Climate Change Initiative Extension (CCI+) Phase 1

New Essential Climate Variables (NEW ECVS)

High Resolution Land Cover ECV (HR_LandCover_cci)

System Specification Document

(SSD)

Prepared by:

Università degli Studi di Trento
Fondazione Bruno Kessler
Université Catholique de Louvain
Università degli Studi di Pavia
Università degli Studi di Genova
Politecnico di Milano
Université de Versailles Saint Quentin
CREAF
e-GEOS s.p.a.
Planetek Italia
GeoVille

Changelog

Issue	Changes	Date
1.0	First version.	16/04/2020
1.1	Revision according to "CCI_HRLC_Ph1_Milestone5_RID-ESA.v1.xlsx". Included description of opt-spectral-adaptation in paragraph 3.3.2.1	04/05/2020

Detailed Change Record

Issue	RID	Description of discrepancy	Sections	Change
1.1	FR-01	Please be informed that the there are difference related to DEM used between the products generated from PDGS (PlanetDEM) and ones from Sen2Cor (SRTM) as stand alone. Further information are available at http://step.esa.int/main/third-party-plugins-2/sen2cor/sen2cor_v2-8/	3.3.2.1 Pre-processing Optical	A note has been added as a reminder in the processor description and ancillary data description.

Contents

1 Introduction 4

1.1 Executive summary 4

1.2 Purpose and scope 4

1.3 Applicable documents 4

1.4 Reference documents 5

1.5 Acronyms and abbreviations 5

2 Platform prototype for HRLC production 7

2.1 Main challenges from SRD 7

2.1.1 Orchestration of resources and IaaS 7

2.1.2 Flexibility in the pipeline development 7

2.1.3 Delivery of results 8

2.2 High level architecture from SRD 8

2.3 Selection of the IaaS infrastructure 9

3 System Specification 10

3.1 Solution concept 11

3.2 Platform architecture 14

3.2.1 Enterprise Viewpoint 15

3.2.1.1 User of final products role 16

3.2.1.2 User of processing service role 17

3.2.1.3 Expert user role 18

3.2.2 Information Viewpoint 18

3.2.3 Computational Viewpoint 19

3.2.4 Engineering Viewpoint 21

3.2.5 Tecnology Viewpoint 23

3.3 Pipelines architecture 24

3.3.1 Product Summary 24

3.3.2 Static Map Chain 24

3.3.2.1 Pre-processing Optical 25

3.3.2.2 Pre-processing SAR 30

3.3.2.3 Multi-sensor Geo-location 34

3.3.2.4 Classification Optical 36

3.3.2.5 Classification SAR 37

3.3.2.6 Decision Fusion 43

3.3.3 Dynamic Map Chain – Classification/Change Map 46

3.3.3.1 Optical/SAR Feature Extraction 46

3.3.3.2 Optical/SAR Time Series Regularization 48

3.3.3.3 Optical/SAR Changes and Trends 49

4 Annex 1 - Tender Requirements traceability 52

5 Annex 2 - Questionnaire to DIAS providers 54

1 Introduction

1.1 Executive summary

Following the activities of user requirements updating according to Climate User Community and other users' consultations, the Consortium has defined the related HRLC products requirements accounting for technical constraints such as main data sources available, spatial and temporal coverage, software and tools for quality control.

This High Resolution Land Cover (HRLC) System Specification Document defines the system architecture and the description of the first version of the system that will generate the CCI+ HRLC products over the areas of interests.

The concept of batch processing is at the basis of the design of the platform, this because the use case of the project is exactly to generate a large amount of products with loose strict time constraints.

- The first part ends describing the concept of the system the available API and gives details of the technological stack used to provide the processing.
- The second part describes the on-boarding mechanism for the processors and the description of the pipeline/processors.

For what concerns the IaaS environment of the platform, DIAS and commercial services like AWS and Google have been considered as requested by ESA. Finally, due to the possibility to use spare resources (AWS Spot Resources), the processing will take place on AWS. Anyway, the system already support Mundi DIAS (and is compatible with all other DIAS). In the second production the IaaS environment will be evaluated in terms of costs to verify the possibility to run the production on a DIAS.

1.2 Purpose and scope

The HRLC System Specification Document defines the requirements for an operational system for production of HRLC maps over the three areas selected in this phase.

Input to this document are the Tender Specification [AD2] and the other Applicable Documents.

- To generate the products specified in the HRLC Product Specification Document (PSD) using data specified in the Data Access Requirement Document (DARD);
- To apply the algorithms specified in the HRLC Algorithm Theoretical Basis Documents (ATBD) for this purpose (not yet finalized and with pending decision at the time of writing);
- To make the product accessible to users as specified in the HRLC User Requirements Document (URD).
- To comply with the System Requirement Document (SRD) even if some minor modification will be foreseen and traced also in the requirement document

1.3 Applicable documents

Ref.	**Title, Issue/Rev, Date, ID**
[AD1]	CCI HR Technical Proposal, v1.1, 16/03/2018
[AD2]	CCI Extension (CCI+) Phase 1 – New ECVs – Statement of Work, v1.3, 22/08/2017, ESA-CCI-PRGM-EOPS-SW-17-0032
[AD3]	Data Standards Requirements for CCI Data Producers, v2.1, 02/08/2019, CCI-PRGM-EOPS-TN-13-0009
[AD4]	User Requirements Document, v1.1, 12/04/2019, CCI_HRLC_Ph1-D1.1_URD
[AD5]	Product Specification Document, v1.0, CCI_HRLC_Ph1-PSD
[AD6]	Data Access Requirement Document v1.0, CCI_HRLC_Ph1-DARD
[AD7]	System Requirement Document v2.0, CCI_HRLC_Ph1-SRD

1.4 Reference documents

Ref.	Title, Issue/Rev, Date, ID
[RD1]	The Global Climate Observing System: Implementation Needs, 01/10/2016, GCOS-200

1.5 Acronyms and abbreviations

API	Application Programming Interface
AOI	Area Of Interest
ARD	Analysis Ready Data
AWS	Amazon Web Services
CCI	Climate Change Initiative
CRC	Climate Research Community
CMUG	Climate Modelling User Group
DIAS	Data and Information Access Services
ECV	Essential Climate Variables
ESM	Earth System Models
EVI	Enhanced Vegetation Index
FTP	File Transfer Protocol
GCOS	Global Climate Observing System
GDPR	General Data Protection Regulation
GIS	Geographical Information System
HR	High Resolution
IaaS	Infrastructure as a Service
L1C	Level-1C
L2A	Level-2A
LAI	Leaf Area Index
LaSRC	Landsat Surface Reflectance Code
LC	Land Cover
LCC	Land Cover Change
LCCS	Land Cover Coverage Classification System
LCML	Land Cover Meta Language
LCZ	Local Climate Zone
LEDAPS	Landsat Ecosystem Disturbance Adaptive Processing System
LSCE	Laboratoire des Sciences du Climat et de l'Environnement
MR	Medium Resolution
NDVI	Normalized Difference Vegetation Index
OGC	Open Geospatial Consortium
OWS	OGC Web Services
PFT	Plant Functional Type
RS	Remote Sensing

SAR	Synthetic Aperture Radar
SFT	Surface Functional Type
TOA	Top Of Atmosphere
URD	User Requirements Document
VM	Virtual meeting
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WP	Work Package

2 Platform prototype for HRLC production

2.1 Main challenges from SRD

The platform prototype scope is to bring the results of the research activities to a pre-operational level by scaling up the processing capacity in order to allow the production of massive land cover mosaics following GCOS requirements. In practice, having to deal with several TB of data (hundreds) means that the concept of pre-operational is not applicable and that the system must provide a huge capability to scale the processing. For this reason, some constrains and requirements coming from the SRD are dealing with the capability of the IaaS platform chosen for the execution of the production.

2.1.1 Orchestration of resources and IaaS

Due to the importance to reduce as much as possible the cost of the production, several IaaS have been evaluated with the following summary conclusions:

- Most of the DIAS platform are technically fit to the purpose considering they provide easy to use VMs with EC2 API, Storage with Object Storage API and access to Copernicus Sentinel data
- AWS (Amazon Web Services) and Google offer similar services and are hosting Copernicus Sentinel data too (hosted in EU data centers)
- AWS/Google provides also access to cheaper resources (e.g. AWS Spot Instances) that can save up to 70% of computing costs

The orchestration platform, described in the following paragraphs, is based on the concept of big-data processing and reactive pipeline execution (Reactive Manifesto, https://www.reactivemanifesto.org/en) and is based on Max-ICS technology (Max-ICS by Earthlab Louxembourg, http://www-max-ics.earthlab.lu/) which is part of e-GEOS CLEOS platform (currently used internally but soon to be released as a service).

In this sense, the choice of the platform used to develop the processing pipelines has taken into account the importance of supporting different environments on the long term. Max-ICS is currently supporting AWS, OpenTelekom Cloud (Mundi DIAS), Microsoft Azure and local HW. Moreover, it is deployable on all other DIAS platforms. The objective is to evaluate the overall costs of the production for this first year in order to consolidate the numbers and evaluate the possibility to run the final production on one of the DIAS.

2.1.2 Flexibility in the pipeline development

Another important point is the capability of the platform to manage easily changes in the pipeline with minimal manual intervention. The following scheme shows the main process which will be supported by the platform prototype development and the organization within the consortium. In particular, the prototype platform will put the basis for future enhancement by allowing easy link between the research activity and the production activity.

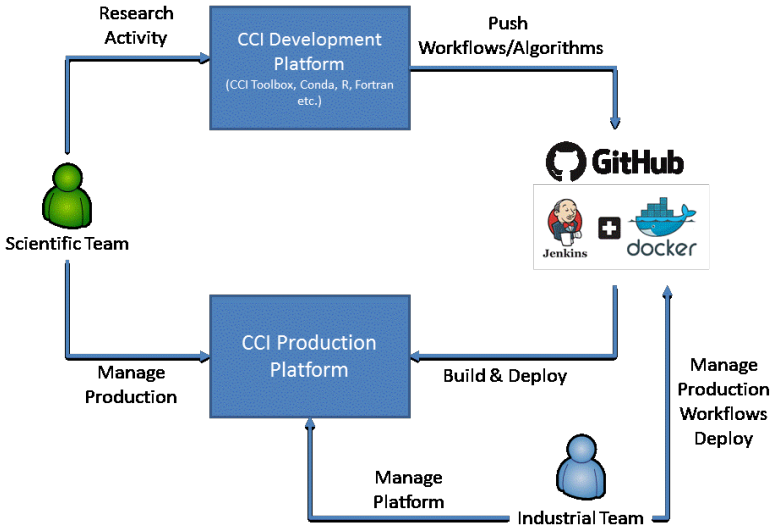


Figure 1. CCI Development/Production Platform Concept

The figure highlights the overall process starting from the research activity using Conda environment (https://conda.io/) and providing the code (and its updates) on the internal GitLab. Then, each processors generates a Docker image that is used by the orchestrator to run the process.

2.1.3 Delivery of results

In addition, a general OGC server is added to the processing platform that is used for the access to the large amount of data. The server allows to search for data collections using OGC CSW interface and to visualize/download data (e.g. HRLC products) using OGC WMS/WCS services.

Another important

2.2 High level architecture from SRD

The following figure shows a detail of the architecture of the platform as it was proposed during the proposal phase:

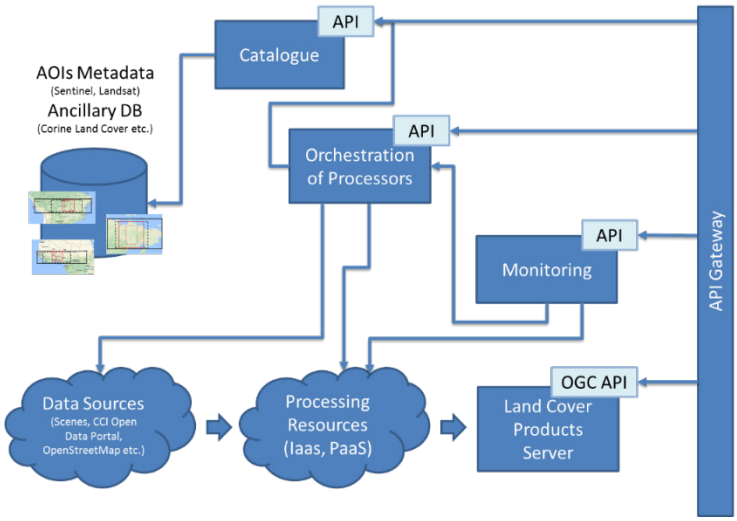


Figure 2. Detail on the architecture for the orchestration of massive processing (from the SRD)

The internal elements of the architecture are:

- **Metadata and Ancillary DB:** which is the store of metadata information (scene boundaries, cloud cover etc.) and of the ancillary data used by the algorithms. The **Catalogue** API allows to search and download the data using standards like OpenSearch
- **Catalogue**: which is the central point for the discovery of all data hosted and referenced in the platform. For example, in the case of imagery (Landsat, Sentinel), it will be possible to do various search and optimization before accessing the **Data Sources** remotely hosted (e.g. in the original archive such as ESA Sci-Hub, DIAS archives or public cloud resources)
- **Orchestration**: is the central element of the architecture. It handles the requests for processing coming from the API and dispatch the processing jobs by retrieving necessary **Data Sources** and allocating the **Processing Resources** to manage (e.g. execute/modify) the workflow used in the generation the final Land Cover Products. The **Orchestration** has also the objective of optimizing the **Processing Resources** in terms of utilization and of maintaining the necessary information (like steps, version control of algorithms/workflows, input/output data) for rerun of the processing.
- **Monitoring**: is the element devoted to check the status of the processing through the access to the **Orchestration** API in order to provide notification in case of anomalies, errors or even in case of job completion. The Monitoring element also interfaces the Processing Resources to check that the used IaaS/PaaS resources match with the expected
- **HR Land Cover Products Server**: is the element that hosts the final products in an optimized format published as OGC service like WMS and WCS. This server will also provide the necessary functions for the analytical Web Interface
- **Web Interface**: will provide a mapping interface capable of interacting with the WMS server and configuring maps and mashups
- **API Gateway**: is the element used to collect the API in a single reference point to gather metrics of the usage of the platform and to manage the access, authentication and authorization functionalities

In addition to the internal elements of the architecture, the external resources are:

- **Data Sources:** all the external available data sources like:
  - **Imagery Data Sources**: Landsat 5/7/8 Atmospherically Corrected retrieved from USGS from 1990 to 2019, Sentinel 1 (GRD) and Sentinel 2 (L2A) for 2019
  - **Reference Data Sources:** No reference data sources are required up to now, in any case OpenStreetMap DB is available as a service
- **Processing Resources:** the first production will be done on Amazon Web Services using Spot resources

During the project early phases, **Docker** has been agreed to be used mainly to guarantee that pipelines/tasks creation and execution can be managed without loss of time in the installation and compatibility checks.

2.3 Selection of the IaaS infrastructure

The selection of the IaaS infrastructure has been carried out by making a technical and economical comparison on the basis of questionnaire submitted to the DIAS services (in June 2019, see Annex 2). The response from DIAS is summarized in the tables below. The criteria for the selection where defined in SRD and basically, from the technical point of view, all DIAS and commercial providers such as AWS (Amazon Web Services) and GCE (Google Compute Engine) comply with the minimal specifications.

	CREODIAS	Mundi	ONDA	WEkEO	Sobloo
Sentinel 2	L1C full archive L2A Orderable (also non-ESA) rolling archive 1PB	L1C: last 12 months L2A: last 48 months (only Europe data)	L1C: full ESA archive L2A: full ESA archive	L1C: available L2A: not available	L1C: last two years L2A: orderable
Sentinel 1	SLC orderable, GRD full archive	SLC: last 12 months	full archive for the requested period (excluding NRT products)	Available	Available
Landsat 8	Only Europe	L1GT & L1TP last 48 months (only Europe data)	Available since 04/2018 (according to website only for Europe!)	Not available	Not included in catalogue, on-demand possible
Landsat 7	Only Europe	Not available (According to Website full archive available on-demand only)	Not available	Not available	Not included in catalogue, on-demand possible
Landsat 5	Only Europe	Not available	Not available	Not available	Not included in catalogue, on-demand possible
Missing data retrieval	Ordering mechanism available	Missing L2A can be retrieved from ESA or processed if not available	Missing data can be Retrieved and hosted in native format. Storage to be paid by user	Not available	On-demand possible

Figure 3: Data offer comparison

	CREODIAS	Mundi	ONDA	Sobloo
Availability of OpenStack API	Openstack API	Openstack API	Openstack API	Different API Available
VM with GPU	Bare Metal with GPU	Best quality GPU	Tailored GPU Machines	Tailored GPU Machines
Availability of Cold Object Storage	Only Hot Storage	Available	Not Clear / Information Not Provided. OVH provides this capability	Available
Price (estimated on S1/S2 pre-processing)	S1 pre-processing MEDIUM	S1 pre-processing MEDIUM	S1 pre-processing LOW	S1 pre-processing MEDIUM
	S2 pre-processing MEDIUM	S1 pre-processing MEDIUM	S1 pre-processing LOW	S1 pre-processing MEDIUM
	Object Storage MEDIUM	Object Storage LOW	Object Storage LOW	Object Storage MEDIUM
Cost report	Information not provided	Available endo of 2019	Available using an API	Available the day after

Figure 3: Computing offer comparison

Finally, among DIAS, ONDA DIAS is the best choice for the purpose of CCI HRLC for the following reasons:

- Lower Prices
- Full availability of Sentinel 2 and Sentinel 1 worldwide
- Availability of cold storage option – ONDA cloud archive

Unfortunately, none of the DIAS offers Spot-like[1] resources like Amazon or Google which reduces the price of computing up to 30% of the overall cost. This finally has been the main difference that led to the decision to perform at least this first year production on AWS. As anticipated, the underlying technology chosen as PaaS (Max-ICS) is already able to provide multi-cloud support. So future deployment on DIAS of the pipeline are feasible with minor effort. The use of DIAS will be re-evaluated next year after also having performed exact measurements of consumption of cloud resources.

3 System Specification

The System Specification is organised in three main paragraphs:

[1] Spot Like resources, User Guide: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html

- Solution concept: explain the concept behind the development of an architecture for the implementation of big-data processing pipelines
- Platform architecture: presents the architecture using the standard RM-ODP (Reference Model for Open Distributed Processing)
- Pipeline architecture: presents the engineering of the big-data pipelines as are going to be deployed on the platform

3.1 Solution concept

In this paragraph, we will introduce the concept of a batch pipeline for data processing and we will highlight the methodology used to meet the challenges described in paragraphs 2.1.1 and 2.1.2.

A generic architecture dedicated to Batch Pipeline is normally dealing with non-real-time treatments of data. This architecture pattern contains two main highlights. The first is the capacity to differentiate structured data, marked as ready to process, and non-structured data requiring a first parsing. Both are stored in Distributed File System (like NFS, HDFS or S3). The second highlight is the preparation of a serving layer with batched workers treating the all data stored in the DFS to populate the Serving Layer. In this kind of architecture, the Client application is totally de-correlated from the data assimilation constraints. This is an advantage in term of performance but is not applicable for Real-Time business operations. Any data entering the architecture will need to wait for the batch worker to integrate it.

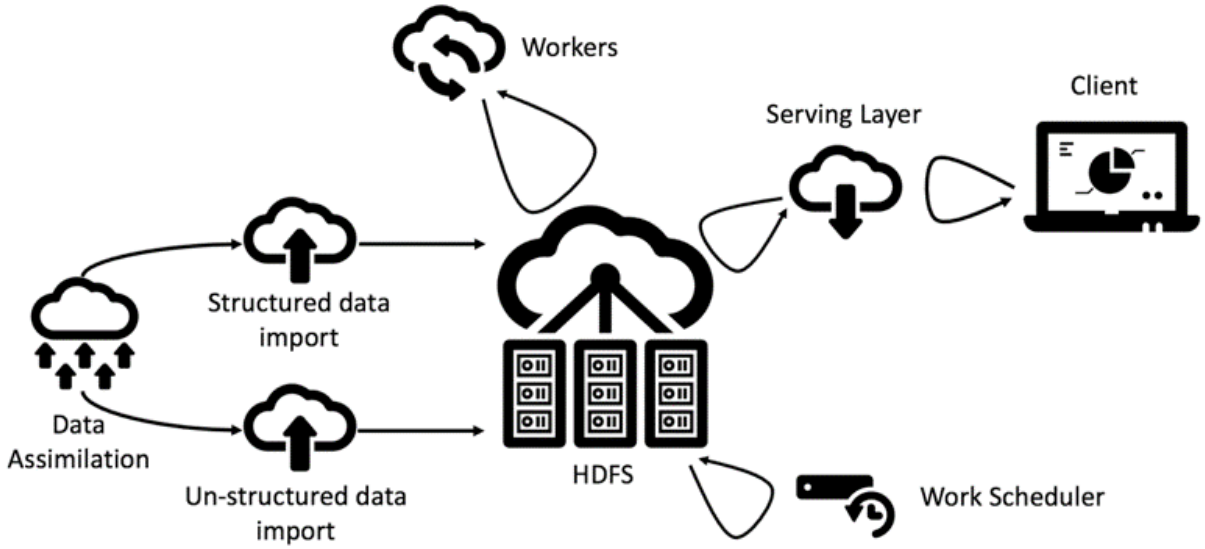


Figure 4: Example of Batch pipeline architecture showing the main concepts of 1) Work scheduler & Workers 2) Incoming data 3) Client interacting with the serving layer which is de-correlated from the processing back-end

The figure above shows an example of architecture of a platform for execution of batch pipeline. It is possible to map these three concepts to our case of EO big-data processing:

- Work scheduler/orchstrator & Workers: is the orchestrator of the steps which handle both the resources dedicated to each step and the dependency among steps. Workers are single nodes/steps that are single processors (described in paragraph 3.3) having the capability to transform a specific input A into an output B.
- Data: is the data selected as input to the pipeline (e.g. Sentinel 1 and 2) and that is treated by nodes. Each node has its own input data (e.g. ancillary data, weights, intermediate outputs from previous steps).
- Client: interacts with the serving layer which is de-correlated from the processing back-end

Coming to the platform solution, the following figure depicts the way in which batch processing is handled to meet both the orchestration of resources and the flexibility in pipeline development. In particular, for what concern the work scheduler/orchestrator the approach is depicted in the following picture.

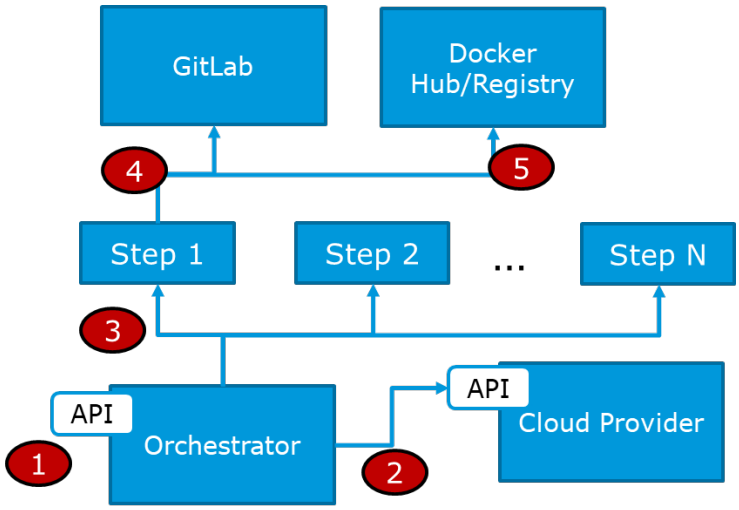


Figure 5: Main process of orchestration of a pipeline that includes the creation of workers by the provisioning of resources and configuration of the step

The events shown in the figure are the following:

1. API Processing request: an API processing call (from a Client) requests a processing like a classification over an area of interest and provides all necessary inputs (input data, parameters)
2. Resource Provision (Step by Step): the orchestrator is responsible for the provisioning of the resources from the cloud provider for the Step 1
3. Start of the Step 1 run: the orchestrator provides all necessary inputs to the provisioned resource to run the step
4. Step creation: each step is created by a GIT Pull (of code to be run) like for example a Dockerfile
5. Step execution: the run consists in a Docker Pull of Image & execution with the provided inputs (data & parameters)

The other important piece of a batch pipeline is the data handling, i.e. how the data is prepared for the run of a step considering that each step has its own peculiarities. The following picture shows the mechanism based on object storage repository:

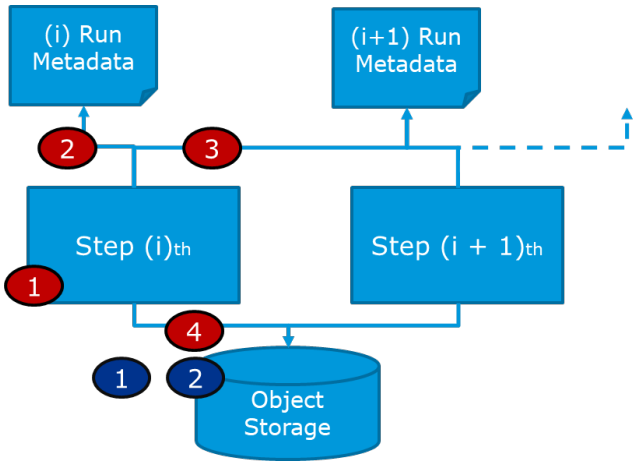


Figure 6: Main process of pipeline data handling and preparation for each step

The figure above summarizes how the data flows through the pipeline in terms of metadata (which lately are also named as messages) and persistent data saved in the object storage. Here are the events depicted in red:

1. *i-Step* is created: like described in previous picture each step is created and configured on a resource provisioned by the orchestrator
2. *i-Step* read the *i-Step* «run metadata»: the step receives a metadata with all the information needed to run. These can be as an example:
    - Parameters: run parameter of an algorithm defined by the Scientist or by the Engineering team
    - Data: some small data can be also sent in a metadata (e.g. an Area of Interest, Time period)
    - Metadata: metadata of the data to be used for the processing step that can be an URI to resources on an object storage (e.g. s3://<path-to-landsat-image>) or any mean to uniquely identify a resource
3. *i-Step* runs and saves additional info (if needed) into *(i+1)-Step* Metadata: additional information can be sent to the following steps like for example the metadata of new data (e.g. a NDVI coverage, a filtered SAR image)
4. *i-Step* runs and saves Output data into the Object storage: additional data processed are saved on the object storage so that the resource used to execute the step can be destroyed by the orchestrator

Then the orchestrator triggers the *(i+1)-Step* and proceed with the pipeline execution. Regarding the persistency of the data, it is important to underline that (blue events):

1. At each step, output files are created as artifacts on the object storage with folder structure reflecting the step execution
2. Logs (Container Logs) are also archived on the object storage and the monitoring system is taking into account of each step status (e.g. created, running, failed, completed) and of the pointers to outputs of the processing step

The Client of the platform interacts with the serving layer that is basically an API gateway forwarding API calls to the back-end according to user policies (authorization) and back-end capabilities. So, a user requesting for a processing will address the part of the API addressing batch pipeline execution and will be checked against the policies enforced by the authorization mechanism. At the same time, a user requesting the download of a product will be redirected to the internal service managing the download (e.g. HTTP simple download).

The interaction with the API triggering the execution of batch pipeline is normally asynchronous as batch processing is commonly used for long running data processing spanning also days.

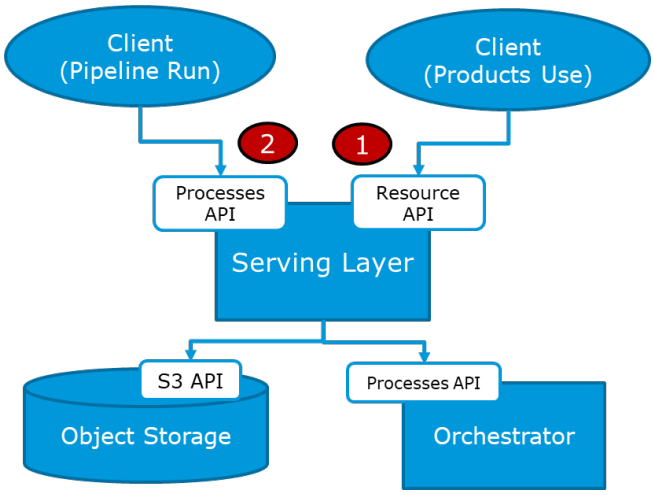


Figure 7: Client interaction with the platform

The figure above shows the architecture of the serving layer that is forwarding the requests to back-ends services. In particular:

1. Resource API exposes the capabilities to give access to data, for example using simple protocols like HTTP/FTP or standard services like OGC WMS/WCS
2. Processes API exposes the capability to execute batch data pipeline on the back-end

3.2 Platform architecture

The paragraph presents the architecture specification based on the RM-ODP methodology. RM-ODP is widely used to describe processing systems and is adopted by large number of architects that work on Geospatial Platforms. RM-ODP offers a conceptual framework and an architecture that integrates aspects related to the distribution, interoperation and portability of software systems, in such way that hardware heterogeneity, operating systems, networks, programming languages, databases and management systems are transparent to the user. In this sense, RM-ODP manages complexity through a “separation of concerns”, addressing specific problems from different points of view. The following figure shows the viewpoints with their fundamental objects:

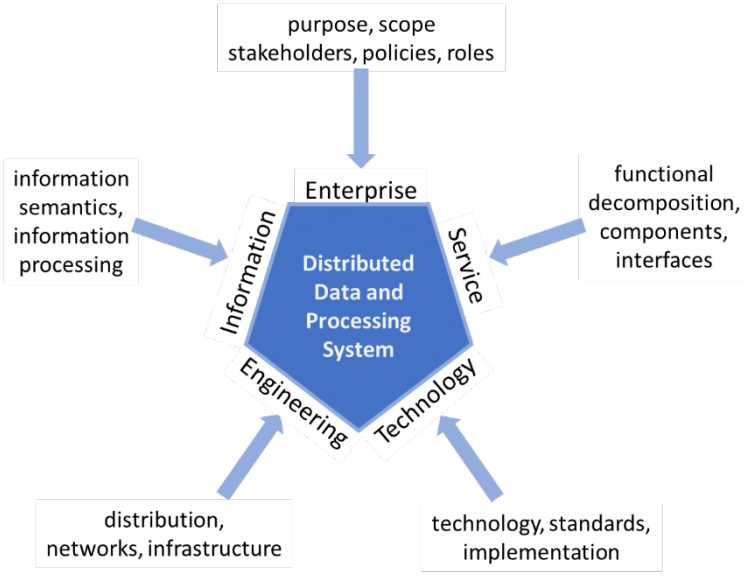


Figure 8: RM-ODP model for the description of systems (in new RM-ODP versions, computational viewpoint is often related as Service viewpoint)

The enterprise viewpoint represents the business model and the business requirements. This view should be understandable by all stakeholders in the business environment.

In RM-ODP, "purpose and objectives" concept is used to capture the reason for the system. It defines a set of objects formed to meet an objective, their activities, and processes in which the system participates. For example, the purpose and objective of this platform is:

"To develop a system capable of processing large amount of EO data through a batch data pipeline to generate classification products following a set of steps"

"To favour the flexibility to change the pipeline according to the need to improve the result of the classification"

"To visualise and access to the classification products"

The information viewpoint is concerned with the semantics of information and information processing. The information specification in this sense is made of the data and metadata flowing in the pipeline and in the interfaces provided by the internal and external API to the platform like OGC WMS/WCS for example.

The computational viewpoint is concerned with the interaction patterns between the components (services) of the platform, described through their interfaces. A computational specification of a service is a model of the service interfaces seen from a client, and the potential set of other services required by that service. The computational model defines types of interfaces such as request/reply or publish/subscribe or whether an interface is designed for exchange of real time or historical data or both. In this platform, the computational viewpoint identify the main component of the platform that are Orchestration component based on Max-ICS (orchestration, pipeline design, API development) and the OGC Service component based on Geoserver (OGC WMS/WCS/CSW).

The engineering viewpoint is concerned with the design of distributed systems. This means how the component handles the distribution of the processing. In the context of this platform, the distribution is handled by Max-ICS for what concerns the batch data pipeline execution and by a scalable deployment of Geoserver for what concerns the OGC Service component. The detail on how the distribution is done is given considering the challenges described in paragraph 2.1.

The technology viewpoint is concerned with the provision of an underlying infrastructure. It focuses on the technologies and the products for implementation. In this case, a brief description of both Max-ICS and Geoserver based architecture is given with pointers to further on-line resources. Moreover, the final AWS deployment is described.

3.2.1 Enterprise Viewpoint

The Enterprise Viewpoint is a look from the user perspective to the system so the starting point is to define the four users' roles with their main actions of the system that are:

1. **User of final products role** - it is the user that access the delivery services to visualise the products and access to them with available interfaces, the use cases are:
    - User discovers and visualises products
    - User analyses value-added product
2. **User of processing service role** - it is the user that access the available interfaces (API/CLI) to execute a processing service, the use case is:
    - Consumer executes a batch pipeline
3. **Expert user role** - it is the user that access to the available interfaces (API/UI) to deploy a new processing service and create a pipeline using those available:
    - Expert user builds a new processing service
    - Expert user builds a new batch pipeline by chaining different processing services

4. **Administrative user role** - it is the user that access to the monitoring facilities of the platform to troubleshoot problems. It basically use the available tool of the chosen technological tools.

The enterprise view of the system is represented in the following diagram using a simplified version with the main objective of keeping the description easy to be understood also by non-specialists.

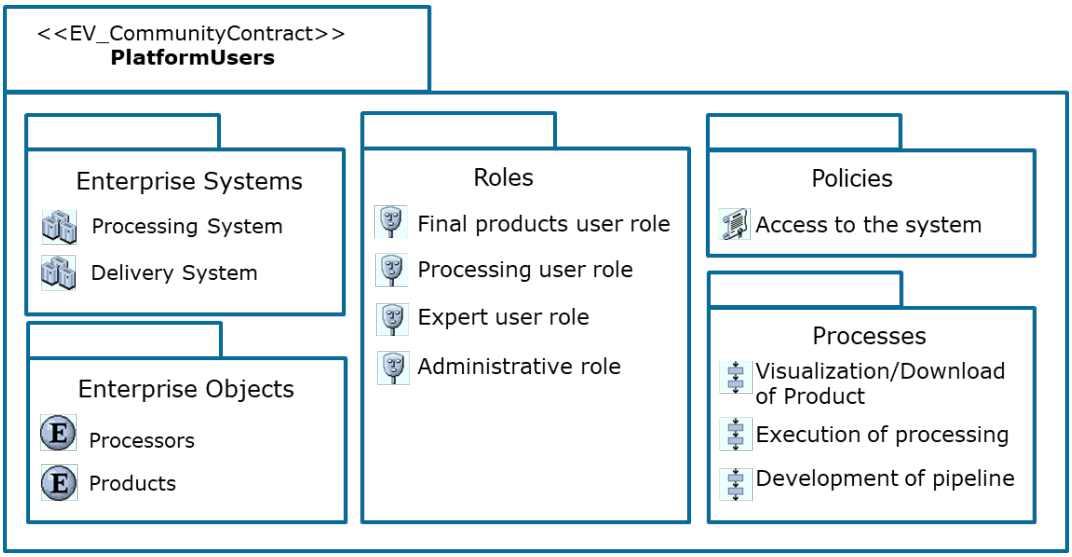


Figure 9: Diagram representing the enterprise view in terms of the community of users of the platform

In the diagram, beside to the already mentioned roles, also the objects have been identified which represents for us the objects responsible to satisfy the requirements of the platforms. In our case, the information to the user is provided by the mean of processor that generates products. Beside, Enterprise Systems are the systems identified to meet the challenges described in paragraph 2.1 (processing, delivery).

In particular, the Enterprise Systems are:

- Processing system: is the object that execute the pipeline and is responsible of the activation of dynamic workers with proper processor configured and ready to be started
- Delivery system: is the object that delivers the products to the product user

While the Enterprise Objects are

- Processor: is the object that contains the algorithm to be executed
- OGC Services: is the object that exposes OGC services on top of the products
- Web Interface: is the object that allow to access to the OGC Services

Policies are the rules applied for the access to the system to different role. Processes are the main processes behind the use of the platforms.

Here are the interaction diagrams for the different roles.

3.2.1.1 User of final products role

The following picture shows the interaction of the Product User role with the platform. As mentioned in the solution concept, in batch processing, the delivery of products (serving layer) is not directly interacting with the batch processing.

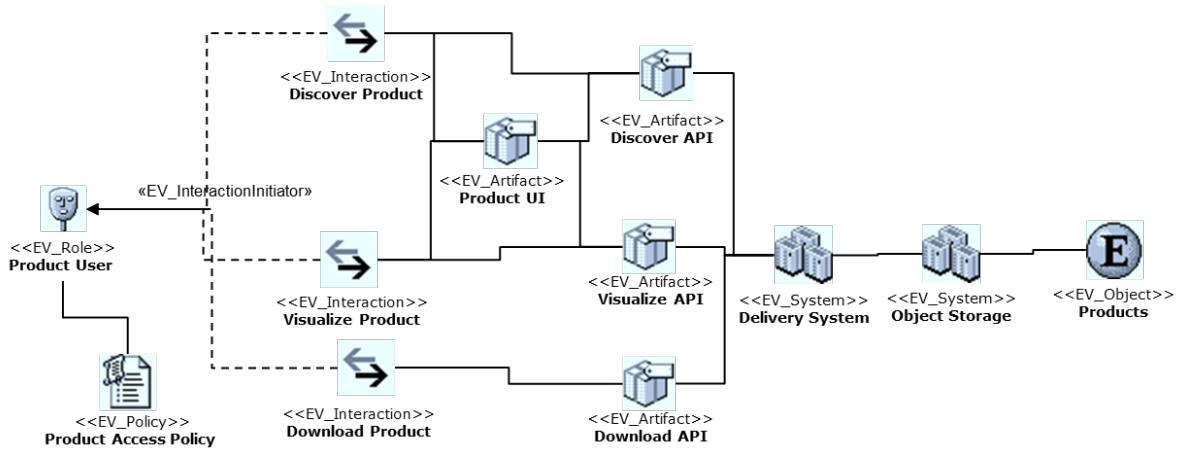


Figure 10: Diagram representing the interaction of Product User as introduced before

The image shows the role of the Product User that must adhere to the Product Access Policy and interact with the platform by discovery, visualization and download of products. Each of the interaction is linked to a platform artefact that is an API providing access to resources offered by the OGC Server. Finally, the Object Storage hosts the products (coming from the processing). Here is a reference to the scheme:

Product Access Policy:

- The Product User access the platform with credentials and is allowed to perform the interaction described below

Interactions:

- Discover Product: the user accesses the discovery and can use both a UI and an API to discover metadata using also filtering capabilities (OGC WFS standard)
- Visualize Product: the user accesses the visualization and can both use the UI and an API to visualise the product with filtering capabilities (OGC WMS standard)
- Download Product: the user accesses the download and can use an API to download the product with filtering capabilities (OGC WCS standard); the download of the product is managed also by a basic HTTP server

Artefact:

3.2.1.2 User of processing service role

The following picture shows the interaction of the Processing User role with the platform. This includes the discovery of the processing capabilities, the execution of a processing (a complex pipeline) and the monitoring of the pipeline execution.

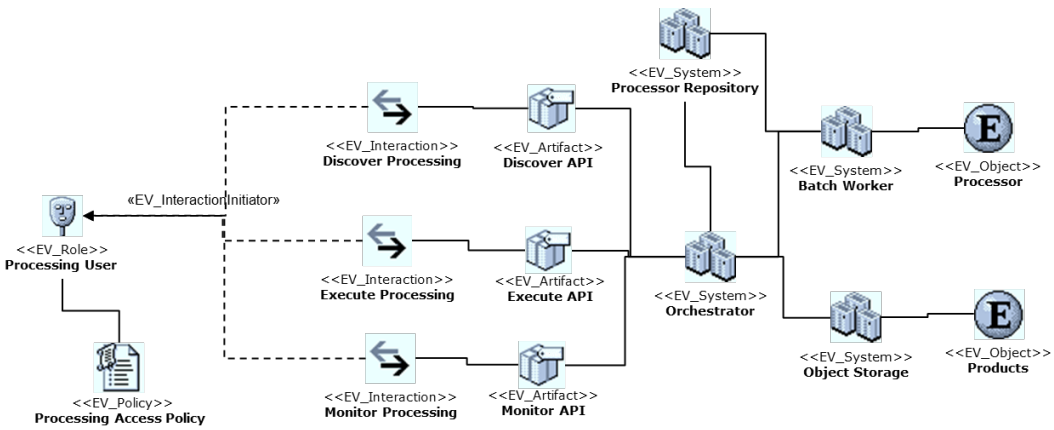


Figure 11: Diagram representing the interaction of Processing User as introduced before

3.2.1.3 Expert user role

The following picture shows the interaction of the Expert User role with the platform that includes the interaction using a Pipeline Development UI that is used both for the deployment of a single processor and for the chaining of processors into a pipeline.

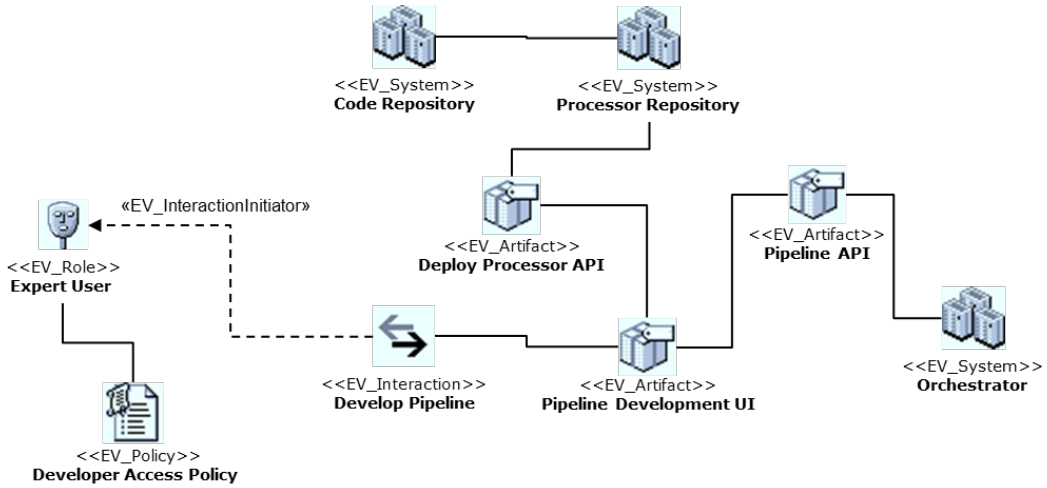


Figure 12: Diagram representing the interaction of Expert User as introduced before

3.2.2 Information Viewpoint

The Information Viewpoint deals with the semantic of the information exchanged during the interactions described in the Enterprise Viewpoint. For each interaction, we describe an interface with an information model coming from a public standard and example from the technology:

Interaction	Role	Information Model Reference	Example [2]
Discovery of products	Product User	OGC WFS 2.0.0 implementing also pagination and CQL filters that allow for complex querying of Metadata	WFS reference WFS filtering using CQL
Visualisation of products	Product User	WMS Standard with versions 1.1.1 and 1.3.0 with some extensions to the WMS specification made by the Styled Layer Descriptor (SLD) standard to control the styling of the map output.	WMS reference
Download of products	Product User	WCS Standard 1.0, 1.1, 2.0 that can return products in GeoTIFF format. NetCDF products are delivered with the use of standard HTTP protocol.	WCS reference

[2] Geoserver Implemented services: https://docs.geoserver.org/stable/en/user/services/index.html and OpenEO standard API reference https://openeo.org/documentation/0.4/developers/api/reference.html

Discovery of Processing	Processing User	OpenEO API standard Process Discovery which allows to list the available processes from the back-end.	OpenEO Process Discovery
Execution of Processing	Processing User	OpenEO Batch Job Management implementing the creation and execution of a Batch Job based on a Process Graph already stored	OpenEO Data Processing
Execution of Processing	Processing User		
Develop Pipeline	Expert User	OpenEO Process Graph showing processing steps in the pipeline. The API allows to create new Pipeline based on available processes. The objective is to create predefined User Stored Graphs to be retrieved and launched using the OpenEO Data Processing API.	OpenEO Process Graph

Table 1: Mapping of interactions with Information Models based on available open standards

Part of the Information Viewpoint is also the system handles the following type of data packages:

- Sentinel-2 L1C, L2A
- Landsat-5 L1TP (Level-1 Precision Terrain) Tier 1 and L2SP
- Landsat-7 L1TP (Level-1 Precision Terrain) Tier 1 and L2SP
- Landsat-8 L1TP (Level-1 Precision Terrain) Tier 1 and L2SP
- Sentinel-1 L1B GRD

In addition, as is shown later in the software architecture, the system is able to ingest and use generic geospatial types according to the capabilities of underlying technology. This allow to ingest and show all the needed satellite imagery (VHR imagery normally in GeoTIFF with specific Metadata natively supported) and ancillary data (like Shapefiles, GeoJSON etc.).

3.2.3 Computational Viewpoint

The Computational Viewpoint deals with the service organization inside the architecture. The architecture is conceived by integrating ExternalSystems providing the needed capabilities to use the ApplicationObjects.

The scheme below resumes the HRLC platform computational objects where the core of the architecture is represented by the Processors and the Pipelines (ApplicationObjects) that are instantiated using the capabilities of the ExternalSystems.

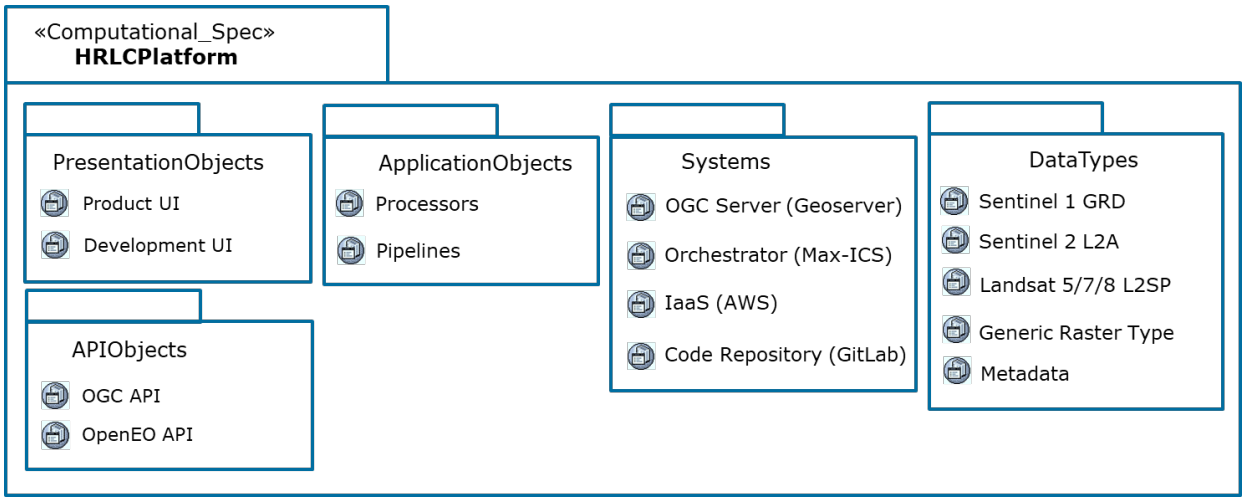


Figure 13: Overall Computational Viewpoint scheme

The following scheme presents the detail of the platform dedicated to the product delivery. The objective is to show some of the details of the architecture.

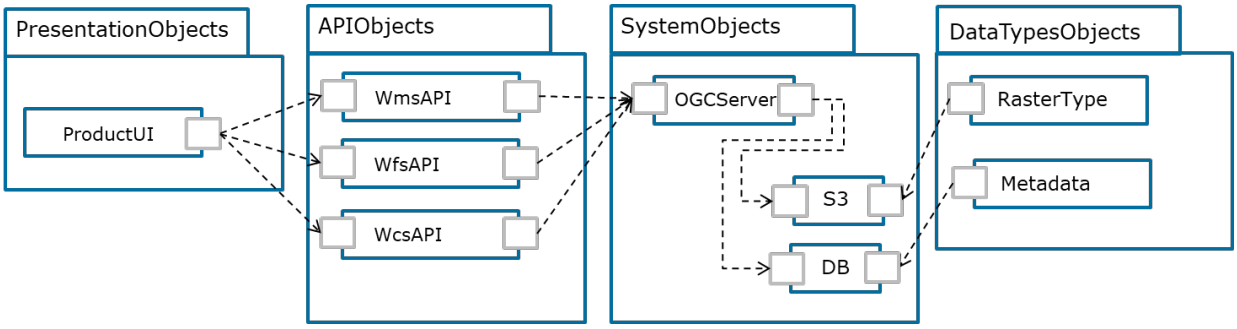


Figure 14: System software architecture for the delivery system

The software architecture of the delivery system is simple and does not give more details with respect t othe schemes presenting the interactions. The point that it is important to remark is the organisation of the data served by the OGCServer. Data is organised in the Object Storage (S3 API) for what concerns any RasterType meaning that for example any final product (consisting in GeoTIFF) is stored in an Object Storage allowing scalable parallel access. The OGCServer offers, for RasterType data, both WMS (visualisation) and WCS (distribution, conversion and delivery of data). For what concerns Metadata made normally of documents (XML or JSON), these are stored in a DB which allows fast access a querying capabilities through the OGC WFS API.

The following scheme presents the detail of the platform dedicated to the data processing that includes both the batch pipeline execution and the pipeline design.

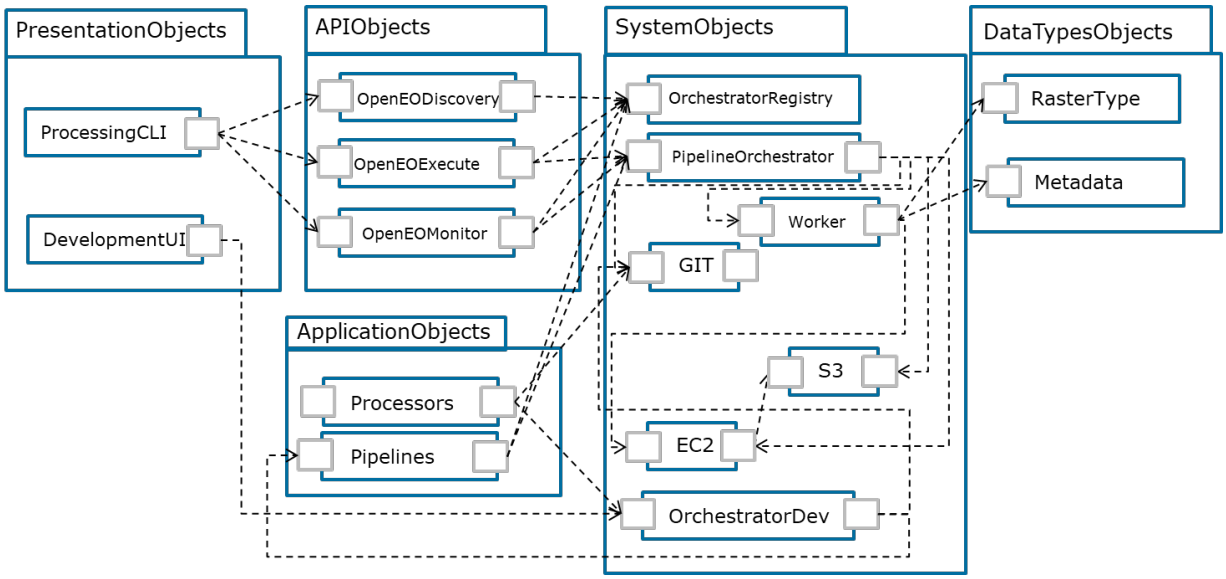


Figure 15: System software architecture for the processing system

The architecture above gives some more details about the internal objects inside the processing system. Even these details are transparent to the Users of the system and in general are handled by the technology (Max-ICS), it is useful to describe the mechanism that is behind the overall process of creating new pipelines and execute the pipelines. This is important because one of the challenges of the architecture is exactly to be flexible in the configuration of pipelines.

In this case, we start from the DevelopmentUI that interacts with what is called OrchestratorDev. OrchestratorDev is a set of functionalities that allow to create structures made by nodes and flows (a flow is a connection between two nodes). Behind the scene, this component handles most of the work as it has an internal registry that maintains these elements (nodes and flows) and is responsible for the flowing of messages among to nodes possible. Moreover, for each node, a GIT repository is created to allow the definition of the code for the processor (so the developer of the processor is responsible for the code of the processor and for the input/output with respect to the flow). Once the Pipeline is defined, it is registered in a public registry that can be accessed by the OpenEODiscovery to allow discovery of processes available on the platform. At this point, a new pipeline is ready to be executed.

The execution that is triggered by the OpenEOExecute is just a flow in the pipeline with messages flowing upon completion of node execution. Each node contains, considering the batch pipeline concept, a resource provider (interface to EC2), a configurator of the processor (starting from GIT/Docker) and the input/output with respect to persistent data (on S3) and messages activating the following node.

3.2.4 Engineering Viewpoint

The Engineering Viewpoint is particularly important for the present architecture as it explains how the distribution of the computation is done in the platform. In fact, the data processing challenge of HRLC is to process big amounts of data with optimization of computing resources (due to limited and constrained budget) and dynamic allocation of computing resources (in order to generate only the needed data products).

For what concerns the Delivery system, there are no particular requirements related to distribution of computation as it is a simple serving layer and the scalability is currently not in the scope of the project.

For what concerns the Processing system, the architecture heavily relies on the technological platform Max-ICS to handle this challenge. As explained in the Solution Concept (3.1) a pipeline is a set of steps that automatically are created (from a repository of code) and then executed following a graph.

A node, as described before, is anything having a strongly integrated view of resources (resource provision, configuration, scalable execution) and in the current architecture we consider it as a whole. In our case, each node can be seen as an elastic processing system acting with a policy. The reason why we have the possibility to defer to Max-ICS the responsibility of the node execution is explained as follow for what concerns the scalability.

For each node in a pipeline, the user can provide:

- A name
- A Type and Subtype
- The expected number of instance of this node to run (also with automatic scaling)
- A security class to define on which cloud type the application needs to run
- The number of CPU cores to assign to each of the node
- Memory to assign to each of the node
- The desired data-stores

Once created, Max-ICS will instantiate the different components with the provided source code by automating the DevOps process. The number of running instance is by default fixed and defined by the user. However, in various cases it might be interesting to modify dynamically the number of running instances in order to cope with data volume and velocity to treat. In particular, when the velocity highly varies the immobilisation of resources can be costly. It is therefore important to have an auto-scaling mechanism to deploy the adequate number of instances according to the data pressure.

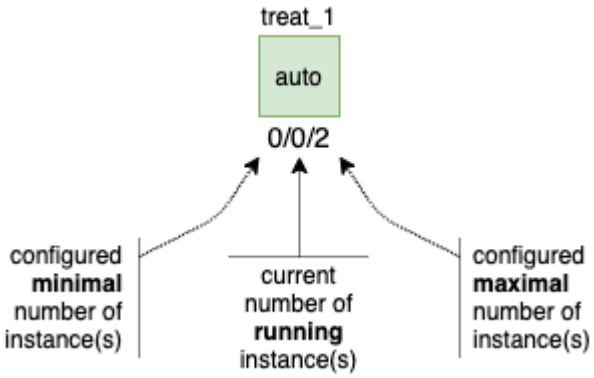


Figure 17: running instance representation in auto-scaling case

The above figure represents the way Max-ICS illustrates the current status of the running instance for a particular node with auto-scaling activated. In this example, the user configures the node to run between 0 (minimum) and 2 (maximum) instances. Currently 0 instances are running: by default, the number of instances running at start is set to the configured minimum. The autoscaling mechanism used by Max-ICS is based on message pressure (i.e. the messages are in the nodes, the more instances are automatically activated). This mechanism allows to have behaviours such as those shown in the following figure where the amount of instances follows smoothly the number of messages in a node.

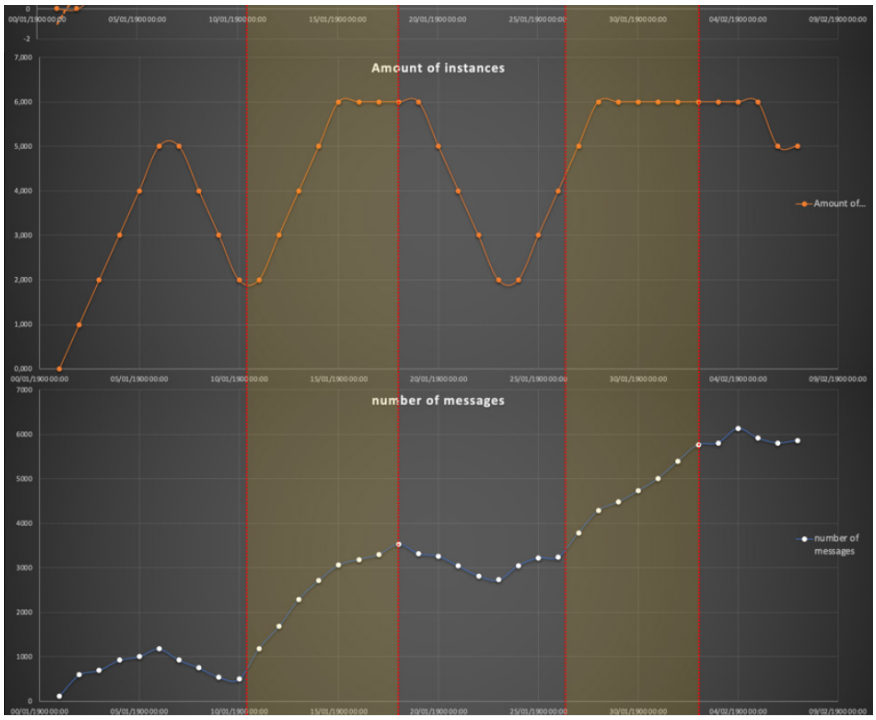


Figure 18: behaviour of a node in autoscaling mode

3.2.5 Tecnology Viewpoint

The following table explain the mapping of the architecture to the technological choices.

Entreprise System	Component	Tehcnology
Processing System	Orchestrator	Max-ICS, http://www-max-ics.earthlab.lu/
	IaaS	AWS S3, https://aws.amazon.com/s3/?nc1=h_ls AWS EC2, https://aws.amazon.com/ec2/?nc1=h_ls
	Code Repository	GITLAB, https://about.gitlab.com/
Delivery System	OGC Server	Gesoerver, http://geoserver.org/
	IaaS	AWS S3, https://aws.amazon.com/s3/?nc1=h_ls AWS EC2, https://aws.amazon.com/ec2/?nc1=h_ls

Table 2: mapping of the architecture to technology integrated into the system

An important point is related to the support of Raster data and in particular of missions that includes metadata like Sentinel 2 SAFE or Pleiades DIMAP. In general this is handled by GDAL library which natively support these raster format (or data package since it includes also metadata), example of this are:

- Pleiades DIMAP: https://gdal.org/drivers/raster/dimap.html
- Sentinel 2 SAFE: https://gdal.org/drivers/raster/sentinel2.html
- Sentinel 1 SAFE: https://gdal.org/drivers/raster/safe.html

Finally, we add a view of the complex architecture behind Max-ICS by EarthLab Luxembourg (http://www.earthlab.lu/), a European Start-up which provides platform services and technology for big-data processing and artificial intelligence. Max-ICS is fully based on open-source technology and is in the class of PaaS

(Platform As As Service) for the provision of big-data processing capability and in the class of SaaS (Software As a Service) for what concerns Artificial Intelligence tools.

The architecture of Max-ICS, as shown in the video https://www.youtube.com/watch?v=hxVcQLIIDqI&t=2498s is based on several components like Mesos (http://mesos.apache.org/), Puppet/Foreman (https://theforeman.org/) etc. and is a complex integration of such components to provide higher level services.

3.3 Pipelines architecture

3.3.1 Product Summary

The table below summarizes the product to be delivered after the processor/pipeline integration phase.

	Static Map	Historical Maps - Classification	Historical Maps - Change Detection	Historical Maps - NDVI/EVI
Products	Classification Map	Classification Map	Change Detection Map	Raster Map
Resolution (meters)	10	30	30	30
Source Data	Sentinel 1, Sentinel 2	Landsat 5-7-8	Landsat 5-7-8	Landsat 5-7-8
Years	2019	1990-1995, 1996-2000, 2001-2005, 2006-2010, 2011-2015	Every year from 1990 to 2015	Every 4 months from 1990 to 2015 (selecting the three less cloudy images in the year)
Projection	UTM WGS84	Latitude Longitude WGS84	Latitude Longitude WGS84	Latitude Longitude WGS84
Econding	Multiple files (both NetCDF and GeoTIFF) organised in tiles	Multiple files (both NetCDF and GeoTIFF)	Multiple files (both NetCDF and GeoTIFF)	Multiple files (both NetCDF and GeoTIFF)

Table 3: Summary table of the products

The following paragraphs describes the pipelines and the processors as foreseen for the first production, some minor adjustments will be made during the integration phase.

3.3.2 Static Map Chain

As described in the SRD, the overall concept of the Static Map chain is to have parallel processing chains for SAR and Optical processing that have two integration points in the Geolocation and Decision Fusion (final step).

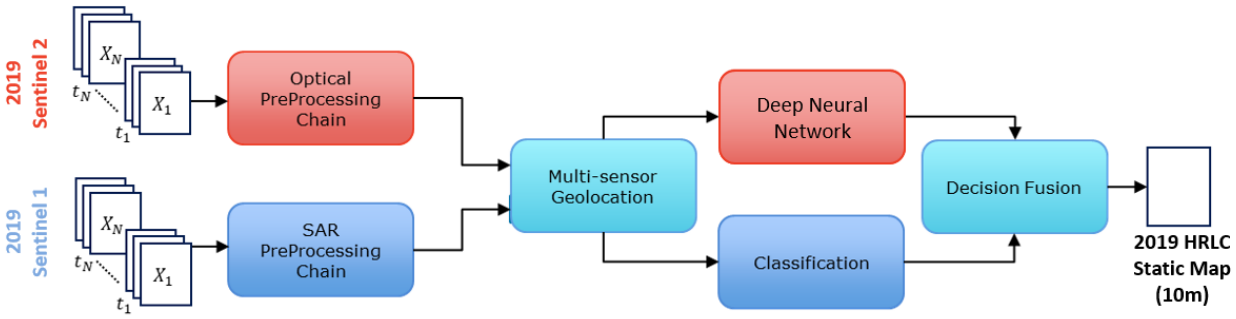


Figure 19: Static map chain illustration

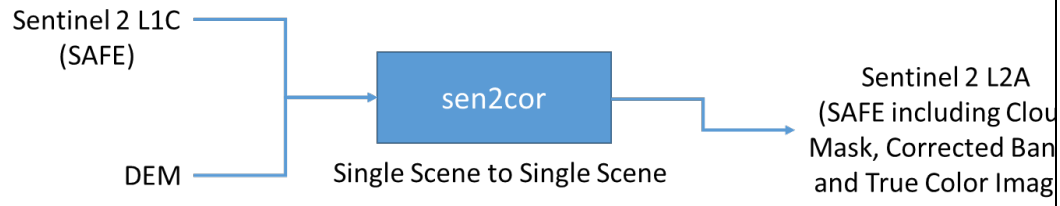
The engineering of such pipeline will be done using the described platform on the basis of the following description in the processors. Some minor modifications and updates are foreseen in the integration phase.

The input data (Sentinel 1 and Sentinel 2) can be retrieved by different sources such as:

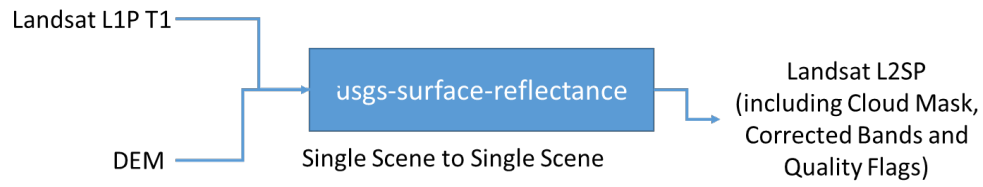
- Local AWS public dataset bucket
- Google Public datasets
- Copernicus Sci-hub and Cop-hub assuming proper credential are given

The download is managed by specific ingestion nodes managed by the engineering team.

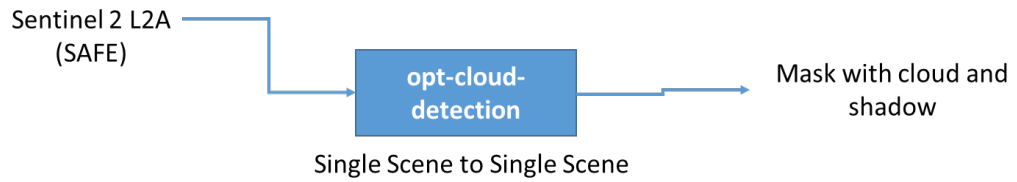
3.3.2.1 Pre-processing Optical

s2-atm-correction	
Description	Atmospheric Correction using sen2cor package included in ESA SNAP
Type	Single Scene to Single Scene | Multi Scene (Time) to Single Scene | Multi Scene (Time) to Multi Scene (Time)
Single Step Workflow	
Description	The sen2cor processor uses Sentinel 2 L1C together with DEM to deliver Sentinel 2 L2A data. The input package consists in L1C SAFE package described in SAFE product structure. The processor takes as input a single scene and outputs a single scene. It is important to note that the images coming from ESA PDGS L2A processor and those coming from sen2cor may have slightly differences due to the use of a different DEM (PlanetDEM for PDGS, SRTM for sen2cor).
Input Format	Sentinel 2 L1C, SAFE product structure, Images are encoded as JPEG 2000 https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/data-formats
Input Example	*<link to a zip or on-line example>*
Ancillary Data	DEM (Automatically downloaded) from SRTM (different from the DEM used by the PDGS L2A processor that is the PlanetDEM as described in sen2cor manual)
Output Format	Sentinel 2 L2A, SAFE product structure, Images are encoded as JPEG 2000 https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/data-formats
Output Example	Available from ESA SciHub

Input (Matrix Size, Bands, Scenes)	Matrix: 10980 * 10980 (at 10 m resolution) Bands: 10 Scenes: 1
Output (Matrix Size, Bands, Scenes)	Matrix: 10980 * 10980 (at 10 m resolution) Bands: 13 Scenes: 1
Hardware Needs	Number of CPUs: 2 Peak Memory consumption: 10GB RAM Working ephemeral storage: 50GB SSD
Performance Estimation (minutes)	10 – 30
Interface Type	Command Line
Interface Description	Installation package & documentation available at: https://step.esa.int/main/third-party-plugins-2/sen2cor/

landsat-atm-correction	
Description	The surface reflectance project contains application source code for producing surface reflectance products. It currently consists of LEDAPS for Landsats 4-7 and LaSRC for Landsat 8.
Type	Single Scene to Single Scene | Multi Scene (Time) to Single Scene | Multi Scene (Time) to Multi Scene (Time)
Single Step Workflow	
Description	The usgs surface reflectance processor uses Landsat L1TP (4/5/7/8) together with DEM to deliver Landsat L2SP data. The processor takes as input a single scene and outputs a single scene.
Input Format	Landsat L1TP products are GeoTIFF images with associated metadata, specification can be found here https://www.usgs.gov/media/files/landsat-collection-1-level-1-product-definition
Input Example	Available from USGS
Ancillary Data	DEM (Automatically downloaded)
Output Format	Landsat L2SP products are GeoTIFF images with associated metadata, specification can be found here https://www.usgs.gov/media/files/landsat-4-7-surface-reflectance-code-ledaps-product-guide and here https://www.usgs.gov/land-resources/nli/landsat/landsat-surface-reflectance
Output Example	Available from USGS upon ordering
Input (Matrix Size, Bands, Scenes)	Matrix: 8000 * 8000 Bands: 11, 7 (for Landsat 4-7) Scenes: 1

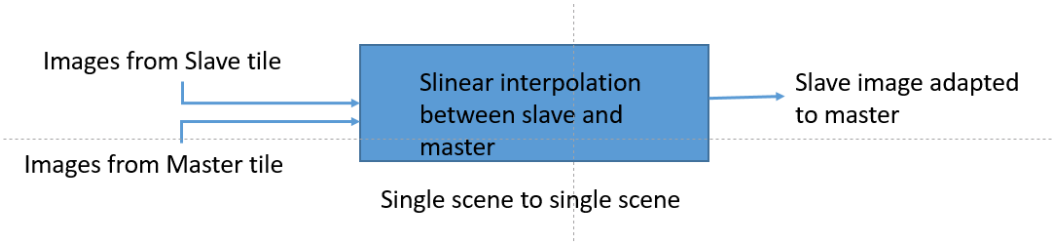
Output (Matrix Size, Bands, Scenes)	Matrix: 8000 * 8000 Bands: 10, 6 (for Landsat 4-7) Scenes: 1
Hardware Needs	Number of CPUs: 2 Peak Memory consumption: 10GB RAM Working ephemeral storage: 50GB SSD
Performance Estimation (minutes)	10 – 30
Interface Type	Command Line
Interface Description	Installation package & documentation available at: Code available at: https://github.com/USGS-EROS/espa-surface-reflectance

opt-cloud-detection	
Description	Cloud and shadow detection
Type	Single Scene to Single Scene
Single Step Workflow	
Description	Reads the cloud mask and shadow mask from original S2 L2A
Input Format	S2 L2A Tiles SAFE
Input Example	https://drive.google.com/drive/folders/1bKLbKS9rv5uHmhnQ4FxfomvsU4kmaBjP *S2B_MSIL2A_20180723T135109_N0206_R024_T21KXT_20180723T202458.SAFE.zip*
Ancillary Data	
Output Format	GeoTIFF
Output Example	https://drive.google.com/drive/folders/1bKLbKS9rv5uHmhnQ4FxfomvsU4kmaBjP MSIL2A_20180723T135109_N0206_R024_T21KXT_cloudMediumMask.tif
Input (Matrix Size, Bands, Scenes)	Matrix : 10980*10980*13 Bands: 13 Scenes: 1 Image size : 1.12 GB
Output (Matrix Size, Bands, Scenes)	Matrix : 10980*10980 Bands: 1 Scenes: 2 Image size : 230 Mb

Hardware Needs	Number of CPUs: 2 Peak Memory consumption: 10Gb RAM Working ephemeral storage: 30 Gb
Performance Estimation (minutes)	Few minutes (1-20)
Interface Type	Command Line
Interface Description	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/opt-cloud-detection

opt-spectral-filtering	
Description	Spectral filtering
Type	Single Scene to Single Scene
Single Step Workflow	
Description	Saves ten bands (B02','B03','B04','B08','B05','B06','B07','B8A','B11','B12') of S2 image to tiff, the bands with 20m spatial resolution are interpolated to 10m.
Input Format	S2 L2A Tiles SAFE
Input Example	https://drive.google.com/drive/folders/1bKLbKS9rv5uHmhnQ4FxfomvsU4kmaBjP S2B_MSIL2A_20180723T135109_N0206_R024_T21KXT_20180723T202458.SAFE.zip
Ancillary Data	N/A
Output Format	tif
Output Example	https://drive.google.com/drive/folders/1bKLbKS9rv5uHmhnQ4FxfomvsU4kmaBjP MSIL2A_20180723T135109_N0206_R024_T21KXT.tif
Input (Matrix Size, Bands, Scenes)	Matrix : 10980*10980*13 Bands: 13 Scenes: 1 Image size : 1.12 GB
Output (Matrix Size, Bands, Scenes)	Matrix : 10980*10980*10 Bands: 10 Scenes: 1

	Image size : 2.24 GB
Hardware Needs	Number of CPUs: 4 Peak Memory consumption: 16GB Gb RAM Working ephemeral storage: 30 Gb
Performance Estimation (minutes)	2
Interface Type	**Command line**
Interface Description	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/opt-spectral-adaptation

opt-spectral-adaptation	
Description	Spectral adaptation
Type	Single Scene to Single Scene
Single Step Workflow	
Description	We select master tile, then we randomly select 300 samples coming from different images, which are belonging to master tile. Next, within images of slave tile, we look for the most spectraly similar pixels for each of 300 master samples. Finally, we define a sline interpolation trained on those 300 samples for every band separetly. The interpolation is run for every image of slave tile.
Input Format	tif
Input Example	https://drive.google.com/drive/folders/1C4T5VSeXJQ69EbxGQWSjM8YKSg_4h47u *MSIL2A_20190328T140051_N0211_R067_T21KUQ.tif*
Ancillary Data	https://drive.google.com/drive/folders/1C4T5VSeXJQ69EbxGQWSjM8YKSg_4h47u MSIL2A_20190830T140059_N0213_R067_T21KUQ.tif MSIL2A_20191223T140051_N0213_R067_T21KUQ.tif MSIL2A_20190330T135119_N0211_R024_T21KXT.tif MSIL2A_20190822T135111_N0213_R024_T21KXT.tif MSIL2A_20191220T135111_N0213_R024_T21KXT.tif
Output Format	tif
Output Example	https://drive.google.com/drive/folders/1C4T5VSeXJQ69EbxGQWSjM8YKSg_4h47u *MSIL2A_20190328T140051_N0211_R067_T21KUQ_adapted.tif*

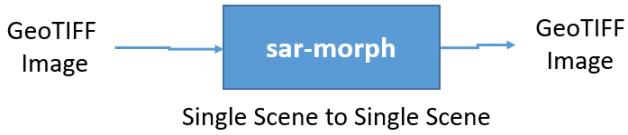
Input (Matrix Size, Bands, Scenes)	Matrix : 10980x10980x10 Bands: 10 Scenes: 6 Image size : 14GB
Output (Matrix Size, Bands, Scenes)	Matrix : 10980x10980x10 Bands: 10 Scenes: 1 Image size : 2.3 GB
Hardware Needs	Number of CPUs: 4 Peak Memory consumption: 16 RAM Working ephemeral storage: 500 Gb
Performance Estimation (minutes)	Around 40 min for the training of each slave tile, than each image around 2min.
Interface Type	**Command Line TBD**
Repository	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/opt-spectral-adaptation

3.3.2.2 Pre-processing SAR

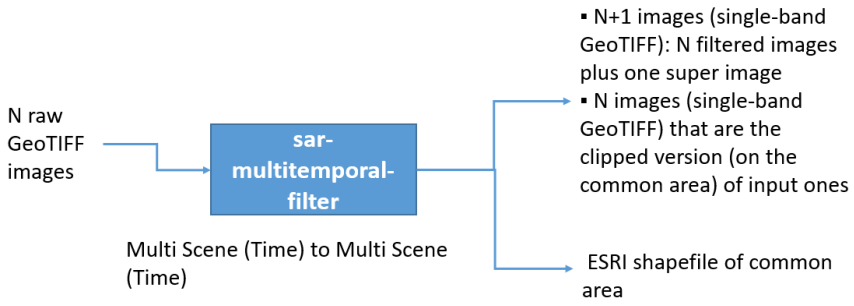
sar-despeckle-filter	
Description	Despeckling filter (Lee)
Type	Single Scene to Single Scene (single polarization)
Single Step Workflow	
Description	The sar-despeckle-filter processor applies the Lee filter for reducing the speckle effect present in SAR image. The processor accepts in input a GeoTIFF single scene and produces as filtered product a GeoTIFF single scene.
Input Format	single image (single-band GeoTIFF image)
Input Example	*https://drive.google.com/open?id=1wScCuqHl8btByOVL597JSxR9icleYwQN*

Ancillary Data	N/A
Output Format	single image (single-band GeoTIFF image)
Output Example	https://drive.google.com/open?id=1zf_koteLfxorBOBStfiziMu98pG0eseR
Input (Matrix Size, Bands, Scenes)	Matrix : 16140*8244 Bands: 1 Scenes: 1 Image size : 933 Mb
Output (Matrix Size, Bands, Scenes)	Matrix : 16140*8244 Bands: 1 Scenes: 1 Image size : 507 Mb
Hardware Needs	Number of CPUs: 4 Peak Memory consumption: 16Gb RAM Working ephemeral storage: 500 Gb
Performance Estimation (minutes)	0.35
Interface Type	Command Line : Python feature_extraction <switchs> ▪ -i is the path of the single raw image (GeoTIFF); ▪ -o is the path of the output; ▪ -t LEE is used for applying the Lee filter; ▪ -ba is the number of band (default: ba=1); ▪ -w size of filter kernel (default: w=9); ▪ -b dimension of sliding window (default: b=256) Example: python feature_extraction.py -i S1B_IW_GRDH_1SDV_20180108T013859_20180108T013924_009073_010386_4F18_VH.tif -o 1_filtered_images\single_LEE_filter\42wxs_20180108_LEE.tif -t LEE
Repository	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/sar-despeckle-filter

sar-morph	
Description	Classes morphological erosion
Type	Single Scene to Single Scene

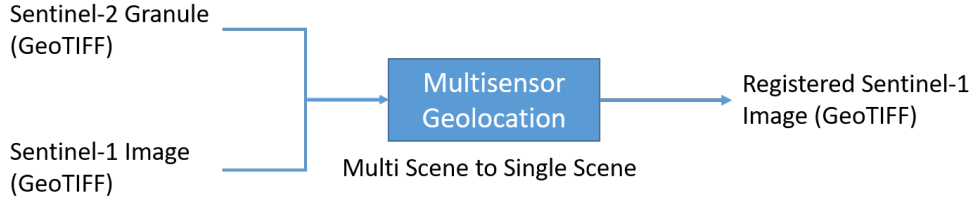
Single Step Workflow	 Single Scene to Single Scene
Description	The sar-morph processor uses a land cover map to carry out the morphological erosion of classes present inside it. The processors takes as input a GeoTIFF single scene and provides in output a GeoTIFF single scene.
Input Format	single scene (single-band GeoTIFF image)
Input Example	*https://drive.google.com/open?id=1r2YEwPi5At16wX6frHtlnyiVOzWOeI2m*
Ancillary Data	N/A
Output Format	single scene (single-band GeoTIFF image)
Output Example	https://drive.google.com/open?id=18ZwJ7nrUXHQD5s0AXZCOgk9bv_51WAK
Input (Matrix Size, Bands, Scenes)	Matrix : 2145*1061 Bands: 1 Scenes: 1 Image size: 2.17 Mb
Output (Matrix Size, Bands, Scenes)	Matrix : 2145*1061 Bands: 1 Scenes: 1 Image size: 801 Kb
Hardware Needs	Number of CPUs: 4 Peak Memory consumption: 16Gb RAM Working ephemeral storage: 500 Gb
Performance Estimation (minutes)	0,033
Interface Type	Command Line: python erosion_class.py <switchs> ▪ -i refers to the path of the single image (GeoTIFF) ▪ -o specifies output product ▪ -c is the list of classes that will be extracted ▪ -t refers to minimum number of pixel of an eroded class (default: t=2000) ▪ -w is the size of erosion kernel

	Example : python erosion_class.py -i ESACCI_42wxs_2018.tif -o input_TrainingSet_extraction\ESACCI_42wxs_2018_eroded.tif -c 50 70 71 72 60 61 62 80 81 82 121 122 160 170 180 130 10 11 12 20 160 170 180 140 200 201 202 210 220 -t 2000 -w 3
Repository	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/sar-morph

sar-multitemporal-filter	
Description	Multi-temporal despeckling filter
Type	Multi Scene (Time) to Multi Scene (Time)
Single Step Workflow	
Description	The sar-multitemporal-filter processor is devoted to speckle reduction on a SAR time series. The processor takes in input a collection of N images and produces a multi scene.
Input Format	N images (single-band GeoTIFF)
Input Example	https://drive.google.com/open?id=1PYcIdcYTAfY-hK1ZxCr5jLePJBJY2wUt
Ancillary Data	N/A
Output Format	▪ N+1 images (single-band GeoTIFF): N filtered images plus one super image ▪ N images (single-band GeoTIFF) that are the clipped version (on the common area) of input ones ▪ ESRI shapefile of common area
Output Example	https://drive.google.com/open?id=1xvqmriKoh7g0UfhrlmymWzSaagJPqdW_
Input (Matrix Size, Bands, Scenes)	Matrix : (16144*8244)*(N=8) Bands: 1 Scenes: 1 Image size : 933*(N=8) Mb
Output (Matrix Size, Bands, Scenes)	Images: Matrix : (16144*8244)*(2*N+1=17) Bands: 1 Scenes: 1 Image size : 507*(N+1=9) Mb + 990*(N=8) Mb ESRI shapefile:

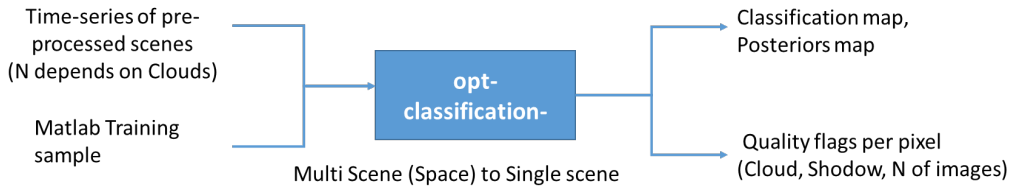
	Matrix : N/A Bands: N/A Scenes: 1 Image size : 808 b
Hardware Needs	Number of CPUs: 4 Peak Memory consumption: 16Gb RAM Working ephemeral storage: 500 Gb
Performance Estimation (minutes)	14,17
Interface Type	Command Line python multi_despeckling.py <switchs> ▪ -i path of the directory where the inputs (N raw GeoTIFF images) are located; ▪ -o is the path of the output directory; ▪ -c for cropping the images on the common area; ▪ -b dimension of sliding window (default: b=256); ▪ -w dimension of filter kernel (default: w=9) Example : python multi_despeckling.py -i S1_data\42WXS\VH\ -o 1_filtered_images\multitemporal_filter\ -c
Repository	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/sar-multitemporal-filter

3.3.2.3 Multi-sensor Geo-location

optsar-coregister	
Description	The multi-sensor geolocation processor is aimed at registering a pair of optical and SAR images. The programming language is Python and it is run in an Anaconda virtual environment.
Type	Multi Scene to Single Scene Input: a pair of images. One S2 granule and one S1 super-image whose area corresponds to the S2 granule (both in number of pixels, resolution, and CRS/projection).
Single Step Workflow	
Input Format	GeoTIFF image
Input Example	https://drive.google.com/drive/folders/1g54_RrKShn4PBLNeGVrsRrdY3xilHW0H

Ancillary Data	
Output Format	GeoTIFF image
Output Example	https://drive.google.com/drive/folders/1ucXac2QPJM6yv11g6laWipakNHNOQxKa
Input (Matrix Size, Bands, Scenes)	S2 granule Matrix: 10980 x 10980 Bands: 10 Scenes: 1 S1 super-image corresponding to the S2 granule Matrix: 10980 x 10980 Bands: 1 Scenes: 1
Output (Matrix Size, Bands, Scenes)	Registered S1 super-image Matrix: 10980 x 10980 Bands: 1 Scenes: 1
Hardware Needs	Number of CPUs: 4 *I did the experimentation on a quad-core i7-4790 @ 3.60GHz. It is necessary to test if the performances are affected by a lower number of CPUs.* Peak Memory consumption: 8GB RAM Working ephemeral storage: 20GB HDD
Performance Estimation (minutes)	30 minutes: Experimentation done on a quad-core i7-4790 @ 3.60GHz. To be tested on different configurations. Reference run: RR Africa dataset (s2 granule + s1 super-image) required 28 minutes on my machine and 67 minutes when run through docker.
Interface Type	Command Line
Interface Description	The options are specified in the *configuration.json* file. It is necessary to write the configuration file before running the container. The Dockerfile already contains the entrypoint running the command line interface that passes the *configuration.json* file as argument. Hence, provided that the *configuration.json* file exists and it is filled with the required data, there is no need to specify anything when running the container using the *docker-compose up* command. Other parameters, which are fixed and setup by the developers, are stored in the *processor_configuration.json* file.
Repository	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/module-name

3.3.2.4 Classification Optical

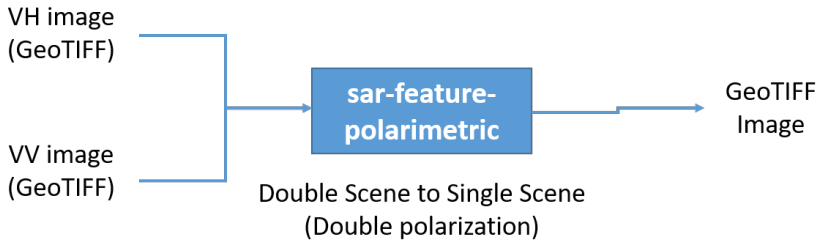
opt-lstm-classification	
Description	Classifier training and inference using multispectral data
Type	Multi Scene (Space) to Single Scene
Single Step Workflow	
Description	The opt-classification-training uses pre-processed multispectral Sentinel 2 scenes (corresponding to SAFE tiles) and a set of points for training the classifier. The processor produces in output a both the classification map and the information needed to the calculation of the confidence level associated to pixels (N images with cloud, N total images, N images with shadow).
Input Format	Multi scene (single-band GeoTIFF image) and ESRI shapefiles
Input Example	Pre-processed multi-spectral data https://drive.google.com/drive/folders/1uajWhtbIMNG7iwbOX3a2ZjVPdET4Gm5V
Ancillary Data	Training sample https://drive.google.com/drive/folders/12Uco_I6mKUb2KcIBgxmJCEG73-k5QVcG
Output Format	▪ JSON file ▪ JOBLIB file ▪ ESRI shapefile
Output Example	https://drive.google.com/drive/folders/1mATsI-gWSMLKxgYv57zf03JaTJOlLELu
Input (Matrix Size, Bands, Scenes)	Pre-processed multi-spectral data Matrix : 10980*10980 Bands: 10 Scenes: Variable Image size: Variable
Output (Matrix Size, Bands, Scenes)	Pre-processed multi-spectral data Matrix : 10980*10980 Bands: 4 Scenes: Variable Image size: Variable
Hardware Needs	Number of CPUs: 4 Peak Memory consumption: 16Gb RAM Working ephemeral storage: 500 Gb

Performance Estimation (minutes)	From 30 minutes to 2 hour
Interface Type	Command Line: python lstm.py <options> <input> <output>
Interface Description	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/opt-lstm-classification

3.3.2.5 Classification SAR

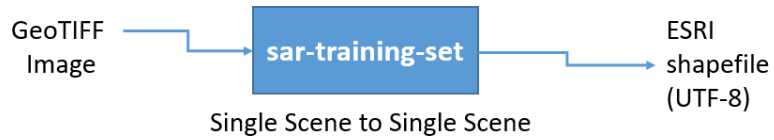
sar-feature-spatial	
Description	Feature extraction (spatial)
Type	Single scene to Single Scene
Single Step Workflow	 Single Scene to Single Scene
Description	The sar-feature-spatial performs the feature extraction for a single GeoTIFF image. The processor accepts in input a GeoTIFF single scene and produces as spatial feature a GeoTIFF single scene.
Input Format	Single-band GeoTIFF
Input Example	https://drive.google.com/open?id=1clrvhouHlDKpJYjz_inahYKDt85nIH43
Ancillary Data	N/A
Output Format	Single-band GeoTIFF
Output Example	https://drive.google.com/open?id=153OI2n8aA-LUIGdpXXLPJpTNcMNNYOQZ
Input (Matrix Size, Bands, Scenes)	Matrix : 16146*8245 Bands: 1 Scenes: 1 Image size : 507 Mb
Output (Matrix Size, Bands, Scenes)	Matrix : 16146*8245 Bands: 1 Scenes: 1 Image size : 507 Mb

Hardware Needs	Number of CPUs: 4 Peak Memory consumption: 16Gb RAM Working ephemeral storage: 500 Gb
Performance Estimation (minutes)	0.33
Interface Type	Command Line: python feature_extraction.py <switchs> ▪ i is the path of the SUPER IMAGE (output of sar-multitemporal-filter processor) (GeoTIFF); ▪ o is the path of the output; ▪ t is used for selecting the filter {LEE, MAX, MIN, MAXMIN, MEAN, MEDIAN}; ▪ ba is the number of band (default: ba=1); ▪ w size of filter kernel (default: w=9); ▪ b dimension of sliding window (default: b=256). Example : python feature_extraction.py -i I:\Tonia\experiment\2_feature\super_image.tif -o I:\Tonia\experiment\2_feature\super_image_MEAN.tif -w 5 -t MEAN
Interface Description	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/sar-features-spatial

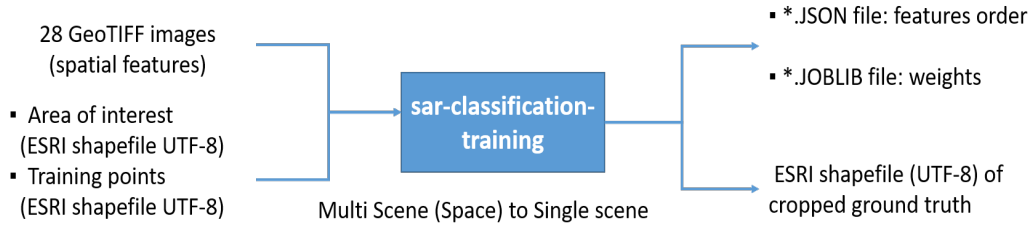
sar-feature-polarimetric	
Description	Feature extraction in double polarization case
Type	Double scene (double polarization) to Single Scene
Single Step Workflow	
Description	The sar-feature-polarimetric processor performs the features extraction taking in input the VH and VV acquisitions of the same scene. The processor outputs the polarimetric feature arranged into a GeoTIFF single scene.
Input Format	Double scene (Single-band GeoTIFF)

Input Example	▪ VH image ▪ VV image
Ancillary Data	N/A
Output Format	Single-band GeoTIFF
Output Example	https://drive.google.com/open?id=1AaujAr7R-RRmJn5AF0hJ5M4XBWBoEdOo
Input (Matrix Size, Bands, Scenes)	Matrix : 16144*8244 Bands: 1 Scenes: 2 Image size : 937*2 Mb
Output (Matrix Size, Bands, Scenes)	Matrix : 16144*8244 Bands: 1 Scenes: 1 Image size : 507 Mb
Hardware Needs	Number of CPUs: 4 Peak Memory consumption: 16Gb RAM Working ephemeral storage: 500 Gb
Performance Estimation (minutes)	0.57
Interface Type	Command Line: python feature_extraction.py <switchs> ▪ i indicates the paths of the raw images (GeoTIFF). One path refers to VH image while the other to VV image; ▪ ois the path of the output; ▪ t is used for selecting the filter {SUM,MEAN_BAND,RATIO,DIFF}; ▪ ba is the number of band (default: ba=1); ▪ w size of filter kernel (default: w=9); ▪ b dimension of sliding window (default: b=256). Example : python feature_extraction.py -i I:\Tonia\experiment\S1_data\42WXS\VH\S1B_IW_GRDH_1SDV_20180213T013858_20180213T013923_009598_0114C0_1D4C_VH.tif

	I:\Tonia\experiment\S1_data\VV\S1B_IW_GRDH_1SDV_20180213T013858_20180213T013923_009598_0114C0_1D4C_VV.tif -o I:\Tonia\experiment\single_analysis\2_feature\img_DIFF.tif -t DIFF
Interface Description	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/sar-features-polarimetric

sar-training-set	
Description	Training set extraction
Type	Single Scene to Single Scene
Single Step Workflow	
Description	The sar-training-set processor aims to provide a set of points useful for training the classifier. The input is a GeoTIFF single scene and the processor outputs a collection of points in ESRI shapefile format.
Input Format	single scene (single-band GeoTIFF image
Input Example	*https://drive.google.com/open?id=18ZwJ7nrUXHQD5s0AXZCOqk9bv_51WAK*
Ancillary Data	N/A
Output Format	ESRI shapefile (UTF-8)
Output Example	https://drive.google.com/open?id=1ZXyfXWu9vFqLPPTw2wMMiocZQUGcDTZ5
Input (Matrix Size, Bands, Scenes)	Matrix : 2145*1061 Bands: 1 Scenes: 1 Image size: 801 Kb
Output (Matrix Size, Bands, Scenes)	Matrix : N/A Bands: N/A Scenes: 1 Image size: 8.72 Mb
Hardware Needs	Number of CPUs: 4 Peak Memory consumption: 16Gb RAM

	Working ephemeral storage: 500 Gb
Performance Estimation (minutes)	0,2
Interface Type	**Command line:** python random_coords_extracion.py <switchs> ▪ -gt idicates the path where the map used as training set (the ground truth) is located. It is a single GeoTIFF image and is the output of sar-morph processor. ▪ -o identifies the path where the output will be saved. ▪ -p is the coverage percentage of samples collected. ▪ -c is the list of classes to mine and use in high resolution legend ▪ -l is the list of label of high resolution legend Example : python random_coords_extracion.py -gt input_TrainingSet_extraction\ESACCI_42wxs_2018_eroded.tif -o training_set\ESACCI_42wxs_2018_eroded -p 80 -c [50] [70,71,72] [60,61,62] [80,81,82] [121] [122] [130] [10,11,12,20] [160,170,180] [140] [200,201,202] [210] [220] -l [10,20,30,40,50,60,70,80,90,100,110,130,140]]
Repository	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/sar-training-set

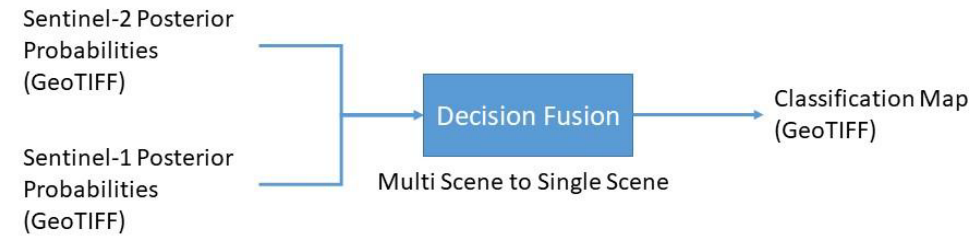
sar-classification-training	
Description	Classifier training
Type	Multi Scene (Space) to Single Scene
Single Step Workflow	
Description	The sar-classification-training uses spatial features (spatial multi scene) and a set of points for training the classifier. The processor requires in input also the shapefile of the scene of interest. The processor produces in output a set of information useful for the final classification.
Input Format	Multi scene (single-band GeoTIFF image) and ESRI shapefiles

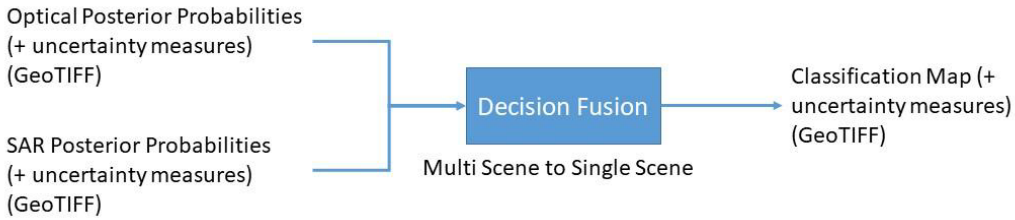
Input Example	▪ Features ▪ Training points ▪ ESRI shapefile of area
Ancillary Data	N/A
Output Format	▪ JSON file ▪ JOBLIB file ▪ ESRI shapefile
Output Example	https://drive.google.com/open?id=1VUeK90FICPCWo2ytXVMaTa5yI2jLqNeg
Input (Matrix Size, Bands, Scenes)	Features: Matrix : 16146*8245 Bands: 1 Scenes: 28 Image size: 507*28 Mb Training points (ESRI shapefile): Matrix : N/A Bands: N/A Scenes: N/A Image size: 8.72 Mb Area of interest (ESRI shapefile): Matrix : N/A Bands: N/A Scenes: N/A Image size: 808 b
Output (Matrix Size, Bands, Scenes)	JSON file: Matrix : N/A Bands: N/A Scenes: N/A Image size: 516 b JOBLIB file: Matrix : N/A Bands: N/A Scenes: N/A Image size: 4.18 Mb Cropped ground truth (ESRI shapefile): Matrix : N/A Bands: N/A

	Scenes: N/A Image size: 4.36 Mb
Hardware Needs	Number of CPUs: 4 Peak Memory consumption: 16Gb RAM Working ephemeral storage: 500 Gb
Performance Estimation (minutes)	1.62
Interface Type	Command Line: python training.py <switchs> ▪ gt idicates the path of training points. It is an ESRI shapefile and is the output of sar-training-set processor. ▪ i is the path of feature images directory; ▪ e image extension without dot (ex: img, tif, raw, ...) ▪ o path of output directory; ▪ c for selecting the classifier type {rf,svc}; rf for Random Forest, svc for Support Vector Classification with RBF Kernel; ▪ cl path of shapefile for cropping the ground truth image. It is one of sar-multitemporal-filter processor output. ▪ ne number of estimators for Random Forest Classifier (default: ne=100); ▪ g Kernel coefficient for SVC (default: g=1 / n_features). Example : python training.py -gt I:\Tonia\experiment\training_set\ESACCI_42wxs_2018_eroded.shp -i I:\Tonia\experiment\2_features\ -e tif -o I:\Tonia\experiment\3_trained_classifier\ -c rf -cl I:\Tonia\experiment\1_filtered_images\multitemporal_filter\clipped_source\clip\clip.shp
Repository	Code available at: https://lab.egeos-services.it/gitlab/cci-hrlc/processors/sar-svm-prediction

3.3.2.6 Decision Fusion

optsar-decisionfusion-1	
Description	The decision fusion processor is aimed at joining optical and SAR posterior probabilities in order to obtain a final classification map. The programming language is Python and it is run in an Anaconda virtual environment.
Type	Multi Scene to Single Scene

Single Step Workflow	
Input Format	GeoTIFF image
Input Example	https://drive.google.com/drive/folders/1KK6AsZ9qqVETQ1442p0rqo0LKBjq7mMY
Ancillary Data	Training set (optional)
Output Format	GeoTIFF image
Output Example	https://drive.google.com/drive/folders/15fpJ3jQloem3QBLbnW09atY0Ks1IoS6s
Input (Matrix Size, Bands, Scenes)	*Optical posterior probabilities on S2 granule.* Matrix: 10980 * 10980 Bands: number_of_classes_optical Scenes: 1 *SAR posterior probabilities on S2 granule.* Matrix: 10980 * 10980 Bands: number_of_classes_SAR Scenes: 1
Output (Matrix Size, Bands, Scenes)	*Same as Image Size IN* Matrix: 10980 * 10980 Bands: 2 (best and second-best class) Scenes: 1
Hardware Needs	Number of CPUs: 4 *Experimentation performed on a quad-core i5-4570 @ 3.20GHz. Need to test if the performances are affected by a lower number of CPUs.* Peak Memory consumption: 16GB RAM Working ephemeral storage: 16GB HDD
Performance Estimation (minutes)	2 *Experimentation performed on a quad-core i5-4570 @ 3.20GHz.*
Interface Type	Command Line
Interface Description	The parameters, which are fixed and setup by the developers, are stored in a *.json* file.
Repository	https://lab.egeos-services.it/gitlab/cci-hrlc/processors/optsar-decisionfusion

optsar-decisionfusion-2	
Description	The decision fusion processor is aimed at joining optical and SAR posterior probabilities in order to obtain a final classification map. The programming language is Python and it is run in an Anaconda virtual environment.
Type	Multi Scene to Single Scene
Single Step Workflow	
Input Format	GeoTIFF image
Input Example	https://drive.google.com/drive/folders/1KK6AsZ9qqVETQ1442p0rqo0LKBjq7mMY
Ancillary Data	Training set (optional)
Output Format	GeoTIFF image
Output Example	https://drive.google.com/drive/folders/15fpJ3jQloem3QBLbnW09atY0Ks1IoS6s
Input (Matrix Size, Bands, Scenes)	*Optical posterior probabilities on S2 granule.* Matrix: 10980 * 10980 Bands: number_of_classes_optical + uncertainty_related_measures Scenes: 1 *SAR posterior probabilities on S2 granule.* Matrix: 10980 * 10980 Bands: number_of_classes_SAR + uncertainty_related_measures Scenes: 1
Output (Matrix Size, Bands, Scenes)	*Same as Image Size IN* Matrix: 10980 * 10980 Bands: 1 (classification map) + n (uncertainty_related_measures) *multichannel output: the first channel will include the output land cover class label; the other channels will encode uncertainty in the land cover prediction. the number of such additional channels is being discussed and is expected to range from 1 to 5.* Scenes: 1
Hardware Needs	Number of CPUs: 4 *Experimentation performed on a quad-core i5-4570 @ 3.20GHz. Need to test if the performances are affected by a lower number of CPUs.* Peak Memory consumption: 16GB RAM

	Working ephemeral storage: 16GB HDD
Performance Estimation (minutes)	2 *Experimentation performed on a quad-core i5-4570 @ 3.20GHz.*
Interface Type	Command Line
Interface Description	The parameters, which are fixed and setup by the developers, are stored in a *.json* file.
Repository	https://lab.egeos-services.it/gitlab/cci-hrlc/processors/optsar-decisionfusion

3.3.3 Dynamic Map Chain – Classification/Change Map

As described in the SRD, the overall concept of the Dynamic Map chain (Multitemporal Change Detection and Trend Analysis) is to have parallel processing chains for generating the different products.

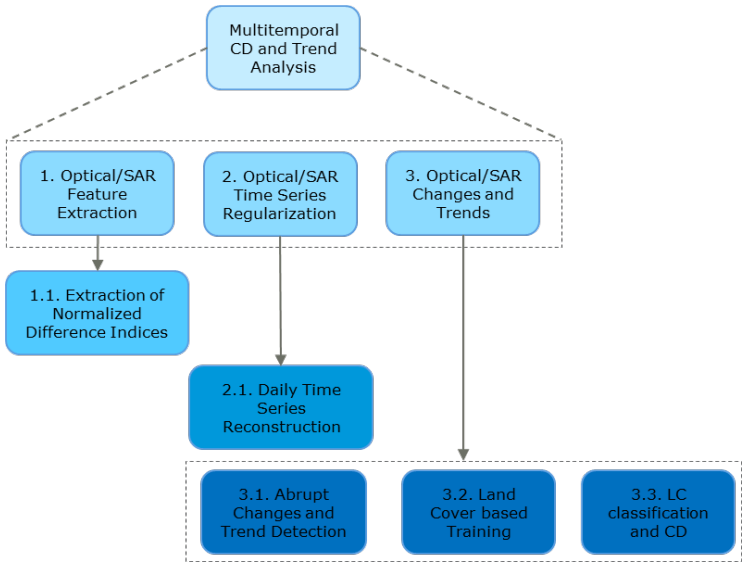


Figure 20: Dynamic map chain illustration

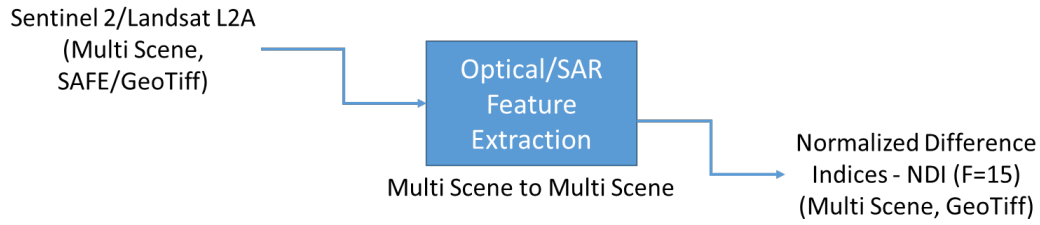
The engineering of such pipeline will be done using the described platform on the basis of the following description in the processors. Some minor modifications and updates are foreseen in the integration phase as some of the processors have to be finalized.

The input data (Landsat 5/7/8) are retrieved from:

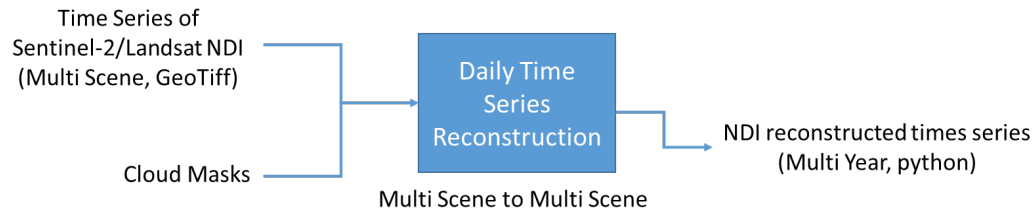
- Local S3 bucket after the retrieval from USGS using the available interfaces (not automatic procedure or API available for level L2SP)

3.3.3.1 Optical/SAR Feature Extraction

optsar-feature-extraction	
Description	1.1 Optical/SAR Feature Extraction – Extraction of Normalized Difference Indices

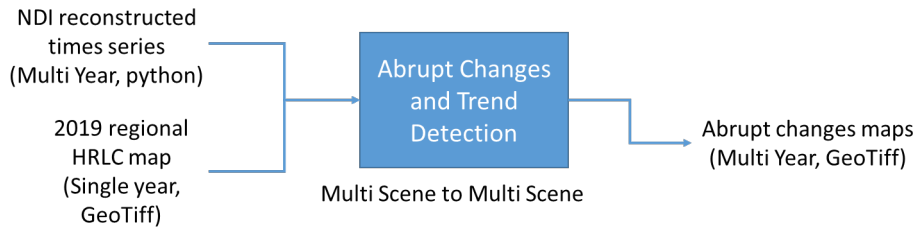
Type	Multi Scene (Time) to Multi Scene (Time)
Single Step Workflow	
Description	The Optical/SAR Feature extraction takes the whole time series and calculates normalized difference indices for 6 input bands, totaling to 15 features per each image. Input: Stack Output: Stack
Input Format	Images acquired over the whole period to be studied, i.e. 1990 – 2019. Format: Sentinel 2 L2A, SAFE product structure, Images are encoded as JPEG 2000 https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/data-formats GeoTiff (for Landsat data)
Input Example	*Not Ready*
Ancillary Data	none
Output Format	Format: GeoTiff
Output Example	*Not Ready*
Input (Matrix Size, Bands, Scenes)	Matrix: 10000 * 10000 (for 2019) and 7141 * 8021 (<2019) Bands: 10 (for S2), 8 (L7), 7 (L5) Scenes: 1 (at the same time), depends on the number of acquisitions and cloud cover (fully cloudy scenes will be discarded)
Output (Matrix Size, Bands, Scenes)	Matrix: 10000 * 10000 (for 2019) and 7141 * 8021 (<2019) Bands: 15 Scenes: 1 (at the same time), depends on the number of acquisitions and cloud cover (fully cloudy scenes will be discarded)
Hardware Needs	Number of CPUs: 6 Peak Memory consumption: 16GB RAM Working ephemeral storage: ~7GB SSD
Performance Estimation (minutes)	~6 (for a single image)
Interface Type	Command Line
Repository	https://lab.egeos-services.it/gitlab/cci-hrlc/processors/optsar-feature-extraction

3.3.3.2 Optical/SAR Time Series Regularization

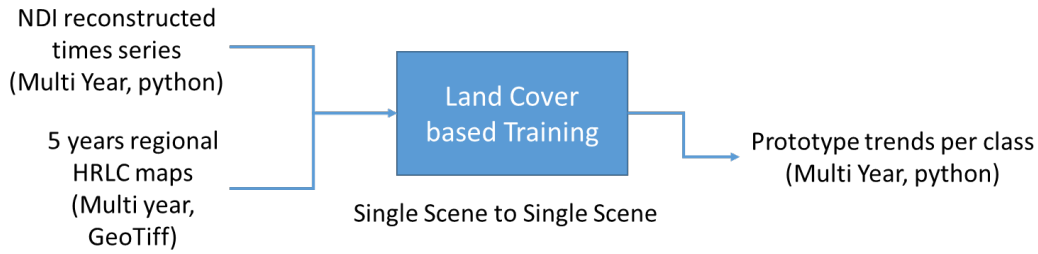
optsar-regularization	
Description	Create a daily imagery from the available time series
Type	Multi Scene (Time) to Multi Scene (Time)
Single Step Workflow	
Description	The Daily Time Series Reconstruction takes all the NDIs produced over 16 months (2 months before and two months after of the year under evaluation) and generates daily times series for each feature over 365 days (this might be done with Artifitial Neural Networks (ANN) or simpler linear regression). One year at the time and by using a common moving window of 4 months. Input: Stack Output: Stack
Input Format	GeoTiff
Input Example	*Not Ready*
Ancillary Data	Cloud Masks
Output Format	Python "*.npy"
Output Example	*Not Ready*
Input (Matrix Size, Bands, Scenes)	Matrix: 10000 * 10000 (for 2019) and 7141 * 8021 (<2019) Bands: 15 Scenes: 1 (at the same time), depends on the number of acquisitions and cloud cover (fully cloudy scenes will be discarded)
Output (Matrix Size, Bands, Scenes)	Matrix: (10000 x 10000) * 365 (for 2019) and (7141 x 8021) * 365 (<2019) Bands: 15 Scenes: 30 (one for each evaluated year)
Hardware Needs	Number of CPUs: 16 Peak Memory consumption: 16GB RAM Working ephemeral storage: 30GB SSD
Performance Estimation (minutes)	~400
Interface Type	Command Line

Repository	https://lab.egeos-services.it/gitlab/cci-hrlc/processors/optsar-regularization

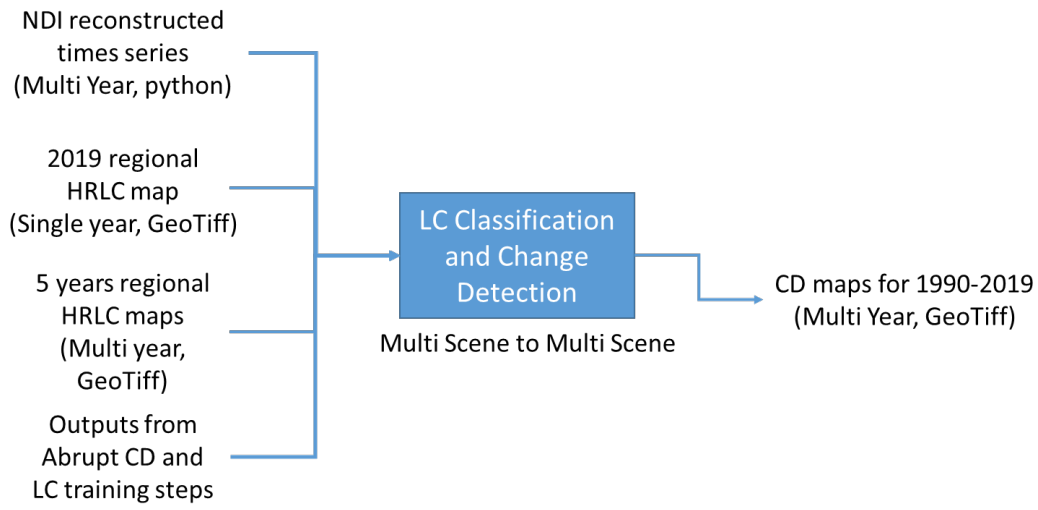
3.3.3.3 Optical/SAR Changes and Trends

optsar-changes-trend	
Description	3.1 Abrupt Changes and Trend Detection. Detection of abrupt changes happening along the 1990-2019 period.
Type	Multi Scene (Time) to Multi Scene (Time)
Single Step Workflow	
Description	Detection of the years in which an abrupt change has happened along all the years from 1990-2019 (in sets of 6 years). Uses BEAST and BFAST with R. Input: Stack Output: Stack
Input Format	Reconstructed NDIs for period 1990-2019 (in sets of 6 years) 2019 regional HRLC map Format: Python "*.npy" and GeoTiff
Input Example	*Not Ready*
Ancillary Data	none
Output Format	GeoTiff
Output Example	*Not Ready*
Input (Matrix Size, Bands, Scenes)	Matrix: (10000 x 10000) * 365 (for 2019) and (7141 x 8021) * 365 (<2019) Bands: 15 Scenes: 30 (one for each evaluated year)
Output (Matrix Size, Bands, Scenes)	Matrix: 10000 * 10000 (for 2019) and 7141 * 8021 (<2019) Bands: 2 Scenes: 6
Hardware Needs	Number of CPUs: 6 Peak Memory consumption: 16GB RAM Working ephemeral storage: 2.4GB SSD

Performance Estimation (minutes)	about 450 (assuming that only 25% of the pixels has changed)
Interface Type	Command Line
Repository	https://lab.egeos-services.it/gitlab/cci-hrlc/processors/optsar-changes-trend

trend-classification	
Description	3.2 Land Cover based Training. Generation of a matrix of prototype trends per class (CPs) every 5 years
Type	Single Scene to Single Scene
Single Step Workflow	
Description	The Land cover based training step takes several training samples from each LC classification and for each of the classes to be studied in order to create a sort of database (in the shape of a matrix) that represents the general trend behaviour over a year of each of the classe.. Input: Stack Output: Matrix of prototype trends per class (CPs) every 5 years.
Input Format	1. Reconstructed NDI for period 1999-2019. 2. 5 years regional HRLC maps Format: Python "*.npy" and GeoTiff
Input Example	*Not Ready*
Ancillary Data	None
Output Format	Python "*.npy"
Output Example	*Not Ready*
Input (Matrix Size, Bands, Scenes)	Matrix: (10000 x 10000) * 365 (for 2019) and (7141 x 8021) * 365 (<2019) Bands: 15 (for NDIs) and 2 (for HRLC maps). Scenes: 6 (one for each evaluated 5 years period)
Output (Matrix Size, Bands, Scenes)	Matrix: 15 * 365 Bands: 15 Scenes: 6

Hardware Needs	Number of CPUs: 6 Peak Memory consumption: 16GB RAM Working ephemeral storage: 18MB SSD
Performance Estimation (minutes)	about 450 (assuming that only 25% of the pixels has changed)
Interface Type	Command Line
Repository	https://lab.egeos-services.it/gitlab/cci-hrlc/processors/trend-classification

lc-change-classification	
Description	3.3 Land Cover classification and CD. Detection of the type of change that has happened.
Type	Multi Scene (Time) to Multi Scene (Time)
Single Step Workflow	
Description	The Land Cover classification and change detection step is in charge of detecting the exact type of change that has happened for each of the 30 years analyzed in the study.. Input: Stack Output: Stack
Input Format	1. Reconstructed $NDIF$ for period 1990-2019; 2. 2019 regional HRLC map or 5 years HRLC maps; 3. Output from Abrupt CD step (3.1); 4. Output from LC training step (3.2) Format: Python "*.npy" and GeoTiff
Input Example	*Not Ready*
Ancillary Data	None
Output Format	GeoTiff

Output Example	*Not Ready*
Input (Matrix Size, Bands, Scenes)	Matrix: (10000 x 10000) * 365 (for 2019) and (7141 x 8021) * 365 (<2019) Bands: 2. Scenes: 12 (6 HRLC maps, 6 abrupt CD maps)
Output (Matrix Size, Bands, Scenes)	Matrix: 10000 * 10000 (for 2019) and 7141 * 8021 (<2019) Bands: 2 Scenes: 30
Hardware Needs	Number of CPUs: 16 Peak Memory consumption: 16GB RAM Working ephemeral storage: 12GB SSD
Performance Estimation (minutes)	about 450 (assuming that only 25% of the pixels has changed)
Interface Type	Command Line
Repository	https://lab.egeos-services.it/gitlab/cci-hrlc/processors/lc-change-classification

4 Annex 1 - Tender Requirements traceability

Current requirement traceability is performed vs the Tender Requirement identified in the Tender specification:

Tender Requirement	Description	Traceability
TR-12	The prototype system to have the capacity to handle data from multiple sensors at different spatial resolutions with the capacity to deal with large high resolution data volumes (e.g. multi-temporal S1, S2, Landsat 8) and very high resolution data volumes (e.g. Pleaides, SPOT-6). Potential contributing sensors are listed in Section 9.	The system is able to handle multiple sensors as specified in the Platform Architecture and in particular in the Information Viewpoint (3.2.2) for what concerns the support of specific data and in the technology viewpoint (3.2.5) for what concerns the technology.
TR-13	The HRLC-System shall provide the scientists a configurable, flexible, agile and open HRLC CCI workflow for managing the evolution of the HRLC ECV. It	Par. 3.1 for the concept, Par. 3.2.3 for the software architecture

	shall have the following capabilities: ● Multi-sensor full mission (re-)processing for all HRLC ECV products ● Systematic, data-driven processing, allowing rapid ingestion of new data ● Configuration management (processor versions, auxiliary data, input and output data, etc.) ● Capability to integrate (existing) tools developed in different programming languages ● Capability for algorithm developers to easily trial plug in, without recompilation where practical, and execute new algorithms, new versions of algorithms or new parameterisation, for testing, intercomparison and evaluation purposes ● Support efficiently all the needs of the scientific algorithms and the new development	allowing execution of batch pipelines and easy and automated development of pipelines. Most of the functionalities are offered by the underlying technology of the orchestrator (Par. 3.2.5)
TR-15	The HRLC-System shall include data access, ingestion, product conversion tools and distribution functionality, and shall address long-term archiving of both input and output products. Both requirements shall meet the generic needs outlined in the main body of the CCI SoW.	Par. 3.1 for the concept, Par. 3.2.3 for the software architecture allowing distribution and conversion of products. Most of the functionalities of the delivery system are offered by the underlying technology (Par. 3.2.5)
TR-16	The design of the HRLC-system shall be based on experience from previous CCI projects (e.g. LandCover_cci and Fire_cci) and from other research and operational processing schemes (e.g. Copernicus).	The design is taking into account of several achievements from previous projects including the underlying technology (Max-ICS) and several other developments.
TR-17	The HRLC-System shall have data access interfaces to handle efficiently, and in a standardised fashion, the data streams from Sentinel 1 and 2. The interfaces shall support the possibility of cross-ECV synergies.	The system is able to ingest data both from official ESA catalogues/systems like scihub/cophub

TR-18	The contractor shall link with the CCI portal and data analysis/visualisation tools available through the CCI Toolbox	The system provides OGC standard interfaces that can be directly used by the CCI Toolbox

5 Annex 2 - Questionnaire to DIAS providers

Item	Questions to DIAS provider
Data access	Which of the required EO data are available? - Sentinel 2: L2A Data are required for AOIs outside of Europe for 2018, On CopernicusHub approximately 570k L1C products are available, out of these for approx. 20k the corresponding L2A. - Landsat: Data are required for AOIs from 1990 until 2017 (approx. 55k L5 , 75k L7 and 30k L8 scenes) - Sentinel 1: SLC Data are required for AOIs outside of Europe for 2018 (On CopernicusHub approximately 33k scenes) (More detailed information about AOIs and required scene IDs can be provided) What options are there to check data availability by oneself? If the above mentioned data are not available, what can you offer to help with: - Retrieving - Processing - Hosting The Sentinel 2 data amount to 300-400TB, The Landsat data to 50-60TB and the Sentinel 1 SLC data to 200TB Which ancillary data are accessible? How can the satellite data be accessed within the cloud infrastructure (mounted drive, client software, HTTP)? In which format are data stored (SAFE, zip,object storage...)? Is metadata-DB available for quick queries? What up-/download rates can be ensured ?
Processing infrastructure	Do you provide VMs with an EC2 like API? (Multinode) scalability: What is the maximum no. of nodes available within a given time frame? What kind of storage is available (e.g. S3)? Which levels are available on S3 storage (hot to cold)? With which level of service (time to have data available from cold storage)? Do you also offer GPU processing capacities? What predefined VMs are available and what relevant software do they include related to Remote Sensing Are there limitations on externally available network addresses or URLs? What is storage capacity and flexibility to increase/decrease capacity? How is this process organized (e-mail or fully automatic interface)? What is processing capacity and flexibility to increase/decrease capacity? How is this process organized (e-mail or fully automatic interface)? What additional services does your cloud infrastructure offer (Queues, APIs, Containers etc.)?
	What is the estimated downtime / guaranteed uptime?

Organisational aspects	Is it possible to get free access to the DIAS infrastructure for evaluation and testing? What is the current level of system implementation (which functions are available and which planned)? Is the system ready for integration of 3rd party processing chains (If not, is it foreseen, ETC?) How is the system documented? Manuals available? Direct support via personal contact? What is the availability time of personal support? What kind of SLAs are offered?
What is cost scheme for:	System access/subscription? Are also Spot/Preemptible modes available? CPU hours on VMs optimized for computation? Can you provide a list of compute optimized VMs and their pricing? See ComputeCost tab for template. Consider the case of 100GB storage. Storage & data access? GPU processing? Traffic (up-/download; internal/external)? Support? Services (metadata-DB, queuing,…)? Cost policy: fixed scheme vs. pay what you consume? Cost transparency: how close to real-time is cost reported? Can cost limits be defined?