standard defines three operations: one returns service-level metadata; another returns a map whose geographic and dimensional parameters are well defined; and an optional third operation returns information about particular features shown on a map. Web Map Service operations can be invoked using a standard web browser by submitting requests in the form of Uniform Resource Locators (URLs). The content of such URLs depends on which operation is requested. In particular, when requesting a map the URL indicates what information is to be shown on the map, what portion of the Earth is to be mapped, the desired coordinate reference system, and the output image width and height. When two or more maps are produced with the same geographic parameters and output size, the results can be accurately overlaid to produce a composite map. The use of image formats that support transparent backgrounds (e.g. GIF or PNG) allows underlying maps to be visible. Furthermore, individual maps can be requested from different servers. The Web Map Service thus enables the creation of a network of distributed map servers from which clients can build customised maps.

The International Standard applies to a Web Map Service instance that publishes its ability to produce maps rather than its ability to access specific data holdings. A basic WMS classifies its geographic information holdings into “Layers” and offers a finite number of predefined “Styles” in which to display those layers. The International Standard defines eleven operations:

- GetCapabilities (discovery operation);
- GetMap (query operation);
- GetFeatureInfo (query operation).

[44] See https://portal.opengeospatial.org/files/12-040

[45] See https://portal.opengeospatial.org/files/?artifact_id=54813

[46] See http://docs.opengeospatial.org/is/12-108/12-108.html

[47] See http://docs.opengeospatial.org/is/14-100r2/14-100r2.html

[48] See http://www.opengeospatial.org/standards/wms

For example, the following URL requests an image from the US National Oceanographic and Atmospheric Administration:

> http://a-map-co.com/mapserver.cgi?VERSION=1.3.0&REQUEST=GetMap&CRS=CRS:84&BBOX=-97.105,24.913,-78.794,36.358&WIDTH=560&HEIGHT=350&LAYERS=AVHRR-09-27&STYLES=&FORMAT=image/png&EXCEPTIONS=INIMAGE

4.4.2.3 Web Feature Service

The Web Feature Service (WFS)[49] represents a change in the way geographic information is created, modified and exchanged on the Internet. Rather than sharing geographic information at the file level using File Transfer Protocol (FTP), for example, the WFS offers direct fine-grained access to geographic information at the feature and feature property level.

This International Standard specifies discovery operations, query operations, locking operations, transaction operations and operations to manage stored, parameterized query expressions.

Discovery operations allow the service to be interrogated to determine its capabilities and to retrieve the application schema that defines the feature types that the service offers.

Query operations allow features or values of feature properties to be retrieved from the underlying data store based upon constraints, defined by the client, on feature properties.

Locking operations allow exclusive access to features for the purpose of modifying or deleting features.

Transaction operations allow features to be created, changed, replaced and deleted from the underlying data store.

Stored query operations allow clients to create, drop, list and described parameterised query expressions that are stored by the server and can be repeatedly invoked using different parameter values.

The International Standard defines eleven operations:

- GetCapabilities (discovery operation);
- DescribeFeatureType (discovery operation);
- GetPropertyValue (query operation);
- GetFeature (query operation);
- GetFeatureWithLock (query & locking operation);
- LockFeature (locking operation);
- Transaction (transaction operation);
- CreateStoredQuery (stored query operation);
- DropStoredQuery (stored query operation);
- ListStoredQueries (stored query operation);
- DescribeStoredQueries (stored query operation).

In the taxonomy of services defined in ISO 19119, the WFS is primarily a feature access service but also includes elements of a feature type service, a coordinate conversion/transformation service and geographic format conversion service.

For example, the following URL invokes the mandatory GetFeatureById stored query to retrieve and present a single feature from the server's data store:

[49] See http://www.opengeospatial.org/standards/wfs

http://www.someserver.example.com/wfs.cgi?SERVICE=WFS&VERSION=2.0.2&REQUEST=GetFeature&STOREDQUERY_ID=http://www.opengis.net/def/query/OGC-WFS/0/GetFeatureById&ID=INWATERA_1M.1013

4.4.2.4 Web Coverage Processing Service

The OGC Web Coverage Processing Service (WCPS)[50] defines a language for retrieval and processing of multi-dimensional geospatial coverages representing sensor, image, or statistics data. Services implementing this language provide access to original or derived sets of geospatial coverage information, in forms that are useful for client-side rendering, input into scientific models, and other client applications.

WCPS relies on the coverage model as defined in OGC Abstract Specification and the OGC Web Coverage Service (WCS) Standard where coverages are defined as “digital geospatial information representing space-varying phenomena”, currently constrained to equally spaced grids.

The WCPS language is independent from any particular request and response encoding, as no concrete request/response protocol is specified by WCPS. For setting up a WCPS instance, therefore, a separate, additional specification establishing the concrete protocol is required. This allows embedding of WCPS into different target service frameworks. One such target framework is OGC WCS. WCPS forms an extension of the Web Coverage Service (WCS).

[50] See http://www.opengeospatial.org/standards/wcps

5 E-Collaboration Services and GUI design

The E-Collaboration Services component facilitates the interaction, communication and collaboration between VRC users. For example, two or more VRC members need communication tools to keep in touch each other to exchange information to agree on the development of a specific algorithm, or the VRC communities have to be informed of the recent changes applied to a Research Object of their interest, etc.

In general, the E-Collaboration Services provide Collaboration Management, Group Management, Peer Review, synchronous (e.g. 1-to-1 and 1-to-N instant messaging, Audio-Visual Conferencing, …) and asynchronous (e.g. forum, blog, wiki pages, Send and receive e-mail, etc.) services.

The Content Management System (CMS) technology allows publishing, editing, modifying, organizing, deleting, and maintaining content from a central interface. Moreover the availability of existing add-ons and plugins providing collaboration functionalities (calendar, mailing, forum, etc.) supports the integration of native E-Collaboration in the Virtual Research Environment, and just in case the CMS does not provide the required functionality, a new plugin/add-on can be developed and easily added.

5.1 The Graphical User Interfaces: overall approach

Next figure provides the approach adopted for the design, implementation and integration of both the VRE Graphic User Interface design (GUI) and the specific VRC GUI's.

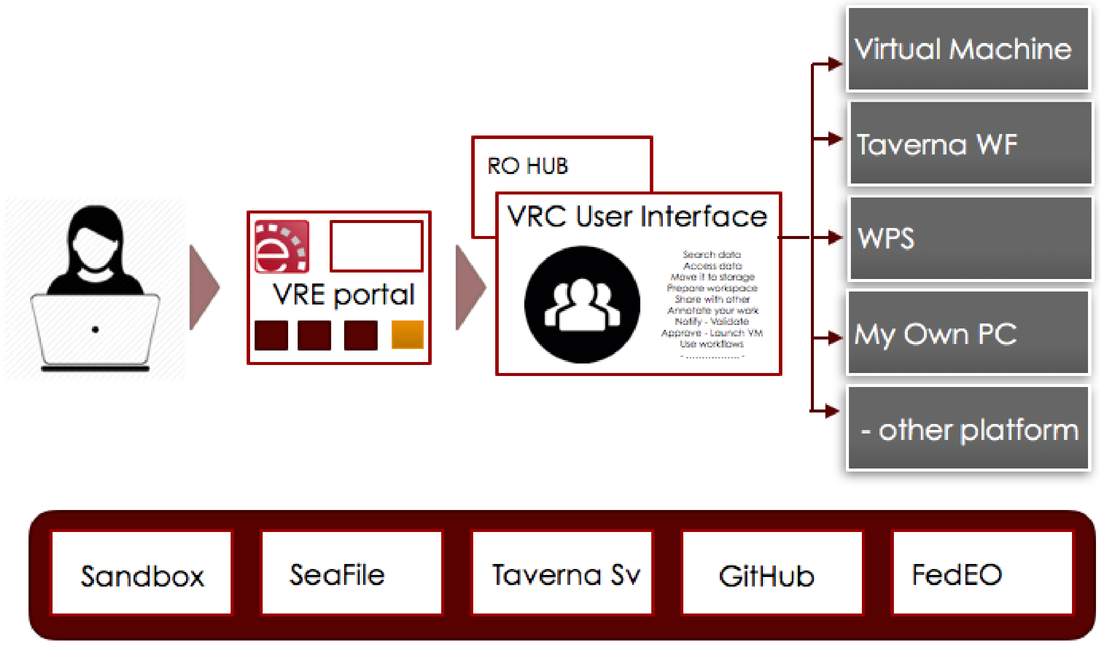


Figure 22 VRE and VRC Graphical User interface

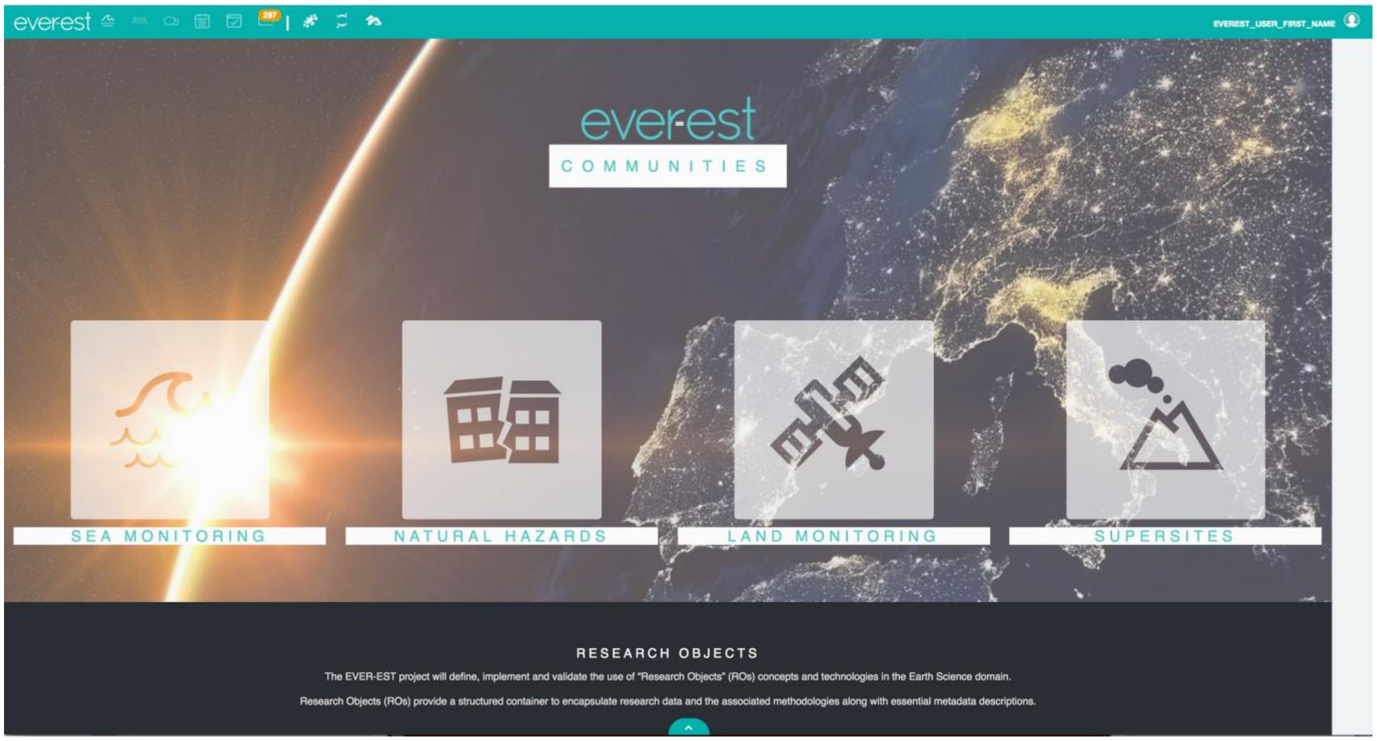
The VRE portal provides access to the EVER-EST infrastructure to both registered and not registered users (i.e. guest users). Each user can access the VRE portal and the VRC GUI can be visualised and/or accessed. The VRC GUI has to be considered the point where (registered) users can perform all those operations that have an impact on the community or are the starting of a new Research (i.e. Research Object creation).

These include:

- Search, access and move data to a Storage Area
- Manage/configure the working space
- Share work with others via RO’s
- Notify/Validate/Message other member of the community
- Launch processing services, get/visualise results

The VRC GUI is the place from where users can launch data exploitation systems such as Virtual Machines, Taverna-based workflows, WPS, Other exploitation platform, even their own pc. In this sense there is a precise distinction between the research preparation, sharing and preserving (via VRC GUI) and the execution of processes via exploitation environment that can be launched and monitored form the VRC GUI. The following sections will provide an overview of how the component is integrated with the other elements of the infrastructure and via which APIs’.

The following pictures shows the current web page of the portal.



RESEARCH OBJECTS

The EVER-EST project will define, implement and validate the use of “Research Objects” (ROs) concepts and technologies in the Earth Science domain.

Research Objects (ROs) provide a structured container to encapsulate research data and the associated methodologies along with essential metadata descriptions.

Figure 23 VRE Portal User Interface

5.1.1 VRE Users

Guest users are allowed to browse the VRE portal and to access the VRC GUI’s, although they are limited to discovery-mode only operations (e.g. RO creation not permitted).

- On the VRE portal they have access to the entire informative section, which is contained on the web-page. All EVER-EST component are displayed as an infographic following the logic of Figure 8, allowing clicking on each component to discover its role and functionalities.

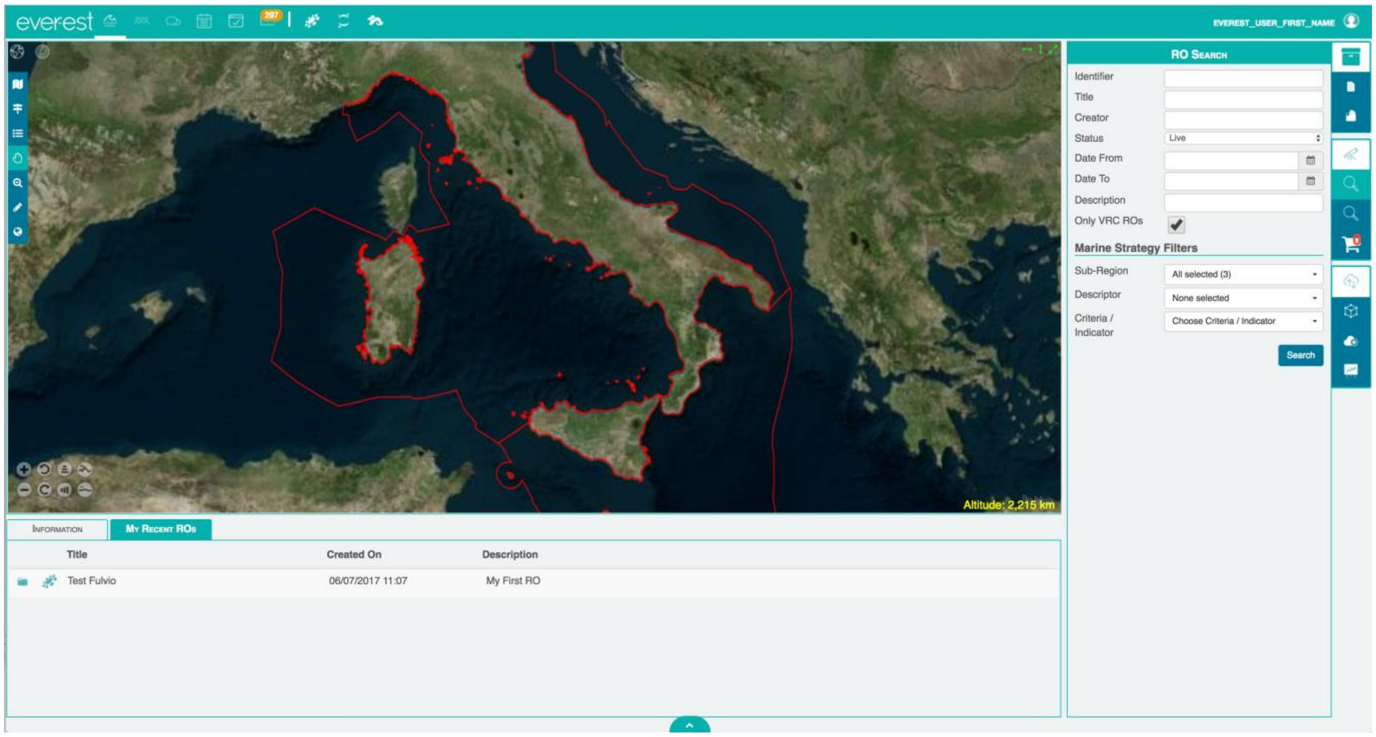
- On the VRC GUI's guests are allowed search for data using the Virtual Globe component. They can search and visualise public RO's (i.e. those RO's that EVER-EST users allow to be seen by everyone). They can visualise (if any) the workflows connected to a specific RO; all the descriptive information related to the RO; the public WPS used to create a specific product; they can browse via the e-learning module what are algorithms used during a specific processing, etc.

This approach to Guest browsing is part of the approach to dissemination and community building. The goal is to enlarge the user community by having the chance to see the interface in action by public and golden RO exemplars.

Registered users have access to all VRE functionalities having access on their specific VRC GUI which has been customised, gathering requirements from each VRC.

5.2 VRC GUI

Each Virtual Research Community GUI has been customised following the advices and needs emerged from each community participating to the project. Nevertheless, the work carried out has been to identify the common areas and functions, to discuss them again with the VRC to find a coherence. During the workshops held in April 2017 the choices for the GUI's where also explained to external research communities to explore their feedbacks. The result is the interface described in this section.



RO Search

Identifier	
Title	
Creator	
Status	Live
Date From	
Date To	
Description	
Only VRC ROs	✓

Marine Strategy Filters

Sub-Region	All selected (3)
Descriptor	None selected
Criteria / Indicator	Choose Criteria / Indicator

Search

Title	Created On	Description
Test Fulvio	06/07/2017 11:07	My First RO

Figure 24 GUI for the Sea Monitoring Virtual Research Community

This includes the following main areas:

- **EVER-EST TOP Bar**: provides access to all the messaging functions (see the E-Collaboration section) and allows the link to external components used by EVER-EST: RO-HUB, SeaFile, E-learning module.

- **Map Section**: based on Virtual Globe, provides all functionalities for data search and 3D map interaction, including zoom, pan, etc.
- **Information panel**: (on the bottom). Displays every type of information related to the objects used in the interface: from data or RO search Results, to the descriptive metadata of a selected object, to the recent RO's opened by the user, etc. It is organised as a multi-tab panel.

The **Right Part** of the GUI provides the EVER-EST toolbar that enables:

- **Research Object Panel**: provides all functions that are needed to manage RO's. These include basic functionalities (i.e. save, delete, refresh), RO lifecycle management services (i.e. fork, create snapshot, publish), Metadata management, RO resource management (i.e. add link, add selected data, remove annotation).
- **Data/RO Search Panel**: provides the proper interfaces to define search criteria for both RO's and EO/ES data. It included a *Basket* section in case the users want to select multiple data to be progressively added to a RO.
- **Services Panel**: divided into three logical sections
  - **Available VMs:** allows connecting to Terradue Cloud controller and launch a specific VM which will be described in the panel
  - **Available WPS:** allows to launch one of the WPS's included in the list of available WPS, applying it to the current RO or to some selected data in order to create a new RO. The characteristics of each WPS are available in the panel. Managed by the Terradue Cloud Controller.
  - **Available Workflows:** as for the WPS section, with the difference that these workflows are managed by a generic Taverna server.

5.3 VRC GUI technologies and architecture

The design and implementation of the user interfaces is built around the concept of ***visual component***, that is an independent web element written using open web technologies (HTML, CSS, and JavaScript/jQuery) that can be added to any web page using a few lines of code. VCs make it possible to centralize a feature, add it on multiple web portals (e.g. VRC portal, RO-Hub, …) keeping the same behaviour and, at the same time, customize it with some page-related options.

5.3.1 VRC GUI related visual components

The EVER-EST visual components constitute the building blocks of the EVER-EST Portal and VRC Portals. The VCs are developed using open web technologies (e.g.: HTML5 standard; Javascript; jQuery; Bootstrap 3 Framework; CSS 3) as independent objects that could be added to a web page by adding just a few lines of code.

The VCs currently developed are:

- Discovery visual component: allows datasets and data discovery via OpenSearch queries on OpenSearch enabled repositories
- Virtual Globe visual component: allows EO data visualisation (described in 4.4.2);
- RO Manager visual component: allows research object creation, visualisation and editing;
- Upload to Seafile visual component: allows file upload and basic management to Seafile within the VRC;
- Workflow Runner: allow workflow execution within the VRC GUI;
- WPS Manager: allows WPS retrieval, execution and input/output management.

A detailed description of the VCs is presented in the following sections.

5.3.1.1 RO Manager visual component

The RO Manager visual component allows the user to search for research objects, view RO details, create a new RO, and edit an existing RO. The RO Manager is composed of a set of several independent VCs:

- RO Bar: creates the right bar;
- RO Manager, RO Info: manages RO visualisation in the right panel;
- RO Search: used to manage RO search, it creates the filters component in the right panel and the list component in the info panel;
- Recent ROs: used to show to the user the list of the last two weeks ROs;
- RO Creator: allows RO creation;
- Refresh RO Content: used to refresh the content of the RO;
- Delete RO: delete a RO;
- RO Snapshot/Publish: used to create a snapshot of the RO or to publish the RO;
- Add Annotation to RO: used add annotations/metadata to a RO;
- RO Folders: used to manage folders inside a RO;
- Add To RO, Add Link To RO: used to add resources to a RO;
- Remove From RO: used to remove resources from a RO.

A few RO functionalities are always visible and accessible by the users: the RO Bar and the discovery bar (depicted in Figure 25) integrate the visual components used to create, show recent ROs and Search ROs.

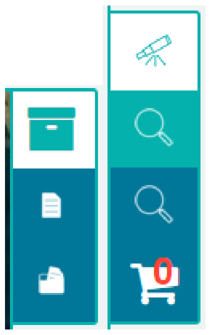
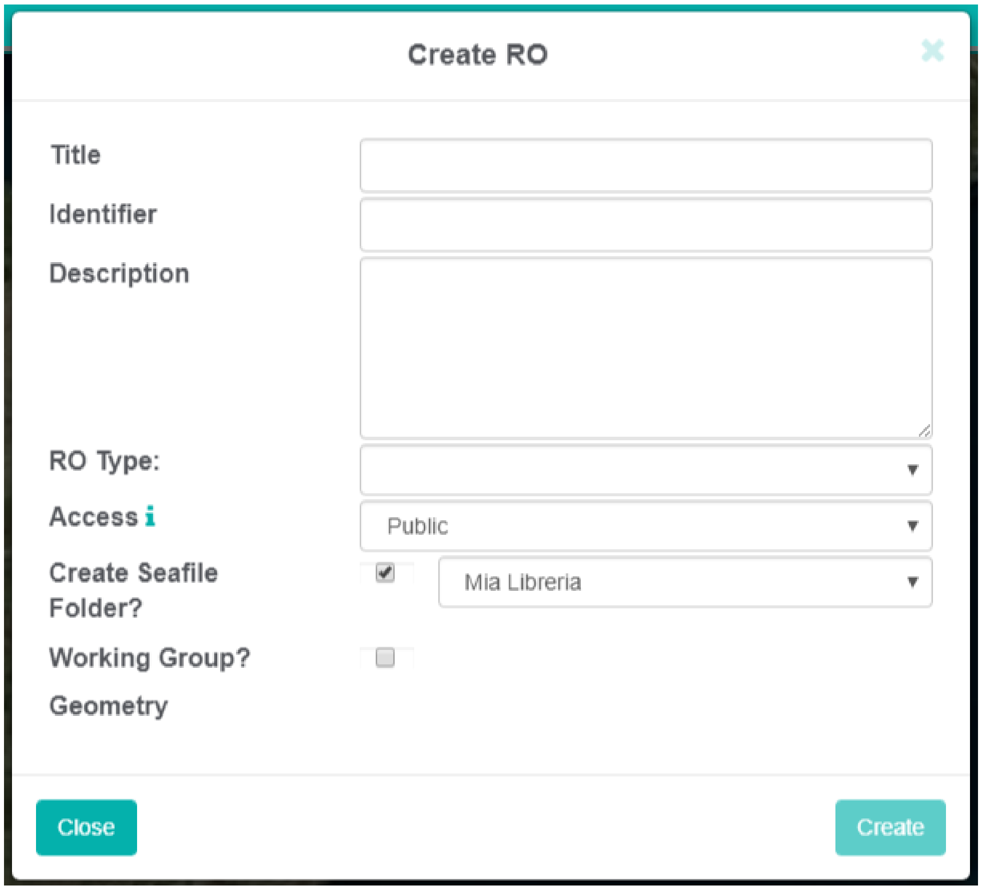


Figure 25 RO Bar (left) and Discovery Bar (right)

When the user wants to create a new RO and the related button is selected in the RO Bar, a wizard is shown (depicted in Figure 26) and the user guided through the creation of the RO.

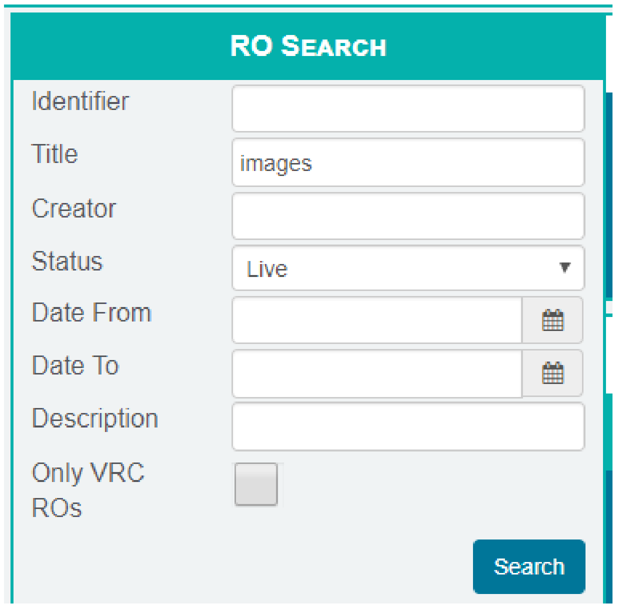


Create RO

Title	
Identifier	
Description	
RO Type:	
Access	Public
Create Seafile Folder?	✓ Mia Libreria
Working Group?	☐
Geometry	

Figure 26 RO creator wizard

The Recent ROs button shows a list of the last ROs while the Search RO button allows the user to search ROs by using several filters as shown in Figure 27.

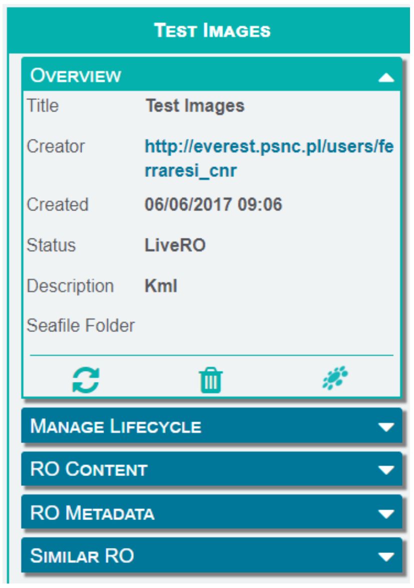


RO SEARCH

Identifier	
Title	images
Creator	
Status	Live
Date From	
Date To	
Description	
Only VRC ROs	☐

Figure 27 RO Search

When a RO is selected because it has been discovered via RO Search or Recent ROs, the RO panel is shown on the right:



Test Images

Overview

Title	Test Images
Creator	http://everest.psnc.pl/users/ferraresi_cnr
Created	06/06/2017 09:06
Status	LiveRO
Description	Kml
Seafile Folder	

Manage Lifecycle

RO Content

RO Metadata

Similar RO

Figure 28 RO Panel

The RO panel integrates the visual components used to manage a RO grouped by thematic sections:

- Overview: shows the key information for the RO, such as title, creator, status and description and allows the user to: refresh the content, delete the RO and to open the RO in ROHUB;
- Manage Lifecycle: allow the user to create a snapshot, publish the RO and check the RO quality;
- RO Content: show the resources contained into the RO and allows the user to create folder inside the RO, add links and files (both from PC and Seafile) and to remove resources;
- RO Metadata: shows the annotations/metadata to the user and allows the creation of new annotations/metadata;
- Similar ROs: show the ROs that are linked to the opened RO using the recommended RO service by ESI.

5.3.1.2 Upload to Seafile visual component

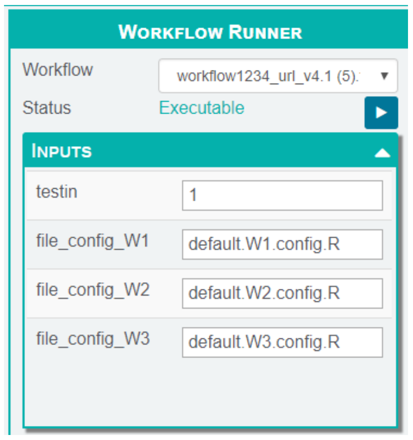
The Seafile visual component provides the EVER-EST users with the basic functionality to upload and manage resources in their private Seafile storage. The visual component is composed of two components:

- Seafile Uploader allows the user to upload data to specific Seafile folders;
- Seafile Content Viewer shows Seafile content to the user and allows file selection.

These components are integrated into other EVER-EST visual components (e.g.: Virtual Globe and Add To RO) and are identified by the Seafile icon.

5.3.1.3 Workflow Runner visual component

The Workflow Runner Visual Component allow the Workflow information retrieval, execution and input/output management. The visual component is integrated with the RO and Seafile visual components to simplify input and output management.



WORKFLOW RUNNER

Workflow: workflow1234_url_v4.1 (5)

Status: Executable

INPUTS

testin: 1

file_config_W1: default.W1.config.R

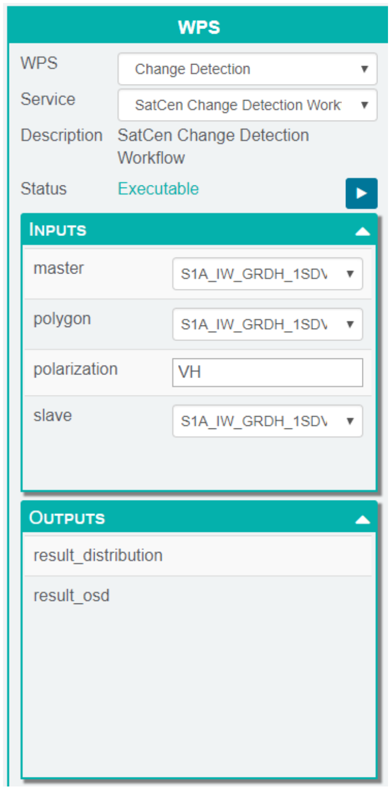
file_config_W2: default.W2.config.R

file_config_W3: default.W3.config.R

Figure 29 Workflow Runner panel

5.3.1.4 WPS Manager visual component

The WPS Manager visual component allows WPS information retrieval, execution and input/output management The visual component is integrated with the Discovery visual component to simplify WPS input management, and with the RO creator to create a RO after the WPS execution.



WPS

WPS: Change Detection

Service: SatCen Change Detection Work

Description: SatCen Change Detection Workflow

Status: Executable

INPUTS

master: S1A_IW_GRDH_1SDV

polygon: S1A_IW_GRDH_1SDV

polarization: VH

slave: S1A_IW_GRDH_1SDV

OUTPUTS

result_distribution

result_osd

Figure 30 WPS Manager panel

5.4 Components

The E-Collaboration services component facilitates the interaction, communication and collaboration between the members of the Virtual Research Community. Django, the Content Management System (CMS) on which the EVER-EST portal and the VRC portals are based, allows the integration of most of the e-collaboration capabilities through third party plugins. Other e-collaboration services have been integrated as JavaScript extensions.

When required, the plugins have been customised to better suits EVER-EST VRC needs.

Based on Users Stories, the following E-Collaboration functionalities have been identified and integrated in to EVER-EST Virtual Research Environment:

- Instant Messaging (1-to1 and 1-to-N);
- Forum / work sharing;
- Tasks / Scheduling;
- My validation request;
- Internal Messaging and Notification.

The E-collaboration services are accessible via the EVER-EST TOP Bar (see Figure 31).

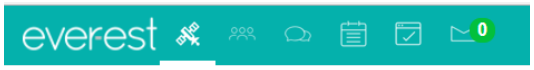


Figure 31 E-Collaboration Toolbar

Groups management functionality is also available to support the sharing of resources among VRC members. Groups can be defined under the user *Settings* page or dynamically defined during the RO creation.

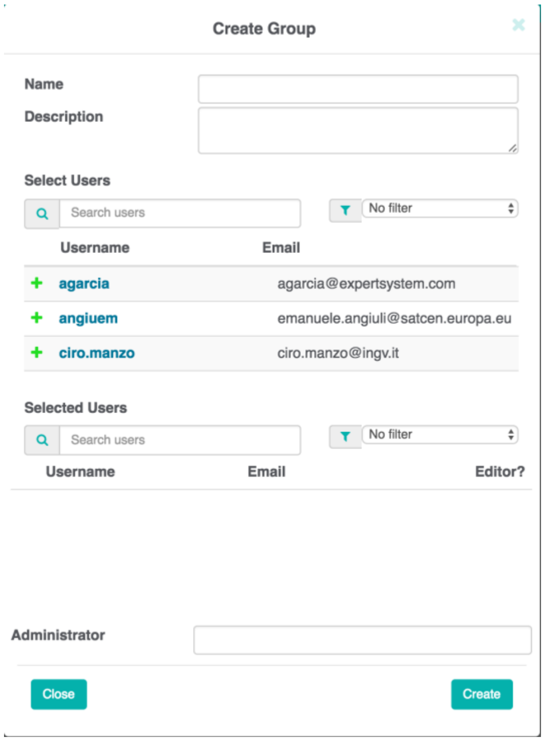
5.4.1 Group management

This component interfaces with the VRE Common Services - Identity management to support provisioning, removing and management of user groups with the same interests, goals, ... Each VRC is an example of a high level group that should be created. It also should be possible to create inner-groups.

Each VRC member has the possibility to create and manage a new group by clicking the "Peer Group Management" icon in the E-Collaboration toolbar (see Figure 32).

The Group owner defines:

- Group Name, the short name of the group used to identify the group itself;
- Group Description, an optional description to provide general info on the group;
- Add to (green arrow) and remove from (red arrow) the Group the VRC members already registered in the VRE portal / VRC environments;
- Share group ownership with other members.



Create Group

Name	
Description	

Select Users

Search users · No filter

Username	Email
agarcia	agarcia@expertsystem.com
angiuem	emanuele.angiuli@satcen.europa.eu
ciro.manzo	ciro.manzo@ingv.it

Selected Users

Search users · No filter

Username	Email	Editor?

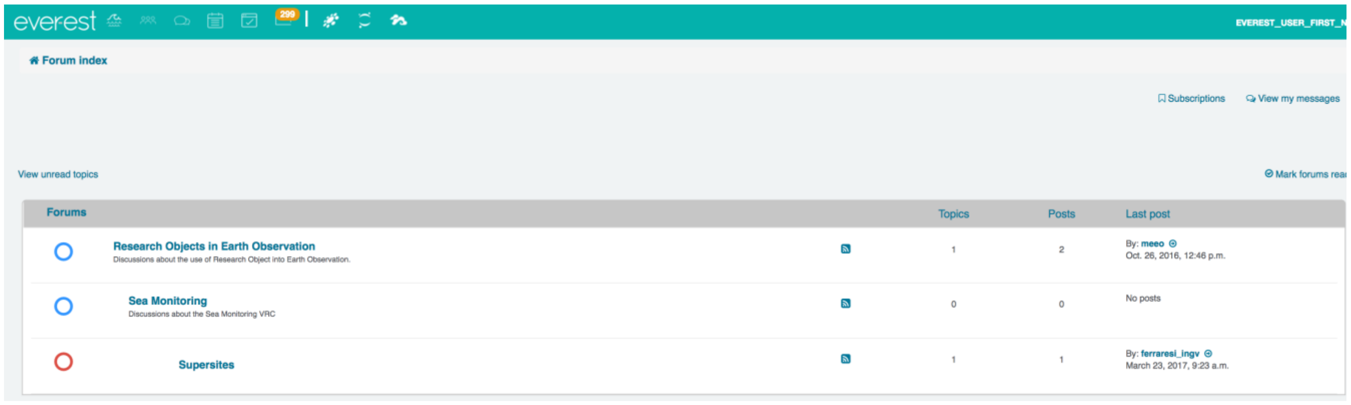
Administrator

Close · Create

Figure 32 Group Management User Interface

5.4.2 Forum / work sharing

The Work Sharing component provides the tools to organize workflows among a group of researchers. In this light, the central components are the Forum module, depicted in the next picture, which allows general discussion about any topics of interest:



everest · 299 · EVEREST_USER_FIRST_N

Forum index

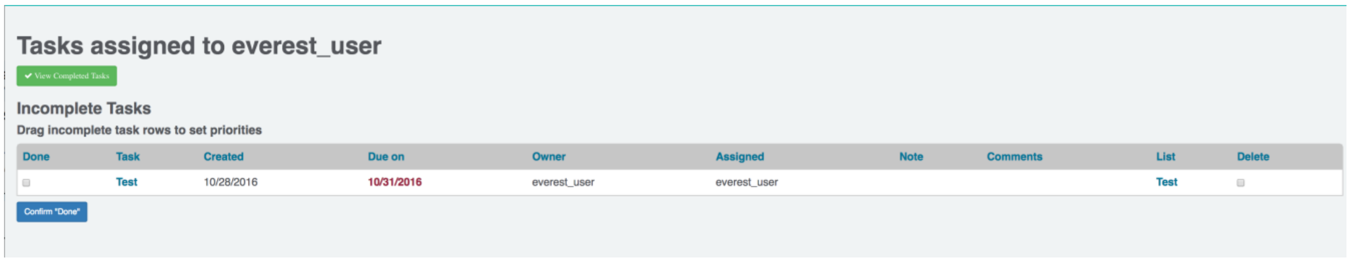
Subscriptions · View my messages

View unread topics · Mark forums rea

Forums	Topics	Posts	Last post
Research Objects in Earth Observation Discussions about the use of Research Object into Earth Observation.	1	2	By: meeo Oct. 26, 2016, 12:46 p.m.
Sea Monitoring Discussions about the Sea Monitoring VRC	0	0	No posts
Supersites	1	1	By: ferraresi_ingv March 23, 2017, 9:23 a.m.

Figure 33 Forum Module

Central to team organisation is the validation module, which can be linked to any type of running workflow. All VRC use case have been examined in detail to define all kind of validation activity inside a group of researchers. The Validation module allows every user to be informed about pending activities related to validation of other users' outputs. The module is depicted in the following picture, under the umbrella and naming convention of Assigned Tasks, which includes validation and other minor activities.


Tasks assigned to everest_user

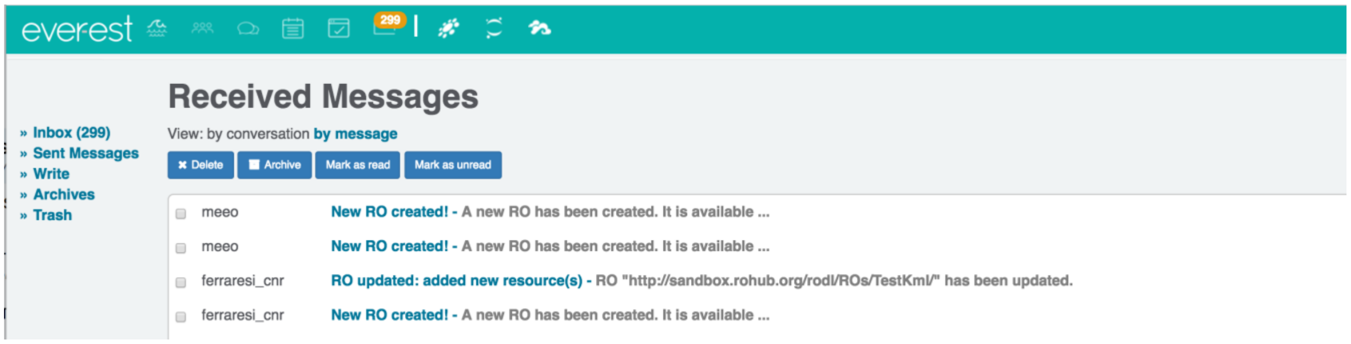
Incomplete Tasks

Drag incomplete task rows to set priorities

Done	Task	Created	Due on	Owner	Assigned	Note	Comments	List	Delete
☐	Test	10/28/2016	10/31/2016	everest_user	everest_user			Test	☐

5.4.3 Internal Messaging and Notification

Each user can belong to generic working groups created by researchers and can be inserted into an RO working group directly from the RO creator. In this case all major events happening within the group or to the RO (i.e. snapshot, fork, etc.) will be notified by internal email. The following picture shows the mailing component:



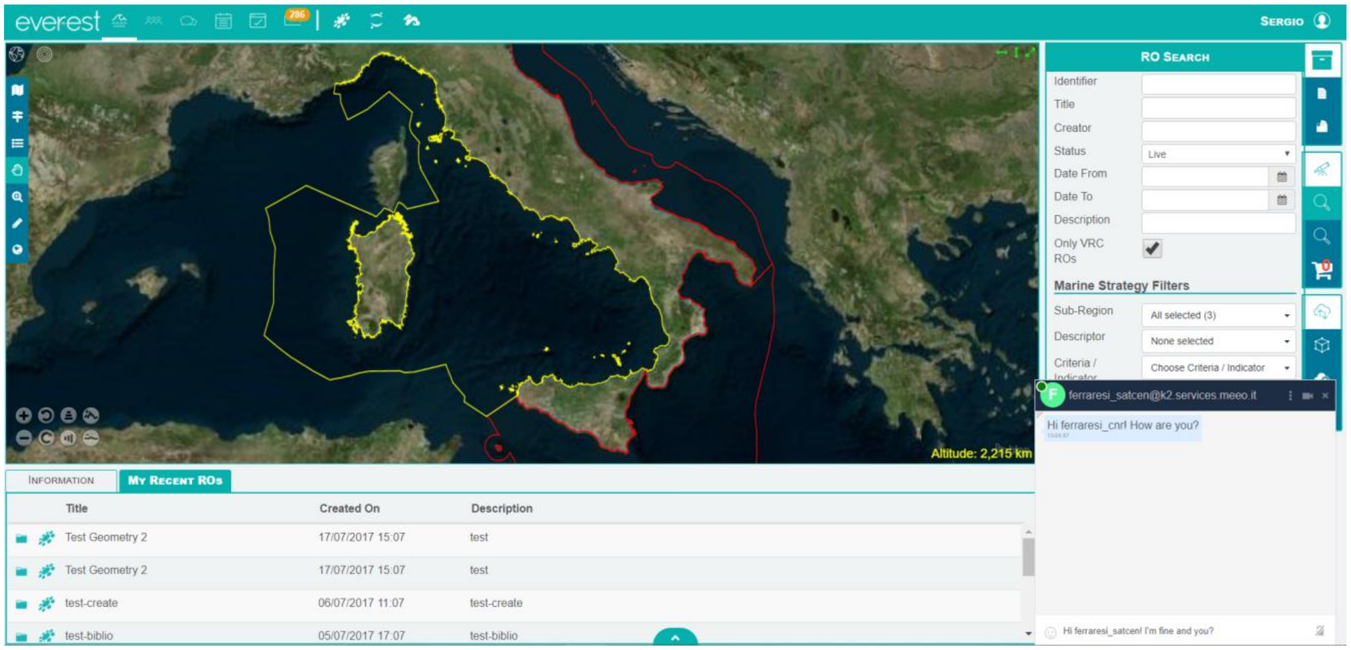
Received Messages

View: by conversation by message

☐	meeo	New RO created! - A new RO has been created. It is available ...
☐	meeo	New RO created! - A new RO has been created. It is available ...
☐	ferraresi_cnr	RO updated: added new resource(s) - RO "http://sandbox.rohub.org/rodl/ROs/TestKml/" has been updated.
☐	ferraresi_cnr	New RO created! - A new RO has been created. It is available ...

Figure 34 Mailing component

In addition, the Instant Messaging component provides the tools for 1 to 1 and 1 to N communication. The VRC member can easily start a chat with one or more VRC members interacting with the main E-Collaboration toolbar. If the user has defined a Group, by default the chat addresses all the group members. The 1 to 1 communication is also supported. The Audio/Video Conferencing component provides the tools for Virtual Meeting management.



Identifier		
Title		
Creator		
Status	Live	
Date From		
Date To		
Description		
Only VRC ROs	✓	

Marine Strategy Filters

Sub-Region	All selected (3)
Descriptor	None selected
Criteria / Indicator	Choose Criteria / Indicator

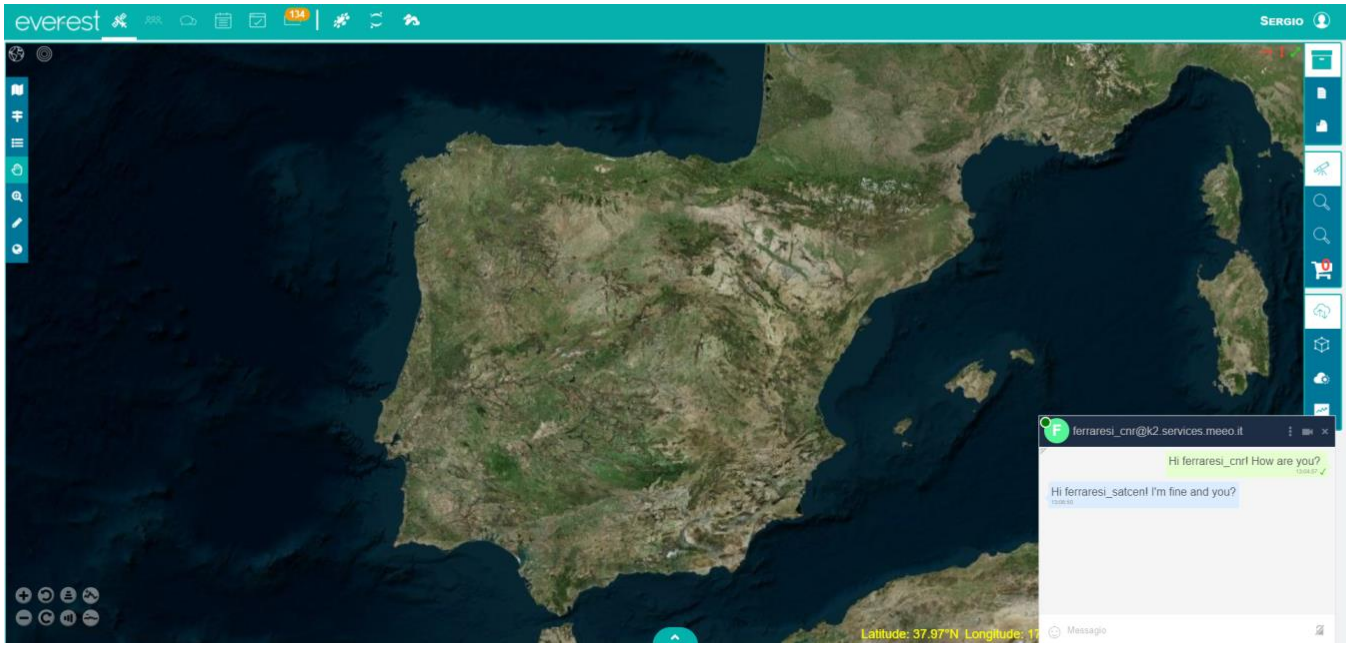
Title	Created On	Description
Test Geometry 2	17/07/2017 15:07	test
Test Geometry 2	17/07/2017 15:07	test
test-create	06/07/2017 11:07	test-create
test-biblio	05/07/2017 17:07	test-biblio

ferraresi_satcen@k2.services.meeo.it

Hi ferraresi_cnr! How are you?

Hi ferraresi_satcen! I'm fine and you?

Figure 35 Instant messaging example (send message)



ferraresi_cnr@k2.services.meeo.it

Hi ferraresi_cnr! How are you?

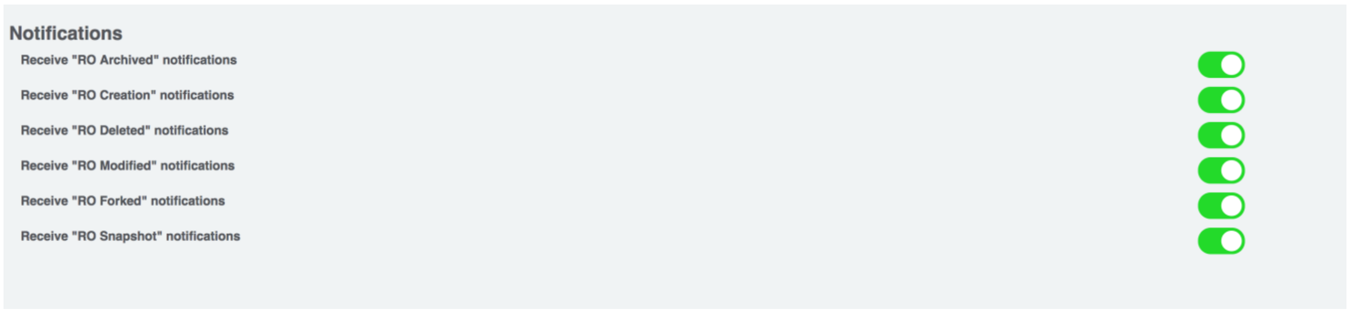
Hi ferraresi_satcen! I'm fine and you?

Messagio

Figure 36 Instant messaging example (Other group receives message and replies)

5.4.4 User profiling

Each user can set up and manage its RO notifications from the User control panel as shown in the following picture.



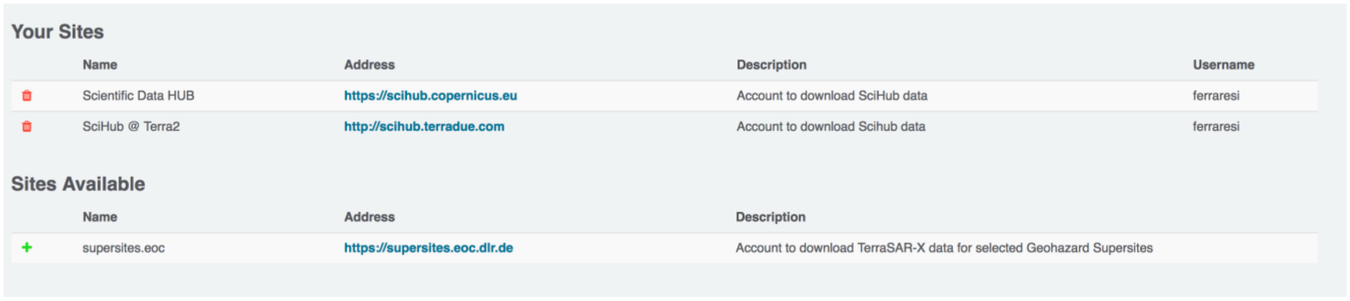
Notifications

Receive "RO Archived" notifications	🟩
Receive "RO Creation" notifications	🟩
Receive "RO Deleted" notifications	🟩
Receive "RO Modified" notifications	🟩
Receive "RO Forked" notifications	🟩
Receive "RO Snapshot" notifications	🟩

Figure 37 RO notification settings

5.4.5 **User workspace monitoring**

Users can use the workspace button from the top bar, as well as the user profile settings, to visualise the workspace (i.e. available services, VM's, etc.) that are available.



Your Sites

	Name	Address	Description	Username
🗑	Scientific Data HUB	https://scihub.copernicus.eu	Account to download SciHub data	ferraresi
🗑	SciHub @ Terra2	http://scihub.terradue.com	Account to download Scihub data	ferraresi

Sites Available

	Name	Address	Description
+	supersites.eoc	https://supersites.eoc.dlr.de	Account to download TerraSAR-X data for selected Geohazard Supersites

6 E-Research Services

6.1 Cloud platform

The data processing infrastructure is based on a Cloud Computing Platform-as-a-Service (PaaS) approach, in which applications can be integrated using Virtual Machines with their own specifications (disk size, processor speed, operating system and so on) that run on private (physical deployment over local hardware) or commercial Cloud infrastructures. The Virtual Machines are appliances running on an infrastructure managed by a Cloud Controller, and accessible as an independent domain once instantiated. It includes a Development Environment where applications or services are first integrated as Cloud Appliances and then are moved to Production Centers when fully deployed for their exploitation.

The Development Environment provides developers with tools and services that support the implementation and deployment of applications in a sandbox to fully exploit the power of distributed computing on a Cloud Infrastructure-as-a-Service. When fully deployed in production it allows users to instantiate (on-demand) a processing service appliance provision the appliance on a pre-configured ICT Provider and invoke the processing via the OGC Web Processing Service interface.

6.1.1 Operational scenarios

This section describes the Platform operational scenarios as:

- Administration of the Cloud infrastructure;
- Administration of the Cloud sandboxes;
- Development of scalable processing applications;
- Data products facility management.

6.1.1.1 Cloud infrastructure administrator

This scenario supports an **Administrator** to setup and configure the Cloud Controller capabilities to manage a Cloud Infrastructure. The process starts from a set of selected basic Operating Systems and software appliances managed as VM Images, to their deployment in a private network offering a concept of "laboratory" where appliances are deployed, and through their lifecycle management (create, start, stop, backup, clone).

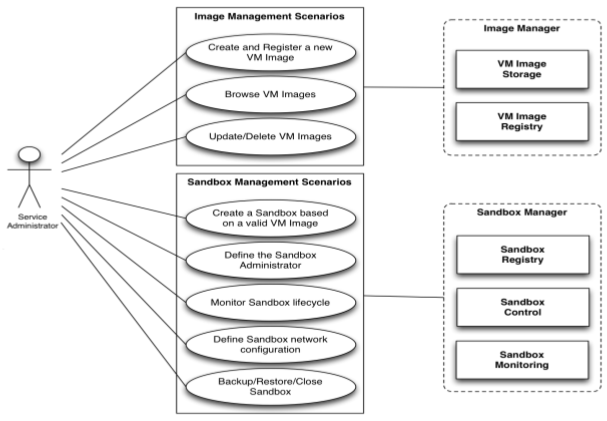


Image Management Scenarios
Create and Register a new VM Image
Browse VM Images
Update/Delete VM Images
Service Administrator
Sandbox Management Scenarios
Create a Sandbox based on a valid VM Image
Define the Sandbox Administrator
Monitor Sandbox lifecycle
Define Sandbox network configuration
Backup/Restore/Close Sandbox
Image Manager
VM Image Storage
VM Image Registry
Sandbox Manager
Sandbox Registry
Sandbox Control
Sandbox Monitoring

Figure 38 Cloud infrastructure administrator use case

6.1.1.2 Cloud sandbox administration

This scenario supports an **Administrator** to setup the PaaS' developer Cloud sandboxes within a given laboratory, covering the exploitation of the infrastructure's VM Image templates and their contextualisation parameters, their allocation to user groups, and the configuration of data access and processing services required for a given application development.

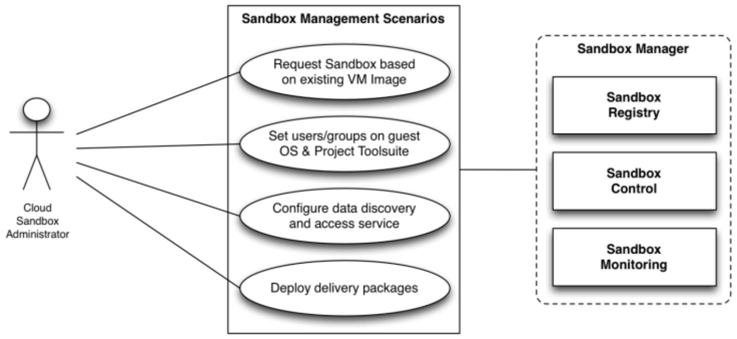

Sandbox Management Scenarios
Request Sandbox based on existing VM Image
Set users/groups on guest OS & Project Toolsuite
Configure data discovery and access service
Deploy delivery packages
Cloud Sandbox Administrator
Sandbox Manager
Sandbox Registry
Sandbox Control
Sandbox Monitoring

Figure 39 Cloud Sandbox administrator use case

6.1.1.3 Application development/integration use case

This scenario supports a **Developer** to browse and access the Sandboxes supporting his application development task, start and stop these Virtual Machines, and exploit their PaaS capabilities for the development of his application. There are several tools to support users in managing the software components contributing to their Application Workflow (systematic versioning and libraries dependencies management in RPM repositories, complete software life-cycle management based on *git* and *maven* tools).

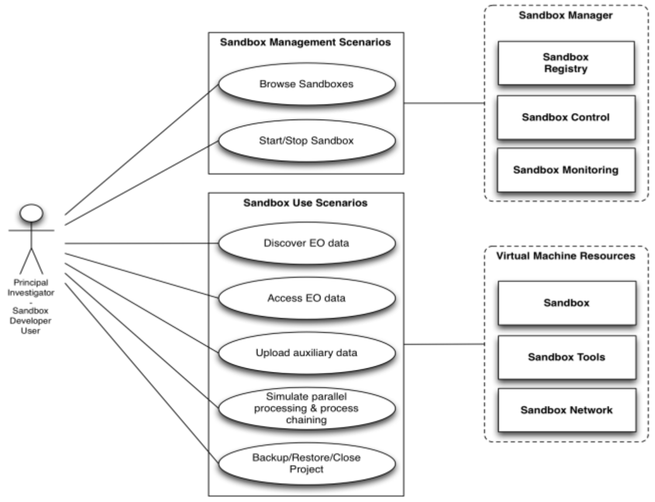


Figure 40 Application Development/ Integration use case

6.1.1.4 Data products facility management use case

This scenario supports a **Developer** in exploring the contents of **Data Provider** by browsing and accessing the Platform catalogues that are supporting the application development task, and exploit a standard interface to request data staging operations from the repositories into selected S3-like buckets.

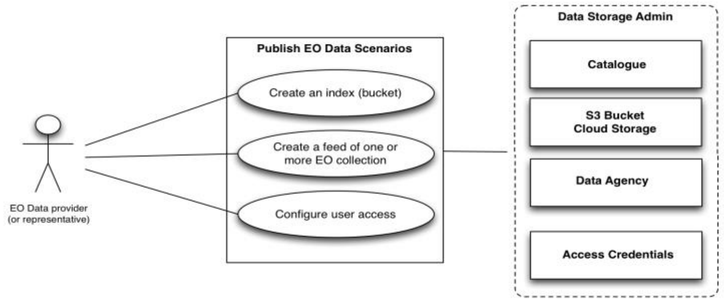


Figure 41 Data products facility management use case

6.1.2 Cloud controller

The Cloud platform is powered by the open-source software OpenNebula, a Cloud computing solution for managing heterogeneous distributed data centre infrastructures available and its toolkit manages a data centre's virtual infrastructure to build private, public and hybrid IaaS.

The Cloud Controller is in charge of the Cloud resources management. It offers several interfaces to provision resources (both private, hybrid and commercial Clouds) and related control such as start/stop Virtual Machines and usage accounting. It is an extension of the OpenNebula Cloud controller, notably featuring Orchestration and Auto-Scaling of Cloud Multi-Tier Applications connecting the federation of infrastructures. This component is in charge of provisioning the resources on the infrastructure either as rented bare metal machines or in the Cloud as Virtual Machines exploiting other public IaaS via their standard APIs (OCCI, JClouds, EC2). The request for resources control can be manual (from an operator) but also automatic in the case of the dynamic clusters (e.g. horizontal scaling up or down of computing resources).

The XML-RPC interface is the primary interface for OpenNebula, and it exposes all the functionality to interface the Cloud Controller and OpenNebula daemon. Through the XML-RPC interface, it is possible to control and manage any OpenNebula resource, including Virtual Machines, networks, images, users, hosts and clusters.

6.1.2.1 PaaS environment

The platform is delivered "as-a-Service" to applications developers, allowing them to:

- Access a dedicated application integration environment in the Cloud, with software tools and libraries available and easily exploitable from that developer environment;
- Search for data collections from distributed EO Data repositories and from Open Data repositories within the Platform, and retrieve results in the application integration environment;
- Integrate a set of data processing chains (data pipeline), leveraging a selected data programming models (e.g. MapReduce and non-MapReduce frameworks such as Distributed-Shell, graph processing, MPI, TEZ, Spark, Storm etc.);
- Deploy a Cloud appliance (e.g. for on-demand EO data processing) prepared for the processing of a range or category of datasets, and running on a selected third party Cloud service provider. Practically the resulting EO data processing appliance can be instantiated on a Cloud cluster for the time of the processing and then terminated (pay-as-you go model), or deployed over a

selected time period, when continuous operations and availability are required (subscription model);

- Ultimately, expose the resulting application through a Web Service endpoint. It is a Web Processing Service (OGC WPS) to allow end-user and B2B client applications to pass processing parameters, trigger a data processing requests and retrieve the information produced.

These properties define the technical platform in terms of a Platform-as-a-Service (PaaS) environment for developing Environmental services that exploit data resources and publish results. As such, it allows domain specialists to concentrate and focus on the development of their products and get them deployed and published to the market sooner.

6.1.2.2 Hybrid cloud API drivers

The Cloud Controller interface delivers functionalities such as multi-tenancy, resource provisioning elasticity, and self-service provisioning on top of a Virtualised Data Centre. Hybrid Cloud capabilities refer to the management of both resources provisioned on a private Cloud (or as the environment dedicated to the development activities of Cloud applications), and their bursting, on demand, to a third party Cloud Provider (or the environment dedicated to the operational exploitation of Cloud applications), which is selected by the Application provider based on the price, resources virtualisation features, virtual resources capacity, and availability over time, to name just a few possible criteria.

The hybrid Cloud model present in OpenNebula enables a centralised management of both local and remote resources for the cloud administrator. A number of Cloud API drivers contributed directly by the OpenNebula community are being leveraged by the platform. The Cloud Controller acts as platform engine empowering this flexible Cloud bursting capacity and it currently supports the following Cloud Provisioning APIs to interoperate with third party Cloud providers (given proper user account subscriptions that are delivering access keys to these 'external' Clouds):

- Public (commercial) Cloud provider Amazon Web Services supported by the EC2 API driver;
- Public (commercial) EGI Federated Cloud supported by the OCCI API driver;
- Private (mostly academic) Cloud Providers supported by the OCCI API driver;
- Private Cloud Providers supported by the vCloud API driver;
- Public (commercial) Cloud Providers supported by the jClouds API, with the *jclouds4one* driver, developed by Terradue and contributed to the OpenNebula open source codebase, that is expanding OpenNebula with the ability to work with currently a variety of up to 30 cloud providers & cloud software stacks, including Amazon Web Services, Microsoft Azure, Google Compute, IBM SoftLayer, CloudStack, OpenStack, GoGrid, Rackspace and Docker.

6.1.3 Development environment

6.1.3.1 Developer cloud sandbox

The Developer Cloud Sandbox PaaS is the environment to integrate scientific applications written in a variety of languages (e.g. Java, C++, IDL, Python, R, scripting languages), then deploy, automate, manage and scale them in a structured engineering approach. The algorithm integration is performed from within a dedicated Virtual Machine, running initially as a simulation environment (sandbox mode) that can readily scale to simulate production (cluster mode). Accessed from a harmonised Shell environment, support tools also facilitate the data access and workflow management tasks. In its simplest case, the development is focused on single machine integration where the *developer* is provided with a Virtual Machine with enhanced capacities to access EO data collections.

When accessing a Developer Cloud Sandbox service, partners are accessing a 'designer' tool that is a meant to design, test and validate a parallel processing workflow (data pipeline) ready to run on a large Cloud processing cluster. This design phase is offered as a cost effective 'cluster simulation' environment: a Sandbox. Typically the baseline is a Virtual Machine running a Developer Cloud Sandbox service that is configured according to the application runtime requirements in terms of CPU, RAM and local storage. These requirements can evolve and the Virtual Machine capacity can be increased on-demand, depending on selected sample datasets that is representative of what each processing node of the cluster receives as part of the platform automated "input files distribution" mechanism depending on the amount and nature of the data to be processed in a given timeframe according to service requirements.

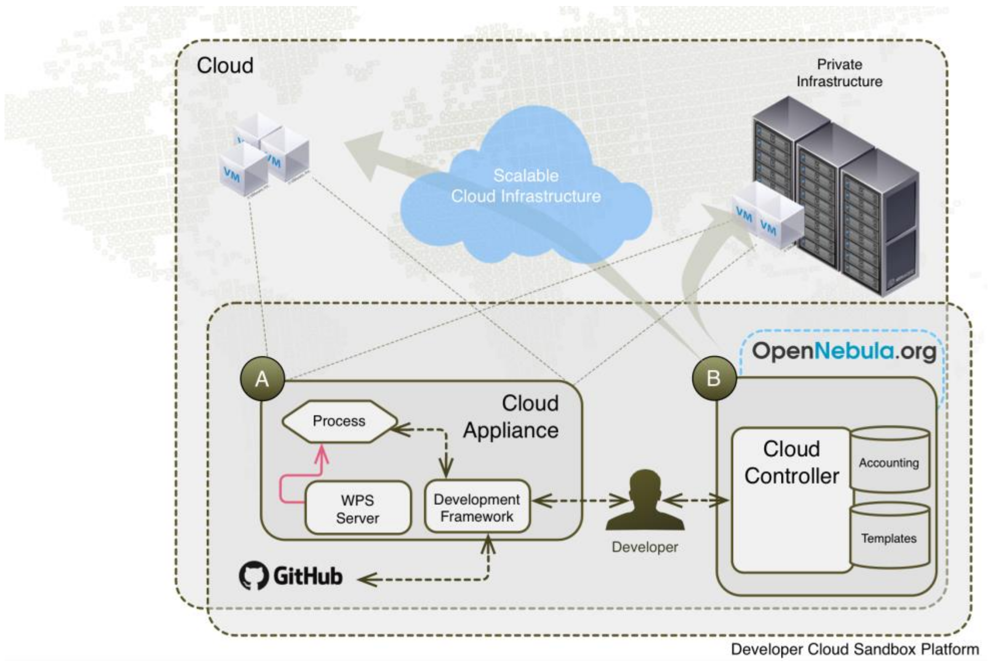


Figure 42 Development and Runtime containers deployment

When working on the Cloud Sandbox, developers and integrators are under a VPN (Virtual Private Network) that can define 'open point's to external IPs to validate the integration of this architecture element in the bigger picture of the service architecture. Once the application design, test and validation is completed, the application can be deployed and configured on a public Cloud providers such as Amazon Web Services (AWS), CloudSigma, Interoute or EGI, with all the required proxies and public IPs configured on the Cloud Provider as exemplified on Figure 42. Terradue supports the partners in this step too, including for the cost assessment and the costs coverage within the project timeframe. All the "developer" resources are intended for the cost effective development and testing of future computationally intensive tasks, that will be handled and deployed by the Platform on large computing resources procured under different models (e.g. pay-as-you-go, subscription) serving operational or validation purposes and thus making the typical transition from Sandbox mode (development) to Cluster mode (deployment and exploitation).

6.1.3.2 Processing workflows

To fully explore the capabilities of parallel computing for EO data processing and Open Data repositories, the development and integration cycle of applications (processors and algorithms) need to take into account the concept of data processing workflows (a.k.a. data pipelines). These are designed to facilitate the data flow, repeatable and systematic tasks, effective large-scale data management, multidisciplinary analysis, and adequate data source identification. The workflows support the automation of methods and the incorporation of best practices in data management that result in extended capabilities for validating and reproducing experimental outcomes, and they also encourage the use of new development methods. As such, workflows are used to describe series of structured activities and computations, and provide a means to specify and enact a service.

From a computational perspective, workflows are directed acyclic graphs, where the nodes correspond to a step in the analysis operations, which can be supplied locally or by third-party web services, and where the edges specify the flow of data between those operations. Powered by a workflow scheduler, they inherit a control dependency that ensures that new actions can't run until the prior actions are completed over the data chunks to be ingested, by specifically controlling flow and action nodes. The flow nodes provide the mechanism to control the execution path fan-out, fan-in and decision. The action nodes are the mechanism by which a workflow triggers the execution of a processing algorithm.

Besides being useful to describe and execute computations, workflows enable the encoding of scientific methods and know-how. Hence, they are valuable objects for several reasons:

- They allow the assessment of the reproducibility of results;
- They can be reused by the same or by a different user;
- They can be repurposed for other goals than those for which originally built;
- They validate the implemented scientific method and its results in pursue of new insights;
- They illustrate how to take advantage of existing data management infrastructure in a particular discipline.

To implement this paradigm it takes advantage of the capabilities of 'Hadoop Map/Reduce Streaming' for legacy applications integration, where processing algorithms are able to access distributed data holdings and to scale over computing clusters. When an executable is specified for the ingestion of input splits, several Map tasks will launch the executable as separate processes (the Reduce task being configured as 'do nothing'). When an executable is specified for the aggregation of intermediate results, a single task is launched with its input streams coming from the previous workflow step outputs. Non-legacy applications can exploit other computational models (Spark, Storm, distributed shell among others). These features are made available directly as a utility able to create and run jobs with any executable or script (legacy applications).

To orchestrate dependencies between these data processing jobs, one robust and well-used solution is Apache Oozie. This workflow scheduler is a collection of actions nodes that are the mechanism by which a workflow triggers the execution of a computation/processing task. On top of the workflow management, Oozie also allows the definition of coordinators that run workflows based on time (e.g. run it every hour) and data triggers (e.g. wait for my input data to exist before running my workflow). The top layer of Oozie, the Bundle Engine provides a higher-level abstraction that enables the batch of a set of coordinator applications improving the operational control of the workflows.

6.1.3.3 Integration

The Developer Cloud Sandboxes service on Terradue Cloud Platform allows a step-by-step approach on how to deploy use several EO toolboxes or integrate/create new applications to achieve practical results with high performance processing infrastructures. The resulting Cloud appliances may be in Sandbox mode where application is tested and debugged, or in Exploitation mode where the application is deployed in clusters of compute resources, to expand the computing capacity horizontally (computing, storage and network).

This approach allows existing toolboxes and software to be integrated and exploited on distributed compute infrastructures without requiring the re-engineering in a new programming language. It offers a practical entry point and capacity building vehicle in preparation for the future of massively distributed processing operations.

To take full advantage of these features, the service developers are invited to follow the basic requirements baseline that ensures the application integration on a Cloud infrastructure. To ensure this compliance, the development team should be able to go through the:

- Definition of Parallelisation Strategy: where the service developer defines the DAG of the service showing the different jobs, inputs and outputs of the workflow. This graph will show the parallelisation strategy to be applied on the Cloud Environment;
- Definition of Data Sources: the data requirements need to be analyzed and their retrieval mechanism accessed to make sure that the data is available in the system to be consumed as expected. In this assessment, not only the technical aspects will be considered but also the legal (data policies enforcement) and financial ones (e.g. commercial data);
- Analysis of Tools and Libraries: the tools and libraries necessary to execute the applications need to be analyze and evaluate their compatibility with the computational resources available. Also here it is necessary to take in consideration the technical, legal and financial aspects for each tool.

Once integrated in the provided framework, the application can scale under the automatic management (job distribution and task trackers over a compute cluster). The resulting applications are exposed via the OGC Web Processing Service (WPS) interface to perform the end-to-end validation and subsequently exploit the application in a B2B environment.

6.1.3.4 Additional application programming models

The service development is facilitated by the access to a Sandbox (virtualised remote development environment) according to the specification defined in the design, where it is possible to develop, test and validate the service on an environment ready to be connected to a Cloud provider. In this integration phase, a Cloud Sandbox model provides a test environment very similar to an operational environment, where the applications are proof-tested against large EO dataset series. Also relevant is that service integration is not limited to the Hadoop MapReduce Streaming programming model, and it is possible to support other application programming models. The platform provides a resource-management element responsible for managing computing resources addressing a larger set of application typologies and includes the support among others:

- Shell commands and scripts in containers on multiple nodes in a cluster;
- Complex directed acyclic graph (DAG) process where the tasks are spread across stages to be run as a single, all-encompassing job;
- Iterative graph processing system built for high-scalability based on the Bulk Synchronous Parallel (BSP) distributed computing model;
- Processing of real-time data streams with Apache Storm and Spark;
- Combining MapReduce and MPI.

The availability of the different programming models positively impacts the VRE user's programming practices and takes into account a critical aspect of the current knowledge and maturity of data processing algorithms: human in the loop is a very dominant constraint, and progressing towards more automated chains operating on Cloud Platforms need to go through programming models allowing more integration points for intermediate results analysis. The applications created with this framework have the main advantage to be self-contained where a single package (specifically in RPM format) contains the application resources and the dependencies specification. The contextualisation process is optimised with the definition of specialised baselines with the packages pre-installed (e.g., IDL, Matlab). Following the proposed approach, the application packaging will include dependencies specification that improves the scientific reproducibility and the identification and management of dependencies.

6.1.3.5 Application packing

There are different valid strategies for packaging and deploying an Application in a Cloud environment. The main goal is to have an application able to run in a predictably manner in a large computing production environment, potentially formed by hundreds of nodes. The simplest approach for packaging is the "snapshot" or also known as the “Golden image”. Its lifecycle is quite straightforward: when the development of the application is finished, the application and the underlying operating system are frozen and then moved in production to be executed (i.e., using multiple resources). In this case the dependencies are included in the snapshot that thwarts the maintainability and versioning of the service. There are a few major disadvantages to this approach due to the large manual intervention to prepare the snapshots, dependencies not being explicitly specified and, most importantly, the precise reproducibly cannot be guaranteed.

The other preferable approach is to guide the user (in this case a developer or application integrator) during the Sandbox mode using automation tools and online documentation, in order to build an application compatible with the platform's applications. Such applications have the main advantage to be self-contained. In other words, a single package (in RPM format) contains the application resources and the dependencies specification. To complement the approach, the contextualisation process can be optimised with the definition of specialised baselines with packages pre-installed (e.g., IDL, Matlab). Following the proposed approach, the application packaging will include dependencies specification that improve:

- Scientific reproducibility;
- Dependencies identification and management;
- Maintainability from the Operations Team perspective and avoid disk pilling with version of the application on the Cloud Controller;
- Portability in different Cloud providers, avoid the snapshot build and import on the specific Cloud Provider (a manual process).

In both cases, the wide usage of virtualisation and the possibility to start virtual environments within Cloud services significantly simplifies creation of environments and provisioning of resources. However, it still leaves a problem of portability of complete environments where applications are developed unresolved. Previous virtualisation solutions are not light and flexible enough partially due to the size of the images. Docker fills this gap providing with a platform designated both for developers and system administrators. Docker is an open platform for developers to build, ship, and run distributed applications. Composed of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a Cloud service for sharing light images, Docker enables applications to be quickly assembled from components and eliminates the friction between development and production environments. As a result, user applications can ship faster and run the same process developed in the Developer Cloud Sandbox to large multi-tenant data centers in the Cloud. The main difference between a Docker container and a Virtual Machine lies in the fact that a Virtual Machine consists of, besides of the application itself, also an operating system with all binary files included. Docker images work as an isolated process in the host operating system, which shares a kernel with other containers. Thereby, still using benefits of virtualisation, it is more portable and more effective – an application uses less disk space.

The platform allows launching Docker containers directly as containers (this is addressed in detail in the following section). Basically, this solution lets the developers package their applications and all of the dependencies in a Docker container to provide a consistent environment for execution and also provides isolation from other applications or software installed on host. Processes are therefore running in an isolated processing environment exactly the same as the developer built for it.

From the existing Developer Cloud Sandbox model, the process to build a Docker image is fairly straightforward. Indeed, all files and dependencies of the application are self-contained and thus easily installable on the Docker image baseline, the same on which the user has integrated his/her application (but stripped from useless content in a production environment). The building of the Docker image is part of the application integration output set. When an application is deployed on the processing environment where the resource manager runs it is only necessary to have a simple Docker engine operation to deploy the image on the processing container (see section below).

6.1.4 Cloud production centre

The Cloud production centre delivers a Production Platform-as-a-Service that provision the cluster based on the complete Apache Hadoop “Yet Another Resource Negotiator” (YARN) stack. The cluster manager ensures a unified and automatic deployment process, which is the management application that administrates Hadoop cluster at any scale. It installs the selected elements of the processing stack (Hadoop, Spark, Oozie, Docker, ...) in minutes and ensures optimal settings as well as configuration and security management from a single console. It allocates cluster resources by workload or by user/group/application to eliminate contention and ensure Quality of Service (QoS).

6.1.4.1 Computational framework

YARN builds and evolves on the lessons learned of Apache Hadoop MapReduce (Hadoop version 1) evolution over the past decade. Apache Hadoop YARN brings a new architecture with a strong code reuse from the previous Hadoop frameworks (for instance it guarantees the backward compatibility with MapReduce v1 applications). The YARN community built (and keeps on building) this new framework with the following requirements baseline:

- **Scalability**: enable horizontal scaling to tens of thousands of nodes and concurrent applications;
- **Serviceability**: enable cluster software evolution decoupled from users’ applications;
- **Multitenancy**: support multiple tenants to coexist on the same cluster and enable fine grained sharing of individual nodes between tenants;
- **Locality awareness**: move computation to the data;
- **High cluster utilisation**: enable high utilisation of the underlying physical resources;
- **Secure and auditable operation**: enable cluster resources secure and auditable operation;
- **Reliability and availability**: enable reliable user interaction and support high availability;
- **Programming model diversity**: enable diverse programming models and evolve beyond just being MapReduce centric;
- **Flexible resource model**: enable dynamic resource configurations on individual nodes and a flexible resource model;

- **Backward compatibility**: completely maintain backward compatibility of existing MapReduce applications.

YARN provides several advantages over the Hadoop framework, including a better scalability, cluster utilisation and user agility and introduces the ResourceManager component that acts as a resource scheduler and sole arbitrator of cluster resources. Applications (including MapReduce jobs) ask for specific resource requests via the ApplicationMaster component, which in turn negotiates with the ResourceManager to create an application container within the cluster. The backward compatibility requirement keeps MapReduce at the core of the Hadoop 1.0 tasks but the introduction of YARN expands the capability of a Hadoop environment to go beyond the basic MapReduce process. Although several Big Data players have adopted MapReduce as the basis for large-scale processing, it may contain limitations in certain problem domains. YARN addresses these needs by allowing to run non-MapReduce jobs, and it does so by providing a generic resource management framework for implementing distributed applications. Dragged by this major architectural overhaul, under YARN, MapReduce is known as MapReduce version 2 (MRv2) and is merely one of the supported applications frameworks within Hadoop. The other application frameworks (e.g. graph processing, MPI, TEZ, Spark, Storm, etc.) are known as non-MapReduce frameworks.

YARN introduces a Capacity Scheduler as the default scheduler and includes the *Fair Scheduler* as another pluggable scheduler. The Capacity Scheduler addresses the sharing of clusters between organisations to run multi-tenant installations leading to the improvement of the utilisation, performance and potential for a more intelligent scheduling. All together, these elements lead a cost effective scheduling and YARN's Capacity Scheduler builds on top of:

- **Elasticity with multi-tenancy** - Resources are shared between individuals, teams and groups in an elastic fashion. Whenever there are free resources, these can be allocated to any entity as long as they remain underutilised otherwise. Instead, when there's a sudden demand for those resources, they are pulled back with minimal impact on the SLA of the previously entitled entities. The Capacity Scheduler supports this with the definition of queues, minimum user percentages and limits. The sharing of resources is done without impacting co-tenants.
- **Security** - to mitigate the security concerns in multi-tenant clusters, the Capacity Scheduler includes the queue-level Access Control Lists mechanism to address cross-organisation security-related issues.
- **Resource awareness** - YARN orchestrates applications with heterogeneous resource requirements based on CPU and memory requirements. This orchestration is under scheduling policies taking into account memory and CPU requests of submitted applications and support the dynamic needs of the ApplicationMaster(s).
- **Granular scheduling** - the granular scheduling targets avoiding the loaning of full nodes to tenants (full node assignment compromises utilisation). YARN does that by defining queues as logical views of resources on nodes.
-

One of YARN's strengths is to support other (beyond MapReduce) application programming models that target addressing a larger set of application typologies. Here is the list a few of the programming models to be available to the VRE:

- **Distributed-Shell** - a simple mechanism for running shell commands and scripts in containers on multiple nodes in a Hadoop cluster.
- **Apache Tez** - Tez generalises the complex directed acyclic graph (DAG) process where the tasks are spread across stages to be run as a single, all-encompassing job.
- **Apache Giraph** - an iterative graph processing system built for high-scalability based on Google's Pregel. Giraph is based on the Bulk Synchronous Parallel (BSP) distributed computing model.

- **Apache Spark** - it targets applications (iterative algorithms, machine learning, data mining) where keeping data in memory helps performance.
- **Apache Storm** - addresses the processing of real-time data streams.
- **Hamster** (Hadoop and MPI on the Same Cluster) - this is a work-in-progress to provide support for MPI in YARN containers.

The list above is not exhaustive and several open source initiatives work to support the development of programming models on top of YARN resource management platform that provides services such as scheduling, fault monitoring and data locality (among others).

6.1.4.2 Cloud containers

As introduced in the previous sections, Docker containers can be launched launch directly as YARN containers. Basically this solution lets the developers' package their applications and all of the dependencies into a Docker container to provide a consistent environment for execution and provide isolation from other applications or software installed on host. Processes are therefore running in an isolated processing environment exactly the same as the developer built for it. The process from Development to Operation becomes transparent and straightforward with the combination of templates and Docker containers that reproduce the exact same runtime environment from integration to production. Docker containers also allow exploiting multi-tenant clusters for a better-cost effectiveness by isolating runtime environment for each user application on the same production cluster. The container concept can also evolve to native clustering where distributed containers are optimally scheduled within the same clusters. Furthermore, monitoring is a key point of the design to optimise the decision making at all levels with a decentralised and fault tolerant system based on agents that check clusters consistency and health continuously. Finally, a capacity manager helps on making decisions when applying different SLA policies on a multitenant cluster and works with two types of Hadoop clusters: static and dynamic.

A cluster is static when capacity can't be increased horizontally. Hardware resources are already provisioned and adding new nodes can't increase the total capacity. For that type of clusters, the Capacity Manager checks the job submission process, monitors the applications and applies SLAs such as:

- Application ordering – can guarantee that a higher priority application finishes before another one (supporting parallel or sequential execution);
- Moves running applications between priority queues;
- Attempts to enforce time based SLA (execution time, finish by, finish between);
- Attempts to enforce guaranteed cluster capacity requests (x % of the resources).

A cluster is dynamic when the cluster capacity can be increased horizontally. This means that nodes can be added or removed on the fly – thus the cluster's total capacity can be increased or decreased based on the cluster load and scheduled applications. The Capacity Manager is able to add or remove nodes from the cluster based on the SLA policies and thus continuously providing a high quality of service for the multi-tenant Hadoop cluster. Given the option of provisioning or decommissioning cluster nodes on the fly, the Capacity Manager allows applying the following set of SLAs:

- Application ordering – can guarantee that a higher priority application finishes before another one (supporting parallel or sequential execution);
- Moves running applications between priority queues;
- Enforce time based SLA (execution time, finish by, finish between) by increasing cluster capacity and throughput - Smart decommissioning – avoids HDFS storms, keeps paid nodes alive till the

last minute - Enforce guaranteed cluster capacity requests (x % of the resources) - Attach priorities to SLAs.

6.1.4.3 Provision scenarios

This section describes and assesses two typical situations that the system manages to ensure cost effectiveness during intensive processing campaigns.

The first is minimising the cost for stable and predictable for processing demands. In this case the Capacity Manager allows the platform operator to configure SLA policies for the processing clusters and scale up or down on demand. The operator may set alarm triggers and notifications for different metrics like pending job requests, unresponsive hosts or over-memory usage, etc. For this, the operator configures a policy based on time interval that instructs the Capacity Manager to shrink the cluster down to a specific number of nodes after a systematic processing period is concluded and grow it back by the time new working data is available.

A second focuses on the cost optimised provisioning for on-demand processing on multi-tenant clusters. In this scenario users may have unpredictable processing needs requesting their self-integrated service they want to test or to use for a limited time. Using the containers approach described in the previous sections, it has been demonstrated how user application are managed on a multi-tenant cluster with the usage of the containers. Once the application is deployed and ready to be invoked, there is a priori no way to foresee the requests. Nevertheless, users can configure specific SLA that will empower the capacity manager to regulate the job submissions in order to optimise the load and cost of the cluster.

6.2 Workflow management

6.2.1 Overview

Workflow as a concept was defined as follows:
"The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules[51]"

Scientific workflows are well-known means to encode scientific knowledge and experimental know-how. By providing explicit and actionable representations of scientific methods, workflows capture such knowledge and support scientific development in a number of critical ways, including the validation of experimental results and the development of new experiments based on the reuse and repurposing of existing workflows [2].

Workflows serve a dual function: first, as detailed documentation of the scientific method used for an experiment (i.e. the input sources and processing steps taken for the derivation of a certain data item), and second, as re-usable, executable artefacts for data-intensive analysis. Scientific workflows are composed of a variety of data manipulation activities such as Data Movement, Data Transformation, Data Analysis and Data Visualisation to serve the goals of the scientific study.
In this section, the Workflow Management approach is describe taking into account the Research Objects paradigm. It will be introduced the concept of Workflow Management Systems, showing several tools, and the design and modelling of workflows and an API component for managing workflows will be explained.

[51] http://www.taverna.org.uk/introduction/why-use-workflows/

6.2.2 Workflow Management Systems

A Workflow Management System (WFMS) provides an infrastructure for the set-up, performance and monitoring of a defined sequence of tasks, arranged as a workflow[52]. From Earth Science perspective, a WFMS is a piece of software that provides an infrastructure to setup, execute, and monitor scientific workflows. In other words, the WFMS provides an environment where in-silico experiments can be defined and executed[53].

These systems have to be able to orchestrate efficiently the workflows executions, building a well-coordinated pipeline of the individual components that constitute the workflow. Also, these tools need to resolve different incompatible data formats that various services produce or consume. In order to address these issues, Workflow Management Systems have emerged to provide an easy way to generate this kind of experiments in the form of workflow executions [60].
For Earth Science needs the features that should be offered by a Workflow Management System would be *Workflow Pattern Flexibility*.

The framework has to provide wide flexibility in the process of building workflows, allowing as much as possible open design and multiple patterns to the users. This feature is crucial at the time scientists decide to use a WFMS; they will choose that one which offers more freedom and possibilities for creating scientific workflows.

- Formal Modelling. The WFMS has to modelise metadata of the workflows and executions in a formal and structured way in order to can export this information, making available the workflow data to other tools which will be able to use this content to run its own executions, generate reports and stats, annotate resources, etc.
- Document and Components Integration. The integration of different components (e.g. WSDL, RESTful, SOAP, XML, etc.), documents and code in different formats (e.g. Excel, R scripts, Python scripts, Java Beanshell, etc.) is a very important feature in Workflow Management Systems to help in the flexibility and open design of workflows.
- Easy-way to include code and customisation. An infrastructure that allows easily the addition of piece of code in components and a high level of customisation in workflows will achieve more impact in the scientific communities.
- Graphical Representation. Visualisation of the workflows in a graphical way helps users to understand the interaction of the components, services and the process itself. The workflow representation usually includes the components within boxes and links each other through arrows.
- Easy-access to external resources and catalogues. WFMS should provide mechanisms for accessing and integrating external resources and catalogues in order to facilitate the work of the scientists when they need to get data as inputs in the workflows.
- Integration of plugins and custom services. The possibility of the inclusion of plugins in the WFMS allows improving the customisation and freedom for the requirements of the users.
- Errors Notification in Workflows Executions. The WFMS has to be able to highlight problems along the workflows executions, showing in the screen the errors notifications and the run traceback in order to the users can know the root of the issue and solve it easily.
- User-friendly to scientists. The interface and functionalities of the WFMS should be user-friendly and should be prepared to be used by non-technical people. This implies that these frameworks should be understandable by people from different disciplines without the need of employing many times in training activities of the tool.

[52] https://en.wikipedia.org/wiki/Workflow_management_system

[53] http://www.taverna.org.uk/introduction/what-is-a-workflow-management-system/

There are a lot of Workflow Management Systems implementations available, which can fit for the requirements, and goals of Earth Scientists in EVER-EST project. The workflow management system implemented in the EVER-EST environment is based on Taverna, which is described below.

6.2.2.1 Taverna[54]

Taverna is an open source and domain-independent Workflow Management System, which offers a suite of tools used to design and execute scientific workflows and aid in silico experimentation. Taverna has been created by the myGrid team[55] and is currently funded through FP7 projects BioVeL[56], SCAPE[57] and Wf4Ever[58].

The Taverna tools include the Workbench (desktop client application), the Command Line Tool (for a quick execution of workflows from a terminal), the Server (for remote execution of workflows) and the Player (Web interface plugin for submitting workflows for remote execution). Taverna Online lets you create Taverna workflows from a Web browser. The main features included in Taverna are:

- Access to local tools/scripts and remote resources and analysis tools, Web and grid services;
- Not restricted to predetermined services. Rapid incorporation of new services without coding;
- Multiple domains supported (e.g. bioinformatics, chemistry, astronomy, etc.);
- Excel and CSV Spreadsheet support;
- R scripts[59] software for statistical computing and graphics;
- Interaction with a running workflow from your Web browser.
- Graphical representation of workflows;
- Creating and sharing workflow fragments as reusable components;
- Standards-compliant workflow run provenance collection;
- Data integration for different catalogues;
- Integration with other tools (e.g. myExperiment[60], BioCatalogue[61]);
- Fully documentation and tutorials.

Besides the wide number of features provided by Taverna, it has the big advantage to speed-up the adoption of the Research Objects because of its closely integration along Wf4Ever project. In Wf4Ever Taverna was considered as the WFMS seed to develop the concept of workflows and Research Objects. In Taverna, users can install a plugin that allows to export/save the provenance of the workflow run (inputs, intermediate and output values) as a Research Object Bundle (file format for storage and distribution of

[54] http://www.taverna.org.uk/

[55] http://www.mygrid.org.uk/

[56] http://www.biovel.eu/

[57] http://www.scape-project.eu/

[58] http://www.wf4ever-project.org/

[59] https://www.r-project.org/

[60] http://www.taverna.org.uk/introduction/taverna-features/myexperiment-integration/

[61] http://www.taverna.org.uk/introduction/taverna-features/biocatalogue-integration/

Research Objects as a ZIP archive). This Bundle can then be consumed by ROHub[62], generating a new Research Object from the zip file.

Also during the RO Hackathon in EVER-EST project in January 2016, the Virtual Research Communities were trained in this tool, showing its capabilities and functionalities through several tutorials. The first formation was the basic tutorial created by the developers of the software, http://slideplayer.com/slide/8106353/. Two other customised tutorials were prepared to extend the basic features, where the Users Communities could to manage new functionalities and see new services. (Tutorials: An extension of basic description, and a workflow to get forecast weather of a country, Figure 43).

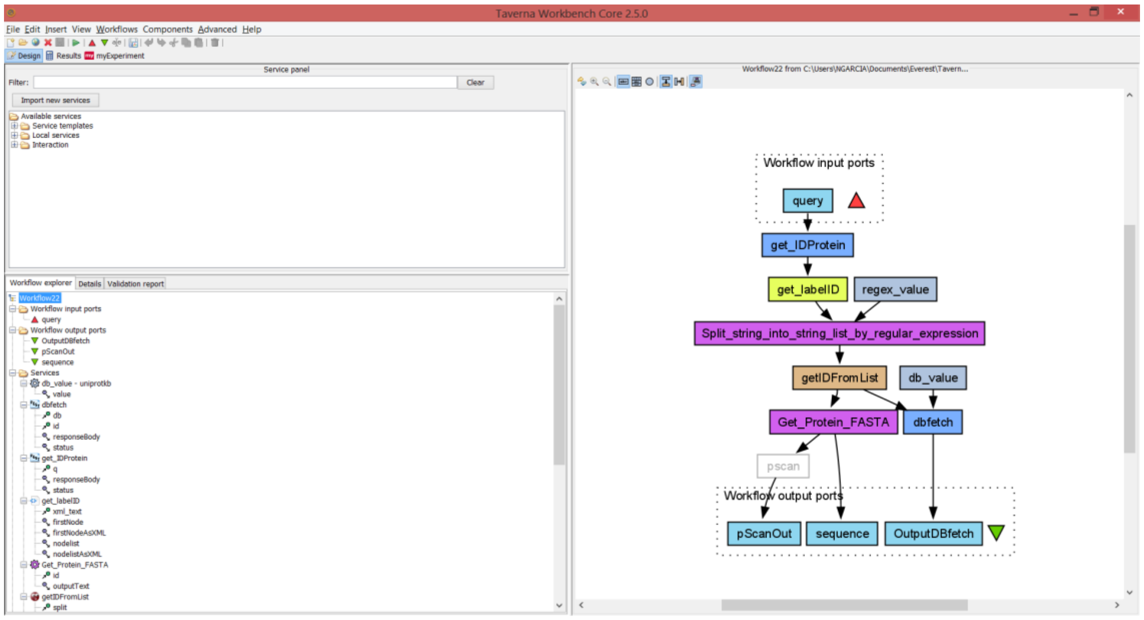


Figure 43 Taverna tutorial - basic example

6.2.3 Workflow modelling

A Workflow-Centric Research Object can be viewed as an aggregation of resources that bundles a workflow specification and additional auxiliary resources, including documents, input and output data, annotations, provenance traces of past executions of the workflow, etc.

For the concrete specification of Workflow Modelling there are several ontologies and vocabularies, which can be used in the EVER-EST paradigm. These ontologies are the following:

- **wf-desc:** A vocabulary for the description of workflows. This provides an abstraction that can be mapped to different particular workflow systems. The `wfdesc` ontology describes an abstract workflow description structure, which on the top level is defined as a `wfdesc:Workflow`. This ontology is meant as an upper ontology for more specific workflow definitions, and as a way to express abstract workflows, which could either be hand-crafted by users ("ideal workflow description") or extracted from workflow definitions of existing workflow systems, like Taverna's `.t2flow`. Further information can be found in the next URL, http://wf4ever.github.io/ro/#wfdesc.
- **wf-prov:** A vocabulary for the description of provenance information. This provides an abstraction that can be mapped to different provenance vocabularies. The `wfprov` ontology describes the provenance of the workflow results.

[62] http://www.rohub.org/portal/home

- A `wfprov:WorkflowRun` describes the activity of running a `wfdesc:WorkflowInstance`. Information in, http://wf4ever.github.io/ro/#wfprov.
- **Workflow Motif Ontology (WFM):** The Workflow Motif Ontology outlines the kinds of data-intensive activities that are observed in workflows (*data-operation motifs*) and the different manners in which activities are implemented within workflows (*workflow-oriented motifs*). These motifs are helpful to identify the functionality of the steps in a given workflow, to develop best practices for workflow design, and to develop approaches for automated generation of workflow abstractions. Further information can be found in the ontology description document, http://vocab.linkeddata.es/motifs/.

For Taverna, in Wf4Ever project has developed a plugin for exporting W3C PROV-O[63] compliant workflow run provenance as RDF, called taverna-prov[64]. This provenance is based on two extensions of PROV-O, the Wf4Ever wfprov model and a Taverna-specific model TavernaProv[65]. This plugin has been used to generate a publicly available Provenance Corpus[66], which contains traces of workflow runs, (including inputs, outputs and intermediate values) of 134 Taverna workflow together with 70 WINGS workflows.

Information in detail about this integration in Taverna can be found into the official website of Taverna platform, http://www.taverna.org.uk/developers/work-in-progress/workflow-preservation/

6.2.4 Workflow runner

Scientists and researches need the validation of experimental results and the development of new experiments every day in their work. Many times they make these tasks based on the reuse and repurposing of existing workflows. Therefore, scientific workflows play an important role for sharing, exchanging and reusing scientific methods [1].

In order to assess if a workflow is functional, it is generally useful to be able to (re)-execute a workflow. But different workflow systems have different ways of running a workflow. For instance, Taverna has the Taverna Server, while Wings has a portal and a Pegasus/Condor engine in the backend.

Workflow Runner component, developed in Wf4Ever project, provides a common lightweight interface for addressing these needs required by scientist people. Moreover, it allows managing workflows in an agnostic and independent way from WFMS. These services offer an API where users can count with features such as "Run this workflow please" and "Show me the data from that workflow run".

Workflow Runner API mirrors the RODL API, but the ROs exposed by this service each represent a particular workflow run structured to show inputs, outputs, console logs, provenance and annotations containing wfprov and wfdesc mappings.

To access the root of the service, the system should redirect to a default server runs resource. From here the client may either:

- POST a new workflow run, providing as a minimum the workflow definition;
- GET a list of existing workflow runs;
- DELETE existing workflow runs.

Navigating the workflow runs would allow inspection of workflow status, outputs and other resources exposed by the underlying workflow server.

[63] http://www.w3.org/TR/prov-o/

[64] https://github.com/wf4ever/taverna-prov/

[65] http://ns.taverna.org.uk/2012/tavernaprov/

[66] https://github.com/wf4ever/provenance-corpus

Resources are located using specific properties in the RO manifest for the workflow run. For example the property description 'runner:workflow' is used in the workflow run description to link the workflow run with the main workflow to run, such as uploaded on RO creation.

All formats are based on RDF in text/turtle[67] and application/rdf+xml (by content negotiation) unless noted otherwise. The Workflow Runner API should support the next features:

- Finding default workspace to redirect to the RO workspace of the default server. HEAD or GET on this entry point should redirect to a workspace of workflow runs on the default server.
- Retrieve runs in workspace to see current runs. The list of server runs is represented as a RODL workspace, where each RO represents a run. Each URI returned, if any, should point to a Research Object representing a workflow run.
- Submit new run to workspace to create a new run. Creating a new run is similar to creating a new Research Object, but requires the content-type text/uri-list to include the URL for the workflow definition to run. The returned location refers to a Research Object representing the run.
- Retrieve run to view a run and its resources. A workflow run is represented as a Research Object, thus retrieving it will redirect to a manifest listing its constituent resources.
- Retrieving the workflow status to check the current status. Retrieving the resource indicated with '*runner:status*' in the manifest must return the current status of the workflow run.
- Changing the workflow status to initiate running of the workflow. The client can request desired state transitions by PUT-ing to the status resource. The client must include one and only one of the available Workflow Runner statuses (initialised, ready, queued, running, failed, finished, cancelled or archived), but may also include third-party statuses.
- Retrieving the outputs when the workflow has status http://purl.org/wf4ever/runner#Finishedhttp://purl.org/wf4ever/runner - Finished. Outputs are shown as a folder structure, similar to inputs, by following the *'runner:outputs'* link in the manifest.

More information in detail can be found in W4fEver documentation,
https://github.com/wf4ever/apis/wiki/Workflow-Runner-API

6.2.5 References

[1] Gómez-Pérez, J. M., García-Cuesta, E., Garrido, A., Ruiz, J. E., Zhao, J., & Klyne, G. (2013). When history matters-assessing reliability for the reuse of scientific workflows. In *The Semantic Web–ISWC 2013* (pp. 81-97). Springer Berlin Heidelberg.
[2] Gómez-Pérez, J. M., García-Cuesta, E., Zhao, J., Garrido, A., Ruiz, J. E., & Klyne, G. (2013, May). How Reliable is Your Workflow: Monitoring Decay in Scholarly Publications. In SePublica (pp. 75-86).

6.3 Preservation

6.3.1 Overview

Digital preservation is a formal endeavour to ensure that digital information of continuing value remains accessible and usable. It involves planning, resource allocation, and application of preservation methods and technologies, and it combines policies, strategies and actions to ensure access to reformatted and "born-digital" content, regardless of the challenges of media failure and technological change. The goal of digital preservation is the accurate rendering of authenticated content over time. Other way to say is that Digital

[67] https://www.w3.org/TR/turtle/

preservation is the method of keeping digital material alive so that they remain usable as technological advances render original hardware and software specification obsolete[68].

The promotion of a preservation e-infrastructure requires a consistent effort to establish a culture where long-term preservation is an integral part of data management procedures. A shared approach to knowledge preservation, both at a logical and technical level, would enhance the state of the art, especially for those organisations, which lack funding for data preservation activities.

In EVER-EST Project this task aims at the application of Long-Term Digital Preservation principle and techniques to Earth Science datasets and Research Objects, which are considered as a particular kind of Digital Objects. The RO lifecycle will be carefully analysed and the relevant (meta-) information coded, stored and preserved along time. An essential set of preservation services will be included in the VRE in order to ensure the availability of data, workflows and results also many years after the initial execution of an experiment.

6.3.2 Design and research lines

This section describes the technical, architectural and research challenges to design an infrastructure, which preserves all the information, required by the Virtual Research Communities and by data providers. In preservation design are addressed two main perspectives to achieve the VRCs vision and goals in EVER-EST project: preserving Earth Science datasets and modelling a stable architecture, which provides and ensures the preservation requirements. The next subsections deal with these two features for preservation schemas.

6.3.2.1 Earth Science data preservation

Earth Science data (and science data in general) are subject to a continuous cycle of: creation, transmission, storing, consuming and then, often reprocessing/combining data to start the cycle all over again. The "Earth Science world" is therefore both producer and consumer creating a feedback loop between ES data and research results generating new data during the scientific processes that starts to change the very nature of research.

In addition, for many Earth Sciences branches the capture of key observational data may be difficult or impossible to repeat. At the same time such unrepeatable observations may be a critical input to environmental, economic and political decision making. This enhances the needs for secure and affordable preservation strategies and techniques to avoid unrepeatable measurements' loss in time.

Digital preservation will add little value to the research process if it serves only as an alternative form of storage from which duplications are produced for use with conventional methods. Preserving digital materials in formats that are reliable and usable, however, will require long-term maintenance of structural characteristics, descriptive metadata, and visualisation, computational, and analytical capabilities that are very demanding of both mass storage and software for retrieval and interpretation.

6.3.2.2 Data providers' perspective

The scope of Earth Science long-term data preservation should not be intended as simple raw bits preservation but should focus on its critical aspect, which is to guarantee in time the access to data (i.e. comprehension or understandability) and data usability (i.e. data visualisation, processing or value added product generation capability).

This heritage ensemble forms what is referred to as "data and its related knowledge" and is constituted by three main categories of digital objects:

[68] https://en.wikipedia.org/wiki/Digital_preservation

- Data – including raw data and higher level products;
- Documents – including user manuals, product description, etc.;
- Software – including software processing, visualisation, etc.

When it comes to "what" is needed to preserve an Earth Science dataset the discussion must turn very specific and cannot provide "generic" guidelines, which could be applicable in other domains. It is necessary to enter the ES field, unfold and analyse the whole process, which is behind Earth Science measurements (or campaigns or missions).

This is needed to identify all digital objects involved and workflows that are propaedeutic to a scientific measurement (or campaign or mission) and to understand the logic that connects all these elements and steps into consistent preservation ontology. The "Preserved Dataset Content" (PDSC) is the result of this exercise.

On the Data Producer or Archivist side, the PDSC is a sort of checklist that can be used to identify and retrieve within the archive – or more often the hard disks of single individuals – the elements that are necessary to ensure the dataset understandability and usability in time. Some of those elements can be categorised as obligatory, some others as optional. It is clear and agreed that the Earth Science domain is quite complex and heterogeneous, therefore specific PDSC instances shall be developed for the different cases.

6.3.2.3 PDSC: What needs to be preserved

For every Earth Observation mission, Oceanographic campaign, Atmospheric balloon launch and in situ experiment and etc. when it comes to the collection of digital objects that constitutes the "generic" Earth Science dataset, they will generically adhere to the following logic:

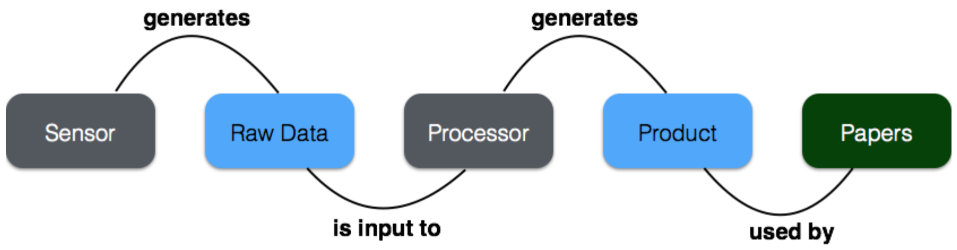


Figure 44 Basic PDSC elements

While setting up a preservation plan all these elements – therefore not only the "product" which is the dataset itself – have to be considered. Losing information on how the sensor was built and tested, or how the processor was deriving added value products from raw data could undermine the understandability of the whole dataset in the long term.

Therefore, elements to be collected refer to:

- Acquisition sensor: all design documents, technical materials, implementation plans, testing reports, feedback forms and assessment plans, calibration and user manuals which refer to the sensor;
- Raw Data (not mandatory);
- Processing Software: as for the sensor, all documents describing the processing / visualisation / accessing software should be proposed and linked;

- Product: this refers to basic products (such as Level 0 data in Earth Observation) to value added products, such as Level-3, mosaics, etc.;
- Papers: this category could refer to every document or study, which has been based on the collected data. This is currently missing in preservation plans and has been formally suggested by the EU as an enhancement. This is an on-going work: the focus should be on the logic for proper linking between single datasets, research results and related papers.

6.3.2.4 PDSC: mission phases

The common understanding about data that needs to be preserved refers only to the moment when the mission (or campaign/experiment, etc.) is actually producing data, in the form of measurements, results or observations. To fully guarantee the comprehension of these datasets, data provider shall insert a number of additional information that refer to phases of the mission itself that came before, or after, the operative phase.

The investigation about relevant campaign phases within the Earth Science domain drove to the basic agreement on the following steps (note: the term campaign is used here but one could refer to the terms mission, experiment, etc. depending on his/her field of activity):

1. *Campaign Preparation*: should include and links all those documents and data referring to the proposal, design, development, implementation, simulation and testing of a mission, campaign or experiment.
2. *Campaign Operation*: indicates the phase when the mission/campaign actually produces scientific data. This stage includes all raw data and products, as well as all the user manuals and software, which are required to interpret, access and use the data.
3. *Post Campaign*: Includes all of the logistical information required for data preservation. This needs to be refined in the future to provide additional links to external documents (e.g. linking ES data with publications).

6.3.2.5 Drafting a harmonised E-PDSC for Earth Science

The next step of our analysis has been a temporal categorisation of the above-mentioned items with the Earth Science Campaign lifecycle phases. The general item to preserve (data, software or document) should refer to one of the phases and also to a specific subcategory of the selected phase. For example, a document of a specific campaign, that refers details or description about instruments of that campaign, should be linked with an unique ID (i.e. CP-3.1) that refers to the Campaign Preparation phase where all documents about instruments should be inserted. This temporal categorisation has been performed sequentially for the all three phases, identifying a set of ID to use in the definition of a Earth Science preservation plan.

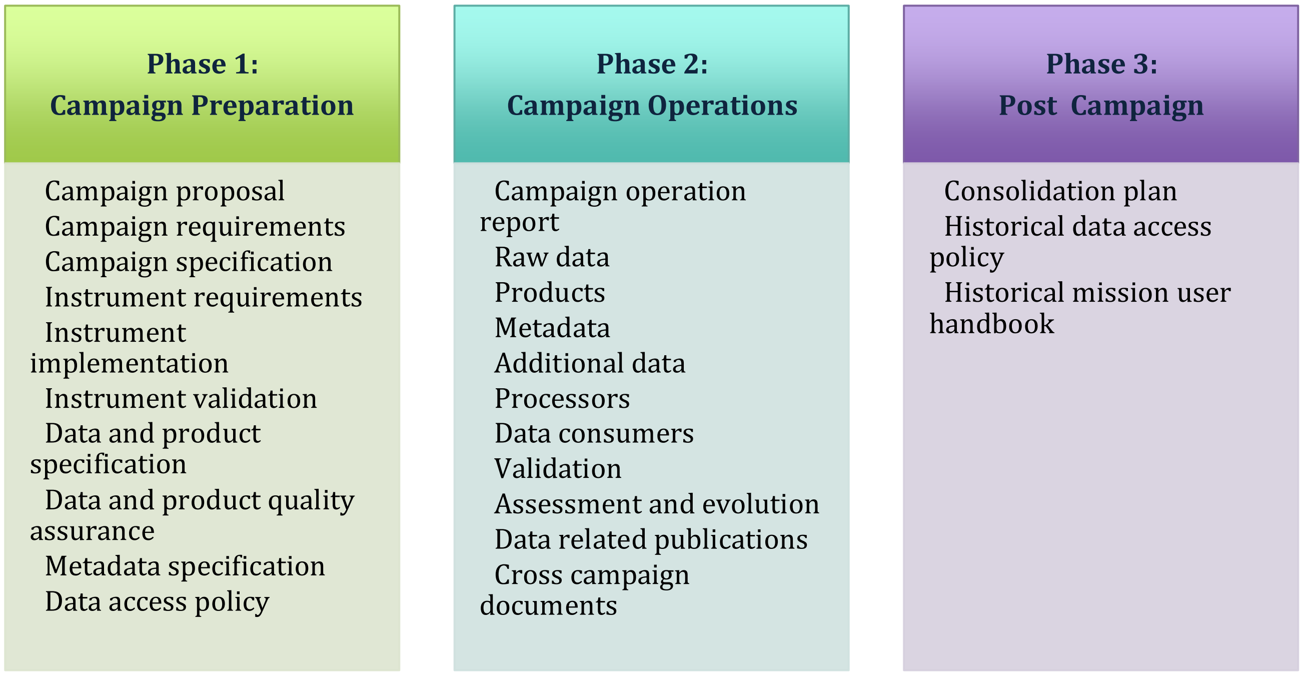


Figure 45 Main elements for each mission phase

The list of subcategories has been refined thanks to the customisation process performed with the Earth Science partners. The natural evolution of the process has been the details specifications of the three categories of items to preserve. An initial set of information details have been specified for each category (documents, data and software) and used in the customisation phase, of course it needs to be verified and improved. The details identified in the initial stage are the following:

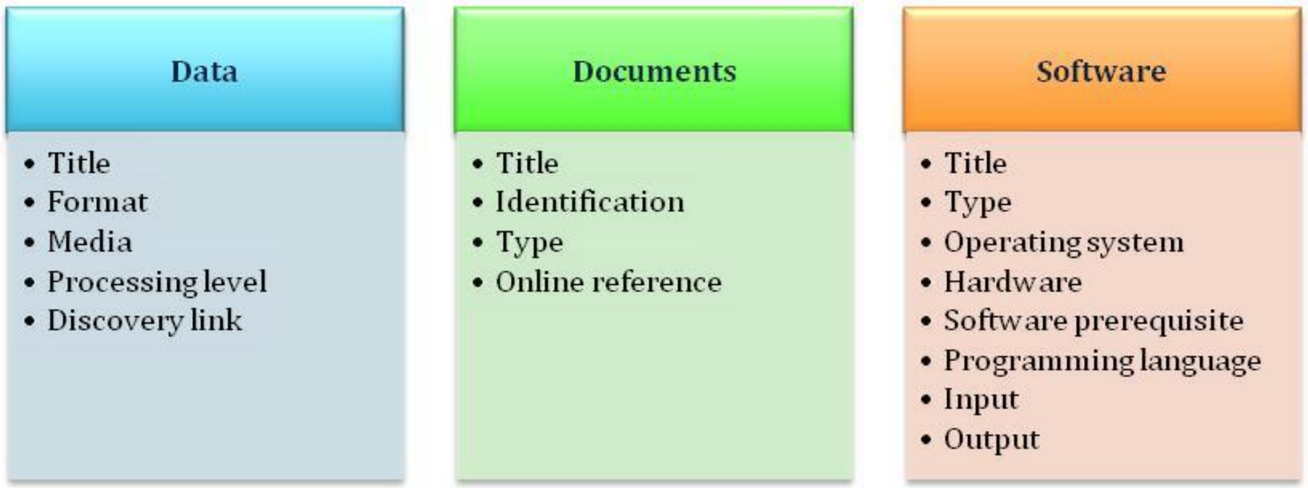


Figure 46 Initial set of information

The results of the customisation of the E-PDSC elements have been formalised in a generic Ontology, which will be used to create the network of information which is needed to preserve an Earth Science Dataset.

6.3.3 Data preservation in EVER-EST

The preservation paradigm can be addressed following different approaches regarding the Research Objects perspective. In this section two possible architectural visions to build the preservation requirements and features needed in EVER-EST project are described. These two proposals are only an initial design and they have to be agreed with the whole EVER-EST Consortium.

EVER-EST Use Case partners retrieve time series data (mainly from satellites), analyze those data with predefined mathematical models and generate results in different formats (graphs, maps, images, files, etc.). Because of this they need to encapsulate all their information in containers, which are able to preserve and curate along time the resources involved in those Earth Science Observations. The focus was on the ROs Preservation architecture design taking into account the needs for Use Cases.

6.3.3.1 Nested ROs

The first approach exploits the idea of nested Research Objects. In this sense an encapsulation mechanism to maintain time-relational ROs in a nesting way is proposed. The Figure 47 depicts graphically this idea for a better understanding. The last Research Object in time for an Earth Observation will have a complex structure with the previous ROs nested in a deep tree. The Checklist API service would allow to include a setting customisation to ensure that the before RO needed is well-encapsulated.

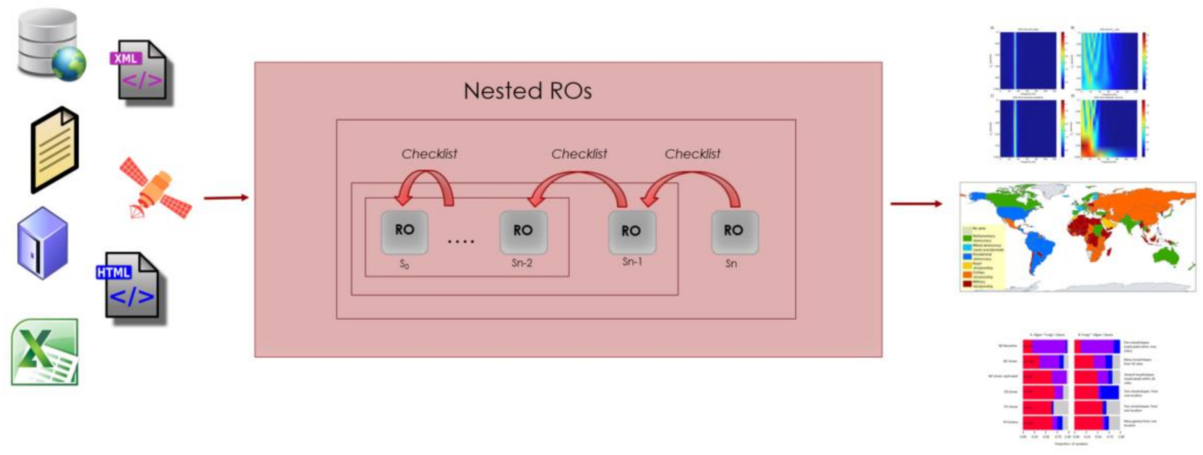


Figure 47 1st Approach: nested RO's

Probably the performance of this approach will bring scalability and memory problems, making the interaction with RODL slower and heavier because of the increase of nested information. There will be the need to evaluate this metrics to check the viability of this architectural design.

6.3.3.2 Versioned ROs

In the second approach (see Figure 48) it is proposed to maintain a unique Research Object for each Earth Observation task. The users would make updates over that RO to save changes and new resources, creating different versions for different time moments. The Research Object Digital Library (RODL) will address this design building ROs snapshots through its RO Evolution API. The system will annotate the relationship between each of such Research Object snapshots through roevo[69] as different time-dependent versions of the same Research Object. This architecture will be the most efficient in terms of performance, due to the

[69] http://wf4ever.github.io/ro/#roevo

system would only manage one RO with its snapshots. However if the Research Object reaches a complex structure, with multiple resources, links, large metadata, etc. the performance can suffer decay.

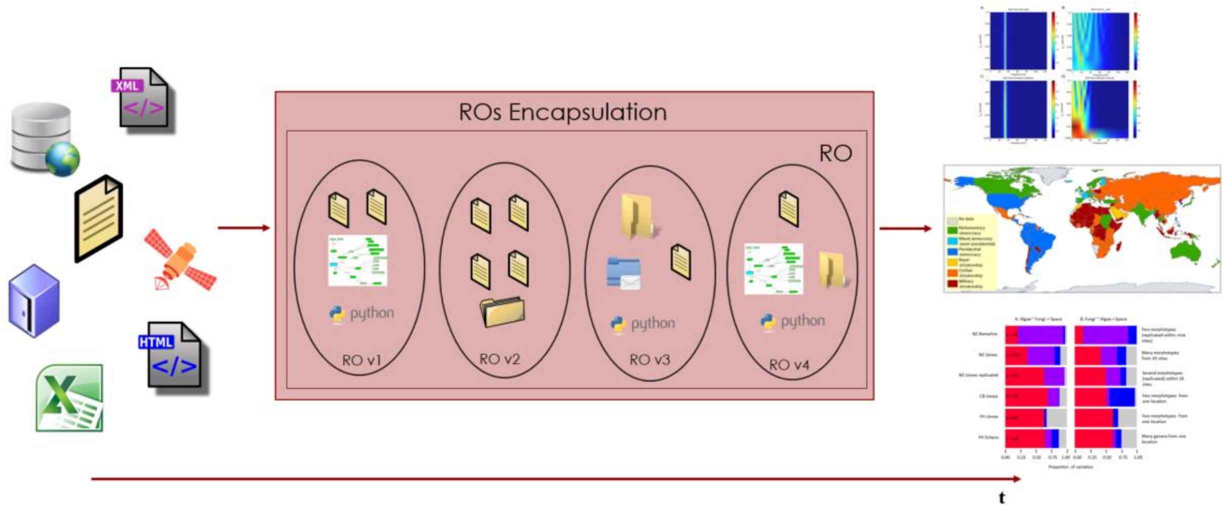


Figure 48 2nd Approach: versioned RO's

Both approaches will adopt and include the Digital Object Identifier[70] (DOIs) schema in the Research Objects. This will allow assigning persistent identifiers to ROs, developing an infrastructure that supports simple and effective methods of data citation, discovery and access. Citable data thus become legitimate contributions to scholarly communication, supporting new metrics and publication models that recognise and reward data sharing. Such initiatives are leveraging the DOI infrastructure, which is well established and already widely used for identifying research articles.

6.3.4 RO components

The Research Object infrastructure offers different components regarding preservation aspects. These services and APIs allow users to maintain, preserve and curate in an easily way their ROs and all the resources contained. Next are described the RO Components and APIs focused on the three main features needed in a preservation model; stability, completeness and reliability.
The preservation of scientific methods, in computational workflow form or in ROs encapsulation, faces challenges, which deal precisely with their executable aspects, and their vulnerability to the volatility of the resources (data and services) required for their execution. Research Objects are artefacts that bundle workflows together with other resources, can play in ensuring the preservation of scientific workflows [1].
In this context, Research Object Evolution[71] refers to the ability of managing changes in a research object and its aggregated resources by creating and maintaining different versions of the research object during its life cycle. To manage this feature RO architecture offers an available API[72], which contemplates all the needs to work with ROs Evolution and snapshots.

RO stability is the quality of an RO to remain functionally unchanged with respect to its specification and properties in the presence of changes in its structure and/or content. This is extensive to external resources linked by the RO and necessary for workflow executions, like dataset and web services owned by third

[70] https://www.doi.org
[71] https://github.com/wf4ever/apis/wiki/Wf4Ever-Services-and-APIs#RO_evolution
[72] https://github.com/wf4ever/apis/wiki/RO-evolution-API

parties. Stable ROs are more likely to support reuse of the scientific knowledge they encompass by providing a higher degree of confidence in the repeatability and reproducibility of their workflows.
The provenance of the Research Object elements (i.e., workflows, datasets, software, web services, etc.) is key to understanding, comparing and debugging scientific work. There is the need to support the logging, browsing and querying of the provenance linking components of Research Objects and the traces of workflow executions [1].

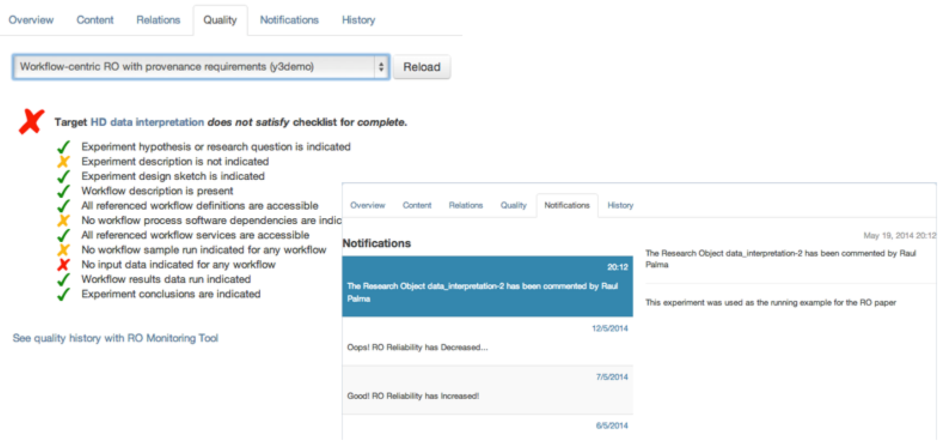
The stability[73] dimension is available as a REST service. In this way any user or consumer is able to analyse the stability of a trace of actions performed on his RO. The service is built using Jersey and JAXB. This component also contains APIs to evaluation. The stability measurement consists on the evaluation of the RO along time. The idea is to capture concrete values provided by the evaluation of the RO in different moments of its evolution. In other words, by subjecting the snapshots of a RO to the checklist evaluation[74].
For completeness, a component is provided to show the minim-based evaluation of Research Objects, i.e. ensure that all the resources and references necessary to have the complete environment are available and encapsulated in the RO [3]. This service is the Checklist Evaluation API[75].

Finally, for reliability success other components in RO infrastructure are described to address this aspect for preservation. The reliability property, i.e. the capability of a Research Object to maintain its properties over time, is key to reuse it as the instrument for knowledge exchange. However, scientific workflows and resources are commonly subject to a decayed or reduced ability to be executed or repeated, due to the volatility of external datasets, sources, scripts, etc. Our approach must consider both these internal and external changes when helping the scientists to judge the reliability of the Research Object [2][3]. The components for ensure reliability issues are the next:

- Notification API[76]: The system will generate warning alerts whenever there are events related to Research Objects;
- RO Monitoring Tool [2]: This tool[77] provides an entire environment with functionalities for time-based computation of the completeness, stability and reliability scores of an RO via RESTful API[78].

It must also be highlighted that RODL[79] platform contains Reliability Notifications[80] services which are in charge of provide the user information about reliability, stability and completeness together with the set of rules of the checklist tool every time that the completeness of the Research Object changes (Figure 49).

[73] https://github.com/wf4ever/apis/wiki/Wf4Ever-Services-and-APIs#Stability
[74] https://github.com/wf4ever/apis/wiki/Stability-Evaluation-API
[75] https://github.com/wf4ever/apis/wiki/RO-checklist-evaluation-API
[76] https://github.com/wf4ever/apis/wiki/Notification-API
[77] http://sandbox.wf4ever-project.org/decayMonitoring/monitor.html
[78] http://sandbox.wf4ever-project.org/decayMonitoring/rest/getAnalytics
[79] http://www.rohub.org
[80] https://github.com/wf4ever/reliability



Workflow-centric RO with provenance requirements (y3demo)

Target HD data interpretation *does not satisfy* checklist for *complete.*

- ✓ Experiment hypothesis or research question is indicated
- ✗ Experiment description is not indicated
- ✓ Experiment design sketch is indicated
- ✓ Workflow description is present
- ✓ All referenced workflow definitions are accessible
- ✗ No workflow process software dependencies are indic
- ✓ All referenced workflow services are accessible
- ✗ No workflow sample run indicated for any workflow
- ✗ No input data indicated for any workflow
- ✓ Workflow results data run indicated
- ✓ Experiment conclusions are indicated

See quality history with RO Monitoring Tool

Notifications

20:12
The Research Object data_interpretation-2 has been commented by Raul Palma

12/5/2014
Oops! RO Reliability has Decreased...

7/5/2014
Good! RO Reliability has Increased!

6/5/2014

May 19, 2014 20:12
The Research Object data_interpretation-2 has been commented by Raul Palma

This experiment was used as the running example for the RO paper

Figure 49 ROHub quality check

ROHub backend is composed of access components, long-term preservation components and controller that manage the flow of data. ROs are stored in the access repository once created, and periodically the new and/or modified ROs are pushed to the long-term preservation repository.

The access components are the storage backend and the semantic metadata triplestore. The storage backend can be based on dLibra[81], which provides file storage and retrieval functionalities, including file versioning and consistency checking, or it can use a built-in module for storing ROs directly in the filesystem. The semantic metadata are additionally parsed and stored in a triplestore backed by Jena TDB[82].
The long-term preservation component is built on dArceo[83], which stores ROs, including resources and annotations. Objects in dArceo can be stored on a range of backends, including specialised preservation repositories such as the Platon[84] service, storing data in geographically distributed copies and guaranteeing their consistency. dArceo gives the possibility to define migration plans that allow to perform a batch update of resources from one format to another, when necessary. In case of workflows, this may be applied for instance when a flat Taverna[85] t2flow format should be converted to a complex scufl2 format (which, N.B. uses the Research Object model similarly to Research Objects). Another case could be a batch update of workflows that depend on a malfunctioning external resource.
Additionally, the preservation component of ROHub provides fixity checking and monitors the RO quality through time against a predefined set of requirements (with stability). If a change is detected, notifications are generated as Atom feeds.

In Wf4Ever project was made a further analysis of preservation activities of ROs according to OAIS[86] and microservices[87] paradigms [4].

[81] http://dingo.psnc.pl/dlibra/

[82] https://jena.apache.org/documentation/tdb/

[83] http://dingo.psnc.pl/darceo/

[84] http://www.platon.pionier.net.pl/

[85] http://www.taverna.org.uk/

[86] https://es.wikipedia.org/wiki/Open_Archival_Information_System

[87] http://microservices.io/

6.3.5 References

[1] De Roure, D., Belhajjame, K., Missier, P., Gómez-Pérez, J. M., Palma, R., Ruiz, J. E., ... & Mons, B. (2011, November). Towards the preservation of scientific workflows. In *Proc 8th International Conference on Preservation of Digital Objects (iPRES 2011)*.
[2] Gómez-Pérez, J. M., García-Cuesta, E., Garrido, A., Ruiz, J. E., Zhao, J., & Klyne, G. (2013). When history matters-assessing reliability for the reuse of scientific workflows. In *The Semantic Web–ISWC 2013* (pp. 81-97). Springer Berlin Heidelberg.
[3] Gómez-Pérez, J. M., García-Cuesta, E., Zhao, J., Garrido, A., Ruiz, J. E., & Klyne, G. (2013, May). How Reliable is Your Workflow: Monitoring Decay in Scholarly Publications. In SePublica (pp. 75-86).
[4] Pérez, S., Corcho, O., Palma, R., & Holubowicz, P. (2014). Best Practices for Archival Processing of Research Objects. http://es.slideshare.net/ocorcho/best-practices-for-archival-processing-of-research-objects-a-librarian-view

7 E-learning Services

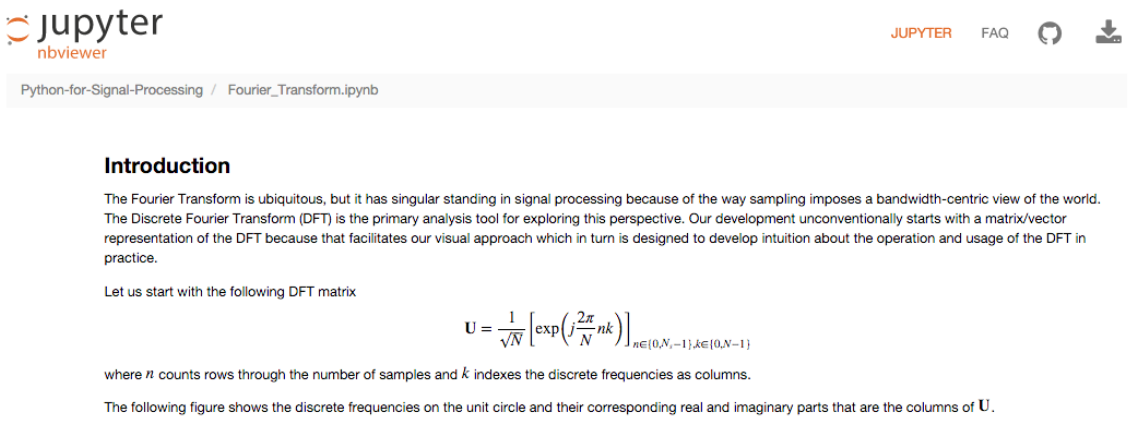
7.1 Overview

A widely accepted definition for the term e-learning is:

learning conducted via electronic media, typically on the Internet

The e-learning service relies on Jupyter notebooks: interactive data science and scientific computing across a high-number of programming languages in the form of executable documents organised in units covering EO data science computing techniques in several thematic areas.

To illustrate the concept, an example of a Jupyter notebook discussing the Fourier Transform is show under the form of a publicly available Jupyter notebook[88]. The first section introduces the Fourier Transform:



Introduction

The Fourier Transform is ubiquitous, but it has singular standing in signal processing because of the way sampling imposes a bandwidth-centric view of the world. The Discrete Fourier Transform (DFT) is the primary analysis tool for exploring this perspective. Our development unconventionally starts with a matrix/vector representation of the DFT because that facilitates our visual approach which in turn is designed to develop intuition about the operation and usage of the DFT in practice.

Let us start with the following DFT matrix

$$\mathbf{U} = \frac{1}{\sqrt{N}}\left[\exp\left(j\frac{2\pi}{N}nk\right)\right]_{n\in\{0,N_s-1\},k\in\{0,N-1\}}$$

where n counts rows through the number of samples and k indexes the discrete frequencies as columns.

The following figure shows the discrete frequencies on the unit circle and their corresponding real and imaginary parts that are the columns of $\mathbf{U}$.

Figure 50 Fourier Transform Notebook

The next notebook cell includes the code used to generate the plots below:

```
Out[1]: <matplotlib.text.Text at 0x5c7fb10>
```

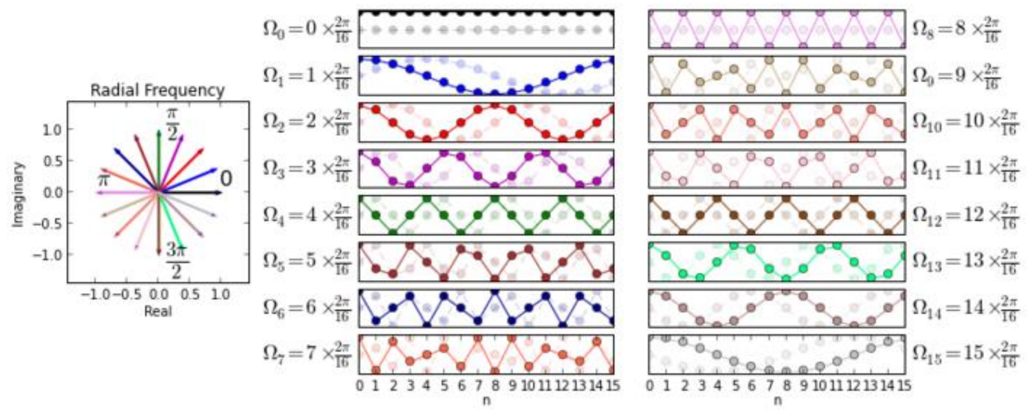
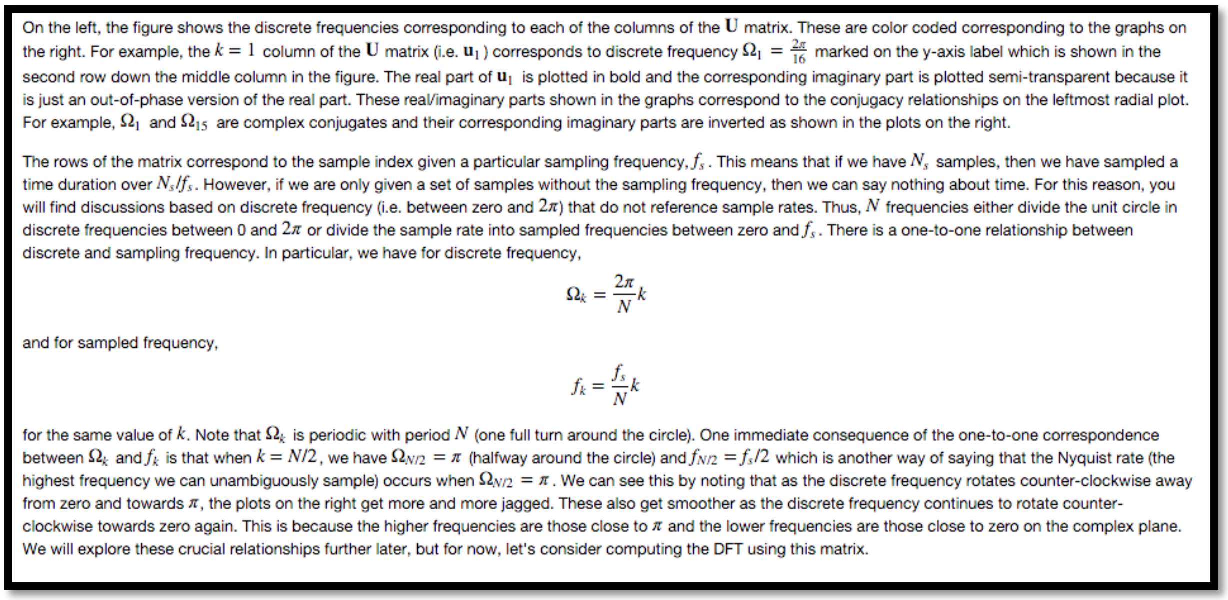


Figure 51 Fourier Transform Notebook output

[88] http://nbviewer.jupyter.org/github/unpingco/Python-for-Signal-Processing/blob/master/Fourier_Transform.ipynb

This output is discussed with the explanations provided as text that include mathematical formulas:



On the left, the figure shows the discrete frequencies corresponding to each of the columns of the $\mathbf{U}$ matrix. These are color coded corresponding to the graphs on the right. For example, the $k = 1$ column of the $\mathbf{U}$ matrix (i.e. $\mathbf{u}_1$) corresponds to discrete frequency $\Omega_1 = \frac{2\pi}{16}$ marked on the y-axis label which is shown in the second row down the middle column in the figure. The real part of $\mathbf{u}_1$ is plotted in bold and the corresponding imaginary part is plotted semi-transparent because it is just an out-of-phase version of the real part. These real/imaginary parts shown in the graphs correspond to the conjugacy relationships on the leftmost radial plot. For example, Ω_1 and Ω_{15} are complex conjugates and their corresponding imaginary parts are inverted as shown in the plots on the right.

The rows of the matrix correspond to the sample index given a particular sampling frequency, f_s. This means that if we have N_s samples, then we have sampled a time duration over N_s/f_s. However, if we are only given a set of samples without the sampling frequency, then we can say nothing about time. For this reason, you will find discussions based on discrete frequency (i.e. between zero and 2π) that do not reference sample rates. Thus, N frequencies either divide the unit circle in discrete frequencies between 0 and 2π or divide the sample rate into sampled frequencies between zero and f_s. There is a one-to-one relationship between discrete and sampling frequency. In particular, we have for discrete frequency,

$$\Omega_k = \frac{2\pi}{N}k$$

and for sampled frequency,

$$f_k = \frac{f_s}{N}k$$

for the same value of k. Note that Ω_k is periodic with period N (one full turn around the circle). One immediate consequence of the one-to-one correspondence between Ω_k and f_k is that when $k = N/2$, we have $\Omega_{N/2} = \pi$ (halfway around the circle) and $f_{N/2} = f_s/2$ which is another way of saying that the Nyquist rate (the highest frequency we can unambiguously sample) occurs when $\Omega_{N/2} = \pi$. We can see this by noting that as the discrete frequency rotates counter-clockwise away from zero and towards π, the plots on the right get more and more jagged. These also get smoother as the discrete frequency continues to rotate counter-clockwise towards zero again. This is because the higher frequencies are those close to π and the lower frequencies are those close to zero on the complex plane. We will explore these crucial relationships further later, but for now, let's consider computing the DFT using this matrix.

Figure 52 Fourier Transform learning material

This notebook example includes several other chunks of code, results and explanations. The point to agree on is that users willing to learn about the Fourier Transform can pick up the Jupyter notebook and manipulate it to autonomously strengthen their understanding of the mathematical concept being conveyed through the Jupyter notebook. The Fourier Transform example can be applied to a myriad of other areas of the Earth Science data processing.

The e-learning service thus delivers a web application that allow platform users to create and share documents that contain live code, equations, visualisations and explanatory text. The typical uses for such documents include: data cleaning and transformation, numerical simulation, statistical modelling, and machine learning.

The Jupyter Notebook is an interactive computing environment that enables users to author notebook documents that include: live code, interactive widgets, plots, narrative text, equations, images and video. The Jupyter Notebook provides a complete and self-contained record of a computation that can be converted to various formats and shared with others.

The Jupyter Notebook combines three components:

- The Jupyter *Notebook web application*: An interactive web application for writing and running code interactively and authoring notebook documents.
- The *Jupyter Kernels*: Separate processes started by the notebook web application that runs users' code in a given language and returns output back to the notebook web application. The kernel also handles things like computations for interactive widgets, tab completion and introspection.
- The *Jupyter Notebook documents*: Self-contained documents that contain a representation of all content visible in the notebook web application, including inputs and outputs of the computations, narrative text, equations, images, and rich media representations of objects. Each notebook document has its own kernel.

This section and its sub-sections do not enter into details about the actual content of the e-learning material (units covering thematic areas of EO data manipulation and processing) but rather describe how technically e-learning is supported by interactive computing that can lead users to learning basic and advanced lab skills for research computing.

7.2 Components

The following picture provides the architectural approach to the E-Learning module integration into the EVER-EST infrastructure.

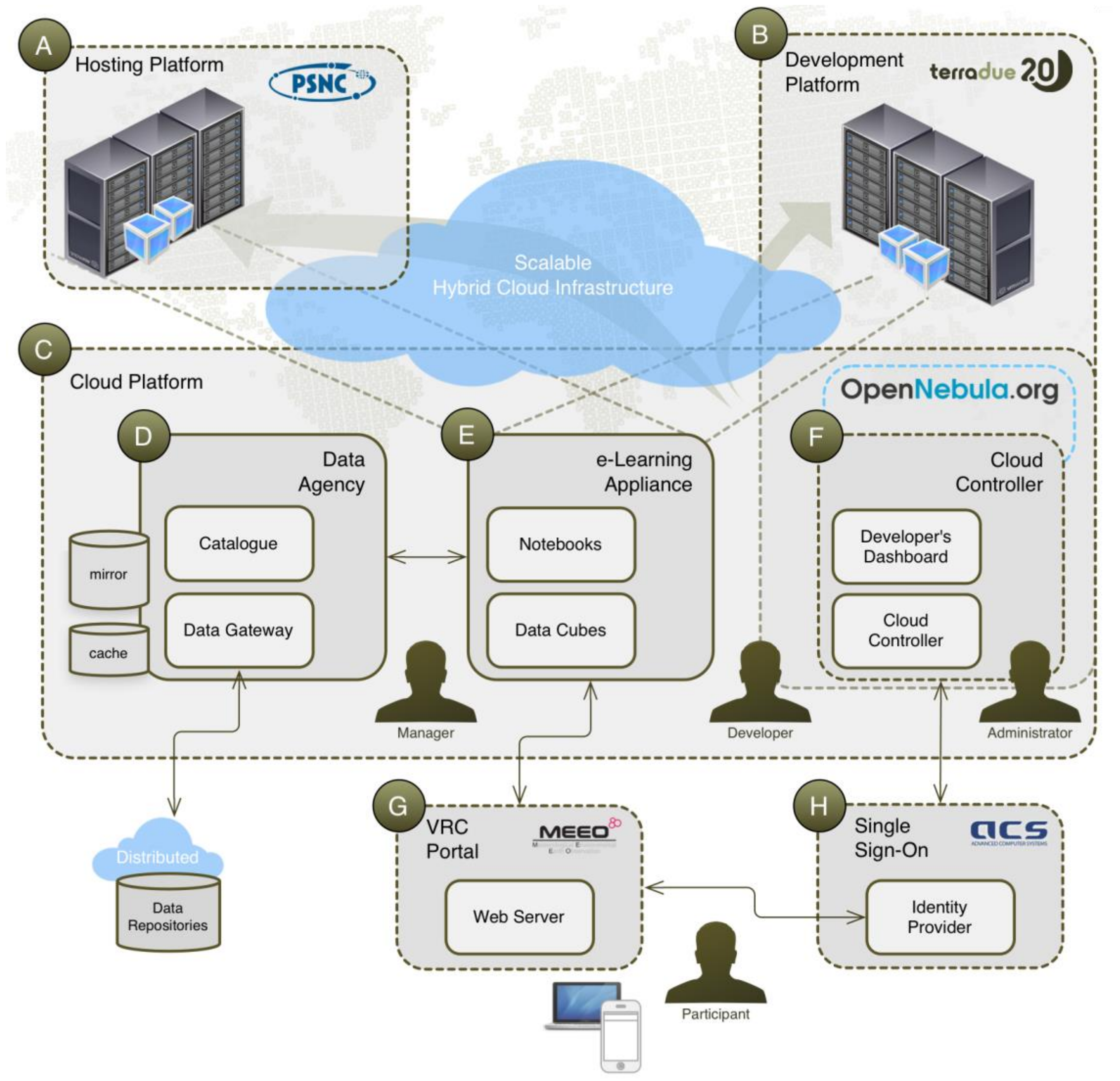
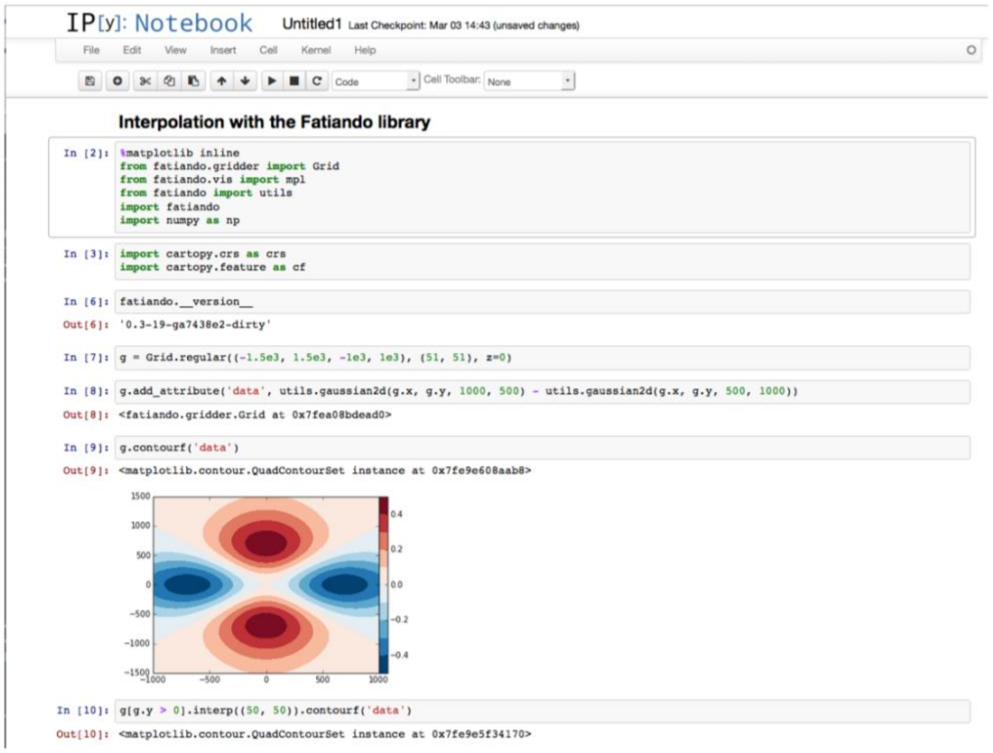


Figure 53 E-learning module and related components

7.2.1 Notebook web application

As introduced above, the notebook web application is an interactive web application for writing and running code interactively and authoring notebook documents. The figure below shows an example of a Jupyter notebook provided via the notebook web application. This example uses a Python library called Fatiando to perform a 2D Gaussian extrapolation. The results are shown in Figure 54 below the code cells.



Interpolation with the Fatiando library

```
In [2]: %matplotlib inline
        from fatiando.gridder import Grid
        from fatiando.vis import mpl
        from fatiando import utils
        import fatiando
        import numpy as np

In [3]: import cartopy.crs as crs
        import cartopy.feature as cf

In [6]: fatiando.__version__

Out[6]: '0.3-19-ga7438e2-dirty'

In [7]: g = Grid.regular((-1.5e3, 1.5e3, -1e3, 1e3), (51, 51), z=0)

In [8]: g.add_attribute('data', utils.gaussian2d(g.x, g.y, 1000, 500) - utils.gaussian2d(g.x, g.y, 500, 1000))

Out[8]: <fatiando.gridder.Grid at 0x7fea08bdead0>

In [9]: g.contourf('data')

Out[9]: <matplotlib.contour.QuadContourSet instance at 0x7fe9e608aab8>

In [10]: g[g.y > 0].interp((50, 50)).contourf('data')

Out[10]: <matplotlib.contour.QuadContourSet instance at 0x7fe9e5f34170>
```

Figure 54 Jupiter notebook results

The Jupyter notebook web application offer a powerful 'scratchpad' paradigm for the creation and management of live computational documents with rich media. Users can execute blocks of code (provided by a given kernel) in the browser with automatic syntax highlighting, indentation, and tab completion/introspection. The results of the computation are attached to the code that generated them as rich media representations, such as HTML, LaTeX, PNG, SVG, PDF, etc. Besides these rich media representations, users can create and use interactive JavaScript widgets, which bind interactive user interface controls and visualisations to reactive kernel side computations. Alongside with the code, users can keep notes and other text by changing the style of a Notebook cell from "Code" to "Markdown". The notes can be organised in a hierarchical structure with different levels of headings and authored as narrative text using the Markdown markup language.

7.2.2 Kernels

Through Jupyter's kernel and messaging architecture, the Jupyter notebook allows code to be run in a range of different programming languages. For each notebook document that a user opens, the web application starts a kernel that runs the code for that notebook. Kernels are programming language specific processes that run independently and interact with the Jupyter applications and their user interfaces. Each kernel is capable of running code in a single programming language and there are kernels available in several languages. The "Kernel Zero" is IPython and it comes as a dependency of Jupyter. The IPython kernel can be thought as a reference implementation. The number of kernels supported by Jupyter is growing, here are a few Julia, R, Ruby, Haskell, Scala, node.js and Go. The notebook provides a simple way for users to pick which of the kernels is used for a given notebook.

7.2.3 Jupyter notebook documents

As described in the previous sections, Jupyter notebook documents contain the inputs and outputs of an interactive session as well as narrative text that accompanies the code but is not meant for execution. Rich output generated by running code, including HTML, images, video, and plots, is embedded in the notebook, which makes it a complete and self-contained record of a computation. When the users run the Jupyter notebook web application, notebook documents are files on the local filesystem with a ".ipynb" extension. This allows users to use classical workflows for organising the Jupyter notebook documents into folders or remote repositories to allow sharing these with others. Internally, notebook documents are JSON data with binary values "base64". This allows the Jupyter notebook documents to be read and manipulated programmatically by any programming language. Because JSON is a text format, notebook documents are version control friendly.

Jupyter notebook documents consist of a linear sequence of cells. There are four basic cell types:

- *Code cells*: Input and output of live code that is run in the kernel;
- *Markdown cells*: Narrative text with embedded LaTeX equations;
- *Heading cells*: 6 levels of hierarchical organisation and formatting;
- *Raw cells*: Unformatted text that is included, without modification, when notebooks are converted to different formats using nbconvert.

Jupyter notebook documents can be exported to different static formats using Jupyter's nbconvert utility. This tool converts notebooks to various other formats via Jinja templates. The nbconvert tool allows users to convert a Jupyter notebook document into various static formats including HTML, LaTeX, PDF, Reveal JS, Markdown, ReStructured Text and executable script.

Furthermore, Jupyter notebook documents available from a public URL on or GitHub can be shared via nbviewer. This service loads the Jupyter notebook document and renders it as a static web page. The resulting web page may thus be shared with others without their needing to install the Jupyter Notebook.

7.3 Main Functionalities

7.3.1 Notebook provisioning

The Jupyter notebook web applications are provisioned in a multi-tenant environment and self-contained in Docker containers. The architectural component delivering this capacity is made of two components: the *JupyterHub*, a server that gives multiple users access to Jupyter notebooks, running an independent Jupyter notebook server for each user and the *spawners* that control how JupyterHub starts the individual notebook server for each user.

Users access JupyterHub via the web browser, as they would do with the Jupyter web application by going to the address of the JupyterHub server. Users authenticate using the defined authenticator (in our case a Single Sign-On) and trigger a new instance of a Jupyter server using the *spawner*. The approach followed uses the DockerSpawner to deploy Docker containers to provide resources to the Jupyter server. The DockerSpawner can provide two types of that spawners: *dockerspawner.DockerSpawner,* for spawning identical Docker containers for each user and *dockerspawner.SystemUserSpawner*, for spawning Docker containers with an environment and home directory for each user.

7.3.2 Authentication

Jupyter notebook servers require the authentication of the users to guarantee the privacy. The JupyterHub provides several ways for users to authenticate via the *authenticators*. The authentication layer is a flexible environment that support several implementations of an Authenticator which is the component delivering the mechanism for authorising users. There are several implementations of the Authenticator creating the link with technologies like LDAP and OAuth. Users will use the Single Sign-On element of the platform.

7.3.3 Data discovery and access

As explained above Jupyter notebook servers are deployed in Docker containers and the access to data also happens within the Docker container. In order to be able to save data and share data between Docker containers, Docker came up with the concept of Docker volumes. Quite simply, volumes are directories (or files) that are outside of the default Union File System and exist as normal directories and files on the host filesystem (the Union File System is combination of read-only layers with a read-write layer on top that is lost when the containers is dismissed).

The Jupyter notebook servers spawned in a Docker container will access data by mounting a Docker volume. The data available in the Docker volume is dictated by the data package that originated it. The definition of a data package relies on the data discovery mechanism offered by the platform typically by accessing OpenSearch catalogues featuring tens of data collections and providing advanced query mechanisms driven by the thematic facets of the data (e.g. interferometric search for SAR data or cloud coverage for optical data). Data packages contain references to one or more catalogue element entries. Once stored, the elements reference within a given data package are fetched from the archives (local or remote) and all together create a Docker volume that is mounted on the Docker container hosting the user notebook server. From the notebook - and user - perspective, access the data contained in the Docker volume is done as with a typical POSIX file system and thus providing high throughput.

7.3.4 Persistent storage

As introduced in the previous sections, the Jupyter notebooks contain live code that can be run over and over and the outcome of these executions can either be inline results (and thus contained in the notebook) or physical results that are written on the local filesystem. In the first case - the inline results - the persistence is guaranteed by Jupyter when the notebook is saved, in the second case, the persistence of the physical results produced must be addressed by other means. In a similar approach as for the data packages the persistence of the physical results is done by creating another docker volume that is associated to the notebook. This solution provides the possibility to share the docker volume as an input data package to another notebook that can be owned by another user within the platform.

7.3.5 Scaling a Jupyter notebook

Jupyter notebooks offer interactive processing of data via a Jupyter Web Application. While the vertical scaling - by nature limited by the capacity of the host - is a possibility to extend the processing capacity of the resources made available to a user, the horizontal scaling offers wider use cases to support the replication of the execution of the code in Jupyter notebook documents against large archives of Earth Science data.

This horizontal scaling of a Jupyter notebook is the process of translating the Jupyter notebook into an operational tool for large-scale and cost effective processing against large sets of data.

The horizontal scaling is done exploiting the Cloud framework offered by the platform where YARN plays a central role. In particular, the capacity of YARN to deploy Docker containers and the YARN capacity of supporting several computational models were explained in detail in section 6.1.4. A new and dedicated computational model will be adopted in YARN to offer the horizontal scaling and thus support the replication of the single Jupyter notebook processing a batch of input data in several tens, hundreds or even thousands of Docker containers each processing a subset of the input data.

8 KPI and Smart Objectives

8.1 Smart objectives

With respect to the Smart Objectives that were defined within the proposal, this document addresses the following:

SM_OB#1.1	**Deploy a VRE based on a Service Oriented Architecture (SOA) tailored for Earth Science**
Measured by	Adherence to SOA standards; services usability outside the VRE for custom clients.
Achievable	Take-up and adaptation of webservices components developed in previous projects and development of new ones.
Relevant	SOA architecture is crucial for a dynamic and scalable system.
Timely	VRE architecture and interfaces will be defined by M4 and deployed at M22.

Table 3 Smart Objectives 1.1

The SOA standards are reflected by the adoption of SOA principles and the use of a Enterprise Service Bus as widely described in section 4.2. Moreover: previous projects services, including Sandbox, Preservation Assistant, ROHub services, MEA services are adopted and integrated into the EVER-EST VRE as illustrated in the various sections of the document.

SM_OB#1.2	**Deploy services to discover, access & process heterogeneous Earth Science (ES) datasets.**
Measured by	VRCs capacity to discover, access and process ES datasets reported in WP3 deliverables.
Achievable	Query/Access/Process components for heterogeneous ES data available through EVER-EST building block components.
Relevant	Heterogeneous data query, access and processing is vital for the VRE adoption by ES VRCs.
Timely	Services deployed in their final version in M22.

Table 4 Smart Objectives 1.2

Graphical user interface for data discovery is described in Visual components for data paragraph. All technical solutions to allow the integration of data provider’s catalogues are listed in the paragraph on Data discovery and .

SM_OB#1.3	**Deploy services to capture and store research activities workflows, processes and results.**
Measured by	VRCs capability to manage the full research activity workflow lifecycle.
Achievable	Using the innovative Research Objects paradigm adapted to Earth Science.
Relevant	Research workflow lifecycle mgt. is key to create new knowledge re-using pre-existing research results and knowledge.
Timely	Research Object and ROHub finalised in M12; integrated in M22.

Table 5 Smart Objectives 1.3

These functionalities are ensured by the adoption of Research Objects technologies with particular emphasis on the ROHub. Research Objects and ROHub are described in Research objects and ROHub chapters.

SM_OB#1.4	**Deploy services to share data, models, algorithms, results within and across communities.**
Measured by	Capacity to share models, data and knowledge across communities measured by feedback received from VRCs and reported in WP3 deliverables.
Achievable	Research Objects and Preservation services to allow resource sharing).
Relevant	The improvement of data and knowledge sharing capabilities is the core objective of the EVER-EST VRE concept and design.
Timely	Research Object study finalised in M12; integrated in VRE final version in M22.

Table 6 Smart Objectives 1.4

These functionalities are ensured by the adoption of Research Objects technologies with particular emphasis on the ROHub. Research Objects and ROHub are described in Research objects and ROHub chapters.

SM_OB#1.5	**Ensure the long-term preservation of resources developed by existing VRC communities.**
Measured by	Preservation registries for data-related knowledge deployed and populated.
Achievable	Leverage on SCIDIP-ES project results to adapt preservation services in the VRE.
Relevant	Data related knowledge is key for data usability by heterogeneous user communities and for interoperability among them.
Timely	Preservation services integrated in their final version in M22.

Table 7 Smart Objectives 1.5

Features for Data and RO preservation are provided within the architectural design. Full description on data preservation is available in Chapter Preservation

SM_OB#1.6	**Innovative and user friendly environment for real-time collaborative working.**
Measured by	Web portal deployment and VRCs usage statistics generated through the web portal.
Achievable	Based on Multi-sensor Evolution Analysis (MEA) technology.
Relevant	Web portal for accessing VRE functionalities will facilitate communities’ engagement.
Timely	EVER-EST web portal for cooperative working available and integrated in M22.

Table 8 Smart Objectives 1.6

The adoption of CMS and the use of SoA COTS for collaborative environment addresses this Smart Objective as described in chapter 5.

The technology described in the current document is also the basis to satisfy SM_OB#3.1, SM_OB#3.2, SM_OB#3.3 that will be validated in future project activities.

8.2 Key performance indicators

This document represents a fundamental deliverable within the EVER-EST project. All findings and results of the activities will leverage on the work carried out on the basis of the Architectural Document. It is therefore behind all the KPIs that refer to establishment of the VRE for Earth Science (KPI1, KPI2, KPI3, KPI4, KPI5) and those Key Performance Indicators referring to the implementation and validation of the Research Object concept in Earth Science (KPI6, KPI7, KPI8).