OGC® DOCUMENT: 22-031R1

External identifier of this OGC® document: http://www.opengis.net/doc/PER/T18-D040

TESTBED-18: REPRODUCIBLE FAIR BEST PRACTICES ENGINEERING REPORT

ENGINEERING REPORT

PUBLISHED

Submission Date: 2022-11-23
Approval Date: 2023-01-02
Publication Date: 2023-01-03
Editor: Pedro Gonçalves

CONTENTS

LIST OF FIGURES

I EXECUTIVE SUMMARY

This OGC Testbed 18 Engineering Report (ER) is a response to the challenge of scientific algorithm portability and reproducibility. The ER defines a set of best practices on how Exploitation Platforms can promote and follow the FAIR Data Principles and support full reproducibility for Earth Observation data processing applications. The best practice objective is to ensure that Exploitation Platforms promote and follow Open Science principles supporting the long-term reproducibility of results associated with publications and the provision of reliable mechanisms to measure the scientific impact of the usage of the EO data, Platforms, and scientific algorithms.

The FAIR Data Principles are a set of guiding principles to make data findable, accessible, interoperable, and reusable. These principles provide the initial guidance for the Exploitation Platform data producers and data publishers to promote maximum use of research data as a framework for fostering and extending a Platform's data services.

The starting point for this activity is the OGC Best Practice for Earth Observation Application Package (OGC 20-089). This set of best practices already supports EO application developers that want to adapt and package their existing algorithms written in a specific language to be reproducible, deployed and executable in different platforms as processing services. This ER identifies a set of requirements necessary for the execution of the services ensuring that they are committed to standards of findability, accessibility, interoperability, and reusability.

The focus of the ER is on data processing EO application services that deal with Earth Observation products or specialized GeoDataCubes and includes guidance at the Application, Package, and Platform levels. The Earth Observation Application Package Best Practice is extended with the use of CodeMeta, a minimal metadata schema for science software and CWLProv, a workflow definition standard designed to enable portability, interoperability, and reproducibility of analyses between platforms.

Recommended future work, building upon the described activities, should focus on validating the proposed extension of the Application Package with additional applications, use cases, and scenarios to ensure the adoption by the Exploitation Platforms of the FAIR Data Principles and support the full reproducibility of the EO Workflows.

II KEYWORDS

The following are keywords to be used by search engines and document catalogues.

Earth observation, application, application package, exploitation platform, portability, reproducibility, unique identifier, FAIR

III SECURITY CONSIDERATIONS

No security considerations have been made for this document.

IV SUBMITTERS

All questions regarding this document should be directed to the editor or the contributors:

NAME	ORGANIZATION	ROLE
Pedro Goncalves	Terradue	Editor
Fabrice Brito	Terradue	Contributor
Josh Lieberman	OGC	Contributor
Gérald Fenoy	GeoLabs	Contributor

V ABSTRACT

The OGC Testbed-18 initiative included a discussion exploring the future of open science and building energy interoperability with the task of developing a set of best practices to make the data processing services of Exploitation Platforms both reproducible and follow the FAIR data principles.

Portability and reproducibility are key factors for the long-term scientific impact of Earth Observation (EO) data processing applications provided by Exploitations Platforms. The EO application developers lack the tools and guidance to preserve all the elements, algorithms, software, and data resources used to produce the results. Without these elements, reproducibility becomes resubmission within the platform and only while the same platform resources such as data are preserved and available.

This Testbed 18 Engineering Report defines a list of requirements and respective best practices to support reproducible Earth Observation science covering the different resources of the Earth Observation Exploitation Platforms such as publications, data, services, products, information, software, or computing environments.

1

SCOPE

1 SCOPE

This Testbed 18 Engineering Report defines the Best Practice to support reproducible Earth Observation science in terms of data inputs, algorithm packaging, and execution on platforms and data outputs. The objective is to guide developers and platform providers to support full reproducibility for Earth Observation data processing algorithms.

Section 3 introduces the current gaps between the data processing EO applications of Exploitation Platforms and the Open Science principles such as FAIR and how these gaps hamper the long-term scientific impact of Exploitations Platforms.

Section 4 brings reproducible research scenarios and identifies the different requirements for EO Applications in terms of findability, accessibility, interoperability, and reusability.

Section 5 identifies the best practices to make the data processing services of Exploitation Platforms FAIR and reproducible and identifies a set of actions necessary for the execution of the services ensuring that they are committed to the FAIR principles.

Section 6 presents the activity conclusions and future activities. == Terms and definitions

1.1. Application

A self-contained set of operations to be performed, typically to achieve a desired data manipulation, written in a specific language (e.g., Python, R, Java, C++, C#).

1.2. Application Package

An independent and self-contained representation of an Application, providing executables, metadata, and dependencies such that it can be deployed to and executed within an Exploitation Platform.

1.3. CodeMeta

A minimal metadata schema for science software and code in JSON and XML. The goal of CodeMeta is to create a concept vocabulary that can be used to standardize the exchange of software metadata across repositories and organizations.

1.4. Computer Platform

The Platform providing the computational resources for the execution of the Application.

1.5. Container

A container is a standard unit of software that packages up code and all its dependencies so that includes everything needed to run an application: code, runtime, system tools, system libraries, and settings.

1.6. Copernicus

Copernicus is the European Union’s Earth observation program coordinated and managed for the European Commission by the European Union Agency for the Space Programme in partnership with the European Space Agency (ESA) of the EU Member States.

1.7. CWLProv

A profile to define how to record provenance of a workflow run (e.g., CWL) captured as a research object using Linked Data standards.

1.8. DataCite

DataCite is a leading global non-profit organization that provides persistent identifiers (DOIs) for research data and other research outputs. Organizations within the research community join DataCite as members in order to assign DOIs to all their research outputs.

1.9. Digital Object Identifier System

The Digital Object Identifier System provides a technical and social infrastructure for the registration and use of persistent interoperable identifiers for use on digital networks.

1.10. Envisat

Envisat is a large, inactive Earth-observing satellite which is still in orbit and is now considered space debris. Operated by the European Space Agency, it was the world's largest civilian Earth observation satellite.

1.11. Exploitation Platform

An on-line system made of products, services, and tools for exploitation of data.

1.12. GeoDataCubes

A multi-dimensional ("n-D") array of values, with emphasis on the fact that "cube" is just a metaphor to help illustrate a data structure that can in fact be 1- dimensional, 2-dimensional, 3-dimensional, or higher-dimensional.

1.13. Landsat

The NASA/USGS Landsat Program provides the longest continuous space-based record of Earth's land in existence. Landsat data provides information essential for making informed decisions about Earth's resources and environment.

1.14. npm

npm is a package manager for the JavaScript programming language.

1.15. Processing Result

The Products produced as output of a Processing Service execution.

1.16. Processing Service

A non-interactive data processing provided as a service by a platform that has a well defined set of input data types and input parameterization producing Processing Results with a well defined output data type.

1.17. Processing Software

A set of predefined functions that interact to achieve a result. For the exploitation platform, it comprises interfaces to derive data products from input data conducted by a hosted processing service execution.

1.18. Products

Earth Observation data (commercial and non-commercial) and value-added data. It is assumed that the Exploitation Platform provides the data access mechanisms for an existing supply of Earth Observation Products.

1.19. Resource Description Framework

Standard originally designed as a data model for metadata that has come to be used as a general method for description and exchange of graph data.

1.20. Sentinel-2

A European wide-swath, high-resolution, multi-spectral imaging mission.

1.21. Software Heritage

Software Heritage is an organization that allows for the preservation, archiving, and sharing of source code for software.

1.22. Spatiotemporal Asset

Any file that represents information about the earth captured in a certain space and time.

1.23. Zenodo

A platform and repository developed and operated by CERN to enable the sharing of research data and outputs.

1.24. Abbreviated terms

API	Application Programming Interface
CERN	European Organization for Nuclear Research
CWL	Common Workflow Language
DAG	Directed Acyclic Graph
DOI	Digital Object Identifier
ER	Engineering Report
ERS	European Remote Sensing satellite
ESA	European Space Agency
FAIR	Findable, Accessible, Interoperable, Reusable
ICT	Information and Communications Technology
INRIA	French Institute for Research in Computer Science and Automation
JSON	JavaScript Object Notation

NASA	National Air and Space Administration
OGC	Open Geospatial Consortium
RDF	Resource Description Framework
REST	Representational state transfer
STAC	Spatio-Temporal Asset Catalog
SWG	Standards Working Group
SWHID	Software Heritage Persistent Identifier
UNESCO	United Nations Educational, Scientific and Cultural Organization
URL	Uniform Resource Locator
USGS	United States Geological Survey
WG	Working Group
XML	Extensible Markup Language

2

NORMATIVE REFERENCES

2 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

Commonwl.org: Common Workflow Language Specifications, https://w3id.org/cwl/

Arliss Whiteside Jim Greenwood: OGC 06-121r9, *OGC Web Service Common Implementation Specification*. Open Geospatial Consortium (2010). https://portal.ogc.org/files/?artifact id=38867.

Benjamin Pross, Panagiotis (Peter) A. Vretanos: OGC 18-062r2, *OGC API — Processes — Part 1: Core*. Open Geospatial Consortium (2021). https://docs.ogc.org/is/18-062r2/18-062r2.html.

Groth, P., Moreau, L.: PROV-overview. An overview of the PROV family of documents. (2013).

Radiant Earth Foundation: SpatioTemporal Asset Catalog specification, https://stacspec.org

STAC Scientific Citation Extension Specification, https://github.com/stac-extensions/scientific

Schema.org: http://schema.org/docs/schemas.html

J. Kunze, J. Littman, E. Madden, J. Scancella, C. Adams: IETF RFC 8493, *The BagIt File Packaging Format (V1.0)*. RFC Publisher (2018). https://www.rfc-editor.org/info/rfc8493.

3

INTRODUCTION

3 INTRODUCTION

For almost four decades, “Earth Observation” (EO) satellites developed and/or operated by Space Agencies have provided a wealth of data. In past years, the Sentinel missions, along with the Copernicus Contributing Missions, Earth Explorers, and new commercial missions provide new ways to perform the routine monitoring of our environment at the global scale delivering an unprecedented amount of data. This expanding operational capability of global monitoring from space, combined with data from long-term EO archives such as ERS, Envisat and Landsat, in-situ networks, and models provide users with unprecedented insight into how our oceans, atmosphere, land, and ice operate and interact as part of an interconnected Earth System.

To support the use of this data deluge, several Platforms for the Exploitation of Earth Observation data are being developed and operated. These Earth Observation (EO) Exploitation Platforms are collaborative virtual work environments providing access to EO and value-added products together with the tools, processors, and ICT resources required to work with them. The fundamental principle of these platforms’ operations concept is to move the users to the data and tools. Users access a platform work environment providing the data, tools, and resources required, as opposed to downloading, replicating, and exploiting data ‘at home’.

Unfortunately, most Platforms are running in closed silos and the scientific impact of their usage is hard to measure as most of the derived publications do not include clear references to the EO applications executed, their algorithms, software, applications, and data resources used to produce the results.

Platforms are missing the tools to make modern scientific results reproducible and able to pass the scientific knowledge over to the next generation of researchers in three main reference areas: the scientific articles that describe the results, the data sets used or produced, and the software that embodies the logic of the data transformation.

Furthermore, the reproducibility of the EO applications is often limited to the boundaries of the Platform itself (e.g., one can resubmit the same execution) but this is only guaranteed while the same Platform resources are available.

In fact, the gaps between the data processing services of Exploitation Platforms and the Open Science principles (FAIR) hamper the long-term reproducibility of results associated with publications and provide reliable mechanisms to measure the scientific impact of the usage of the Platforms.

The best practices described in this ER are a response to this challenge and provide guidance on how to make the data processing services of Exploitation Platforms not only reproducible but also FAIR. The starting point is the OGC Best Practice for Earth Observation Application Package (OGC 20-089) that already supports developers in adapting and packaging their existing algorithms written in a specific language to be reproducible, deployable, and executable in different platforms. This ER identifies a set of additional actions necessary for the execution of the services ensuring that they are committed to standards of findability, accessibility, interoperability, and reusability.

The FAIR principle states that resources should meet standards of findability, accessibility, interoperability, and reusability. The best practices defined in the ER ensure that these principles are applied to the EO application resources that were used to generate results that can be referenced in a scientific publication.

In the Exploitation Platforms context, the resources are publications, data, services, products, information, software, or computing environments. Drilling from the FAIR principles, this ER extracts the requirements of each of these elements and provides guidance on how to ensure reproducible Earth observation science in terms of application package, application container, application input data, and application output.

4

REPRODUCIBLE RESEARCH

4 REPRODUCIBLE RESEARCH

EO Exploitation Platform users have different expectations and goals that can be addressed through the definition of two main roles: consumers and developers.

Consumers want to access existing and available services within the platform to acquire new knowledge. This knowledge may be consumed immediately or delivered to external downstream services that may be scientific, research oriented, or commercial in nature. Conversely, developers aim at integrating an existing algorithm or application, improving a workflow, or developing a new data processing workflow.

When developers bring a new data processing service to a platform, they are in fact integrating their own software, written in a specific programming language, as a containerized application (or a set of containerized applications), to be exposed as a new processing service. To achieve this, they will follow the deploy, test, validate, and publish steps described below.

- Prepare one or more container images containing the execution dependencies of the software.
- Prepare the application package manifest orchestrating the container(s).
- Test the application package by deploying and executing it in a hosted test environment with access to platform data for testing.
- Register the application package as a new processing service, supported by ancillary information including metadata and processor user manual.
- The new processing service is discoverable in the platform and made available to a community of Consumers according to specific access rights.

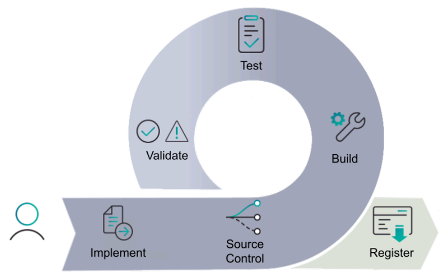


Figure 1 — The Development Cycle From Implementation to Registration

When Consumers discover and execute data processing services with specific input parameters, they will follow a set of steps moving from discovery, submission, monitor, and result steps described as follows.

- Discover the on-demand Processing Services available on the Platform.
- Retrieve Processing Service complementary information such as the description, processing requirements and costs, terms and conditions, and license.

- Define the Processing Service input parameters with access to data catalogs to discover compatible data.
- Optionally obtain an estimation of the execution costs and/or time based on the input parameters.
- Monitor the Processing Service status upon submission.
- Access and visualize results obtained.
- Publish and share the results within a given community.

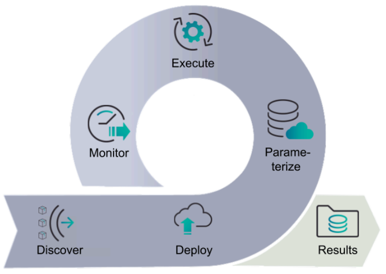


Figure 2 — The Production Cycle From Service Discovery to Results Generation

Both of these actors may be experts in their domain or require specific filters and visualization tools in order to acquire and consume the information obtained. Most importantly, a service developer may be also a consumer of a service and vice-versa. To achieve reproducibility in an Exploitation Platform, both use cases must follow a set of best practices that can be mapped to the FAIR Data Principles.

The FAIR Data Principles were developed as guidelines to enhance the scientific data reusability and presented a new view on how APIs could enhance the reproducibility of results and traceability of workflow and processes, paving the way towards a new generation of scientific publications.

Following the four principles —Findability, Accessibility, Interoperability, and Reusability— data producers and publishers maximize the added-value of scholarly digital publishing. Most importantly, the principles were not only applied to data but were also targeting algorithms, tools, and workflows that created the data. The Platforms for the Exploitation of Earth Observation (EO) operate applications and workflows that use a combination of pre-processed data, fusion, and analytical functions. The platform resources are thus publications, data, services, products, information, software, or computing environments. The following section identifies the technical requirements that a Platform needs to address so that elements essential to the elaboration of data are described and ensure transparency, reproducibility, and reusability.

In an EO Exploitation the FAIR Guiding Principles are mapped to requirements for findability, accessibility, interoperability, and reusability.

4.1. Findability Requirements

F1 — EO Application resources shall be assigned with a globally unique and persistent identifier (e.g., DOI).

F2 — EO Application data resources shall be described with rich metadata.

F3 — EO Application resources metadata shall be compliant with DataCite's Metadata Schema minimum and recommended terms potentially with a few additional enrichments.

F4 — EO Application resources metadata clearly and explicitly shall include the identifier of the data being described.

F5 — The DOI shall be a top-level and a mandatory field in each record of the EO Application resources.

F6 — EO Application resources (meta)data shall be registered or indexed in a searchable resource.

F7 — EO Application resources metadata of each record shall be indexed and searchable.

4.2. Accessibility Requirements

A1 — EO Application resources metadata shall be retrievable by their identifier using a standardized communications protocol.

A2 — EO Applications resources metadata for individual records as well as record collections shall be harvestable using the standard harvesting protocols by the record identifier and the collection name.

A3 — EO Applications resources metadata shall be also retrievable through public REST APIs.

A4 — Protocols for information retrieval shall be open, free, and universally implementable.

A5 — Protocols for information retrieval shall allow for an authentication and authorization procedure, where necessary.

A6 — EO Applications resources metadata shall be publicly accessible and licensed under public domain. No authorization is ever necessary to retrieve the resource's metadata.

A7 — EO Applications resources metadata shall be accessible, even when the data are no longer available.

4.3. Interoperability Requirements

I1 — EO Applications resources metadata shall use a formal, accessible, shared, and broadly applicable language for knowledge representation.

I2 — EO Applications resources metadata shall use vocabularies that follow FAIR principles.

I3 — EO Applications resources metadata shall include qualified references to other metadata entries (metadata is qualified by a resolvable URL).

4.4. Reusability Requirements

R1 — EO Applications resources metadata shall be richly described with a plurality of accurate and relevant attributes.

R2 — Each EO Applications resources metadata record shall contain a minimum of DataCite's mandatory terms, with optionally additional DataCite recommended terms.

R3 — EO Applications resources metadata shall be released with a clear and accessible data usage license.

R4 — EO Applications resources metadata shall contain detailed provenance.

R5 — EO Applications resources metadata shall be traceable to the user that published the metadata.

R6 — EO Applications resources metadata may optionally describe the original authors of the published work.

R7 — EO Applications resources metadata shall meet domain-relevant community standards. == Best Practices

This section identifies the best practices to make the data processing services of Exploitation Platforms FAIR and reproducible. The starting point is the OGC Best Practice for Earth Observation Application Package (OGC 20-089) that already supports developers that want to adapt and package their existing algorithms written in a specific language to be reproducible, deployable, and executable in different platforms. This Testbed 18 ER identifies a set of additional actions necessary for the execution of the services ensuring that they are committed to standards of findability, accessibility, interoperability, and reusability.

In the context of the EO Exploitation Platforms, the resources are publications, data, software, results, and computing environments that have different (and independent) life cycles. As such this ER is also divided in terms of Application Development, the Application Packaging, and their instantiation when hosted in a Platform.

4.5. Application

Earth observation (EO) applications use a combination of input data, fusion, and analytical functions. The application itself is written in a given coding language(s), uses specific software libraries, and is delivered in a container image with all the necessary software, libraries, and configuration files. In terms of scientific publications, if researchers reference their software in their articles, they normally use many different types of locations: web pages, development repositories, publication annexes, etc. In a typical development — deployment — production of an Application in an Exploitation Platform scenario, the application would be black box identified with name, version, and whatever additional metadata the developer would see fit to provide.

4.5.1. Source Code

In a Reproducible FAIR Workflow scenario, ensuring that the code behind the application is uniquely identified and traceable is necessary. Even if software configuration management systems have become a common tool for tracking and controlling changes in the software, the FAIR publication of applications also needs methods for software citation, software retrieval, and long-term preservation. This is achieved with the issuing of a persistent identifier.

The persistent identifier provides a long-lasting reference to source code and is usually understood as an actionable and accessible reference over the Internet that retrieves the necessary files. As such, the persistent component is dependent on the service commitment to resolve the identifier and the dedicated storage lifespan.

Several initiatives provide this persistent dual functionality of identification and archivation. This Best Practice identifies one solution with Software Heritage. Software Heritage is a non-profit multi-stakeholder initiative unveiled in 2016 by the French Institute for Research in Computer Science and Automation (Inria) and supported by UNESCO. The mission of Software Heritage is to collect, preserve, and share all software that is publicly available in source code form, with the goal of building a common shared infrastructure at the service of industry, research, culture, and society.

Software Heritage provides *SoftWare Heritage persistent IDentifiers* (SWHIDs) and ensures preservation of the software source code by crawling code hosting platforms, like GitHub, GitLab.com, or Bitbucket, and package archives, like npm or PyPI, ingested into a special data structure, a Merkle DAG, that is the core of the archive.

The use of Merkle DAG is a generalization of a hash list and a hash chain similar to a Merkle tree (i.e. a tree of hashes) but while a Merkle tree connects transactions by sequence, the Merkle-DAG connects transactions by hashes. In a Merkle-DAG, addresses are represented by a Merkle hash. The use of Merkle DAGs allows efficient and secure verification of the contents of a large data structure and verifies any kind of data stored, handled, and transferred in and between computers.

To obtain a SWHID, the code must be hosted on a publicly accessible repository (Github, Bitbucket, a GitLab instance, an institutional software forge, etc.) using one of the version control systems supported by Software Heritage (Subversion, Mercurial, and Git).

The top level of the source code tree most contain a linked data metadata file with the CodeMeta vocabulary (i.e., codemeta.json). This file helps indexing the source code in the Software Heritage archive and provides an easy way to link to other related research outputs.

CodeMeta is a minimal metadata schema for science software and code with the goal to create a concept vocabulary that can be used to standardize the exchange of software metadata across repositories and organizations. With the CodeMeta schema it is possible to map between the different services (GitHub, figshare, or Zenodo) and compare software metadata used across multiple repositories. The software metadata concepts are arranged into a JSON-LD context for serialization.

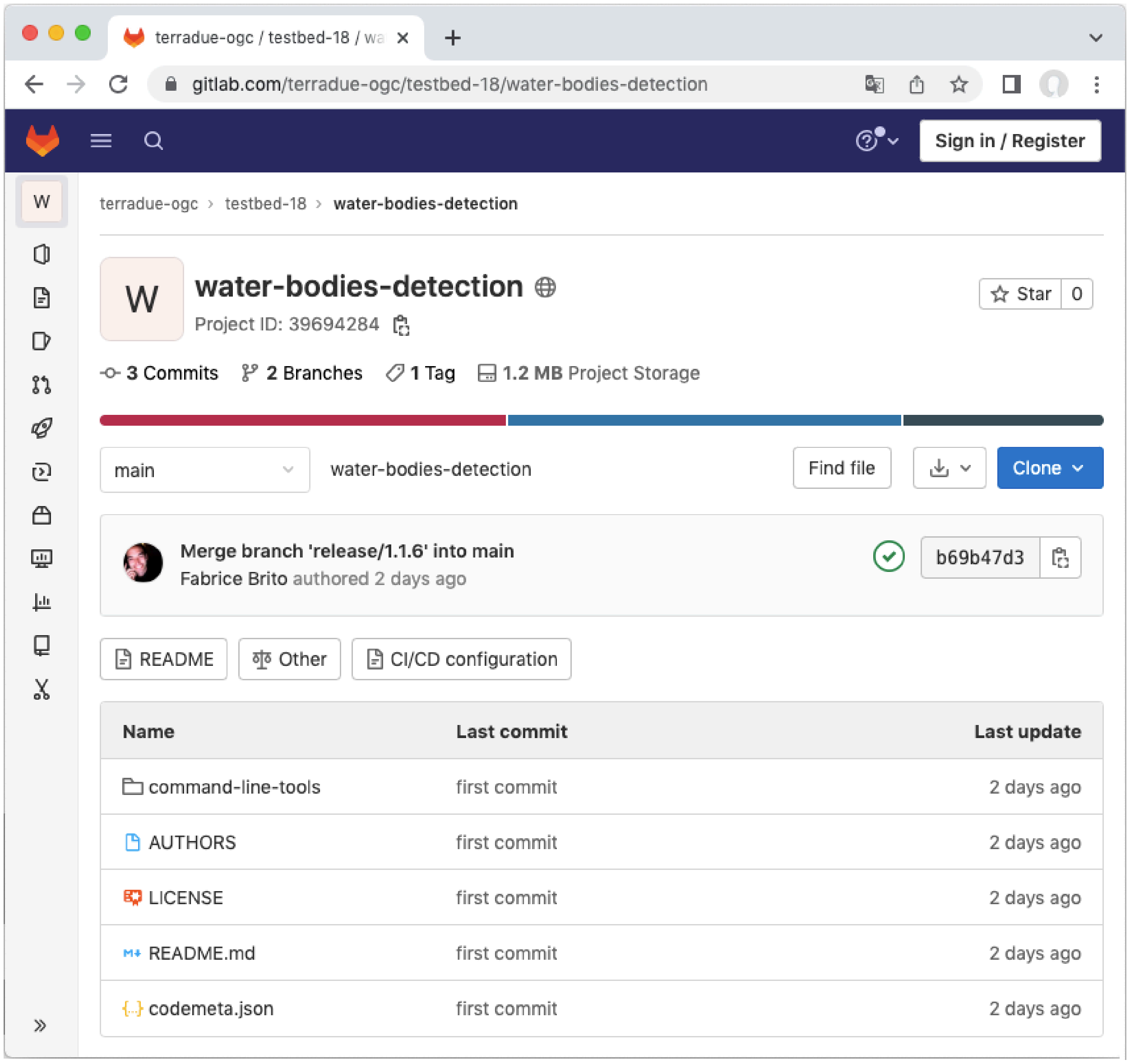
CodeMeta developed the translations from the different vocabularies. The CodeMeta vocabulary is an extension of the *SoftwareApplication* and *SoftwareSourceCode* classes found in the vocabulary of the Schema.org initiative [schema]. Metadata information conformant to the CodeMeta vocabulary can be represented in JSON format, named codemeta.json, like the example below with the information for Water Bodies Detection application.

```
{
    "@context": "https://doi.org/10.5063/schema/codemeta-2.0",
    "@type": "SoftwareSourceCode",
    "license": "https://spdx.org/licenses/CC-BY-NC-SA-4.0",
    "dateCreated": "2022-09-01",
    "datePublished": "2022-09-25",
    "dateModified": "2022-09-25",
    "name": "Water Bodies Detection",
    "version": "1.0.0",
    "description": "The Water Bodies Detection is an Application that uses
the NDWI index and the Otsu threshold to detect water bodies using Sentinel-2
data",
    "developmentStatus": "concept",
    "funder": {
        "@type": "Organization",
        "name": "OGC Testbed 18"
    },
    "keywords": [
        "NDWI"
    ],
    "programmingLanguage": [
        "Python",
           ],
    "softwareRequirements": [
        "container runtime"
    ],
    "author": [
        {
            "@type": "Person",
            "givenName": "Jane",
            "familyName": "Doe",
            "email": "jane.doe@acme.earth"
        },
        {
            "@type": "Person",
            "givenName": "John",
            "familyName": "Doe",
            "email": "john.doe@acme.earth"
        }
    ]
}
```

For convenience and readability, the following files also need to be present on the top level of the source tree.

- README containing a description of the software (name, purpose, pointers to website, documentation, development platform, contact, and support information, etc.).
- AUTHORS, a list of all the persons credited for the software.
- LICENSE, the project license terms.

The following image shows the contents of a repository with the needed application elements for it to be uniquely identified and traceable with *Software Heritage*.


terradue-ogc / testbed-18 / wa
gitlab.com/terradue-ogc/testbed-18/water-bodies-detection
Sign in / Register
terradue-ogc > testbed-18 > water-bodies-detection
water-bodies-detection
Project ID: 39694284
Star 0
3 Commits
2 Branches
1 Tag
1.2 MB Project Storage
main
water-bodies-detection
Find file
Clone
Merge branch 'release/1.1.6' into main
Fabrice Brito authored 2 days ago
b69b47d3
README
Other
CI/CD configuration

Name	Last commit	Last update
command-line-tools	first commit	2 days ago
AUTHORS	first commit	2 days ago
LICENSE	first commit	2 days ago
README.md	first commit	2 days ago
codemeta.json	first commit	2 days ago

Figure 3 — Gitlab repository for the Water Bodies Detection application with the needed application elements for it to be uniquely identified and traceable by *Software Heritage*.

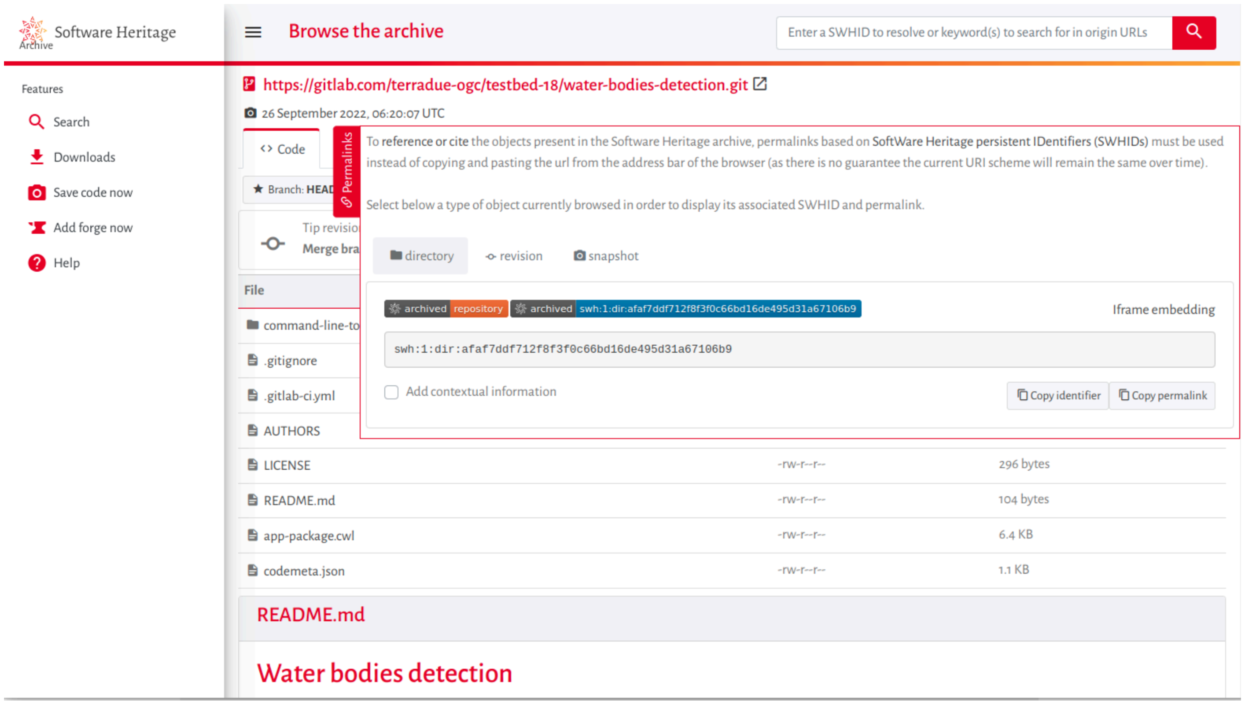
Once the code repository is properly prepared it can be submitted to Software Heritage to be archived with its full development history. If the repository is hosted on one of the major known forges the process takes just a few hours. The submission process can be done through the Software Heritage web site or by requesting the archival programmatically using the dedicated Software Heritage API entry point.

After submitting the software repository for archival at the Software Heritage it obtains the following identifier:

```
swh:1:dir:afaf7ddf712f8f3f0c66bd16de495d31a67106b9;origin= https://gitlab.com/terradue-ogc/testbed-18/water-bodies-detection.git; visit=swh:1:snp:79dd9bcb0327d0cd75abceccbb7bccbe51e889a3; anchor=swh:1:rev:b69b47d3dc663031cee7793813e01a2db6f582e7
```

and the permanent link:

https://archive.softwareheritage.org/swh:1:dir:afaf7ddf712f8f3f0c66bd16de495d31a67106b9; origin=https://gitlab.com/terradue-ogc/testbed-18/water-bodies-detection.git; visit=swh:1:snp:79dd9bcb0327d0cd75abceccbb7bccbe51e889a3; anchor=swh:1:rev:b69b47d3dc663031cee7793813e01a2db6f582e7



https://gitlab.com/terradue-ogc/testbed-18/water-bodies-detection.git

26 September 2022, 06:20:07 UTC

To reference or cite the objects present in the Software Heritage archive, permalinks based on SoftWare Heritage persistent IDentifiers (SWHIDs) must be used instead of copying and pasting the url from the address bar of the browser (as there is no guarantee the current URI scheme will remain the same over time).

Select below a type of object currently browsed in order to display its associated SWHID and permalink.

```
swh:1:dir:afaf7ddf712f8f3f0c66bd16de495d31a67106b9
```

File		
command-line-to		
.gitignore		
.gitlab-ci.yml		
AUTHORS		
LICENSE	-rw-r--r--	296 bytes
README.md	-rw-r--r--	104 bytes
app-package.cwl	-rw-r--r--	6.4 KB
codemeta.json	-rw-r--r--	1.1 KB

README.md

Water bodies detection

Figure 4 — Application Package on Software Heritage

4.5.2. Container

Together with the source code, a Reproducible FAIR Workflow scenario also requires that the application container(s) is(are) uniquely identified and traceable. Initially thought of as a straightforward responsibility of the container registry, it became clear that the common practices in well-known container formats like Docker do not guarantee the unique identification and traceability of the container.

For example, when building container images the developer tags the image with a specific name such as:

```
docker build -t docker.terradue.com/crop:1.1.6 .
```

And, when performing a Docker *pull* that tag is used to retrieve the required container. However, if by mistake the developer or continuous integration tool builds another container image with the same tag but with changed content, the reproducibility cycle would be broken.

To address this issue a deterministic approach is required using the sha256 signature of the container which will act as an immutable identifier. To obtain this with Docker would require execution of the following command:

```
docker inspect docker.terradue.com/crop:1.1.6 | jq '.[0]["RepoDigests"][0]'
```

This returns a Docker identifier guaranteed to be unique.

```
docker.terradue.com/crop@sha256:ec2d8e71ab5834cb9db01c5001bde9c3d6038d0418ad085
726b051b4359750e1
```

This approach ensures a deterministic identification of a container image and build instance. In fact, two separate builds from the same Docker file result in differing hash values. === Package

As seen in the previous section, developers build a container image with the application's command line tool(s) and respective runtime environments and then publish the container image on a repository.

Subsequently, following the OGC Best Practices for EO Application Packages (OGC 20-089), the developers create an Application Package encoded in a Common Workflow Language (CWL) document to describe the data processing applications, provide information about the parameters, software items, executables, dependencies, and metadata. Aside from the role of the CWL in packaging existing applications, the Application Package can orchestrate the execution of an application of several pre-packaged processing steps. In this scenario the Application Package combines and/or chains multiple processing services, potentially offered by different developers/organizations, in parallel or sequentially.

The Application Package documents the orchestration steps to invoke the command line tool(s) included in the image(s). This document ensures that the application is fully portable among all supporting processing scenarios and supports automatic deployment according to the OGC API - Processes (OGC 18-062).

The OGC Best Practices for EO Application Packages (OGC 20-089) recommends that to enrich the application package with new concepts, these should originate from schema.org and be linked with their RDF encoding. While OGC 20-089 only enforces the *version* element as a mandatory, it already suggest several additional elements that are key for the Reproducible FAIR Workflows scenario such as the following.

- *author*: The main author of the Application Package - https://schema.org/author.
- *citation*: A citation or reference to a publication, web page, scholarly article, etc. https://schema.org/citation.

- *codeRepository*: Link to the repository where the Application code is located (e.g., SVN, github). https://schema.org/codeRepository.
- *contributor*: A secondary contributor to the Application Package https://schema.org/contributor.
- *dateCreated*: The date on which the Application Package was created. https://schema.org/dateCreated.
- *keywords*: Keywords used to describe this application. Multiple entries in a keywords list are delimited by commas. https://schema.org/keywords.
- *license*: An URL to the license document that applies to this application. https://schema.org/license.
- *releaseNotes*: Description of what changed in this version. https://schema.org/releaseNotes.

All these elements can be obtained from the CodeMeta vocabulary used in the previous section to characterize the Application (i.e., codemeta.json) and can be directly added when defining the Application Package.

In the Reproducible FAIR Workflow scenario, it is necessary to ensure that the Application Package is uniquely identified and traceable. As such, this ER recommends the assignment of a DOI to the application package and extends the metadata section of the Application Package CWL with:

- *sameAs*: URL of a reference Web page that unambiguously indicates the item's identity. https://schema.org/sameAs.

For the previous example of the Water Bodies Detection application, the metadata section of the Application Package encoded in CWL would become:

```
cwlVersion: v1.0
$graph:
...
$namespaces:
  s: https://schema.org/
s:author:
- class: s:Person
  s:affiliation: Planet Earth
  s:email: john.doe@somedomain.org
  s:name: Doe, John
s:contributor:
- class: s:Person
  s:affiliation: Planet Earth
  s:email: jane.doe@somedomain.org
  s:name: Doe, Jane
s:sameas: https://doi.org/10.5072/zenodo.1107209
s:softwareVersion: 1.1.6
schemas:
- http://schema.org/version/9.0/schemaorg-current-http.rdf
```

4.6. Platform

An Earth Observation Exploitation Platform provides interfaces, processing functions, tools, and processing services used individually or in workflows. Developers can test and execute their own applications, register them into the platform, and make them available for exploitation by other users individually or in their own workflows.

An Earth Observation Exploitation Platform provides not only a mechanism to deploy the Application Package but also the mechanism to execute the process defined by the Application Package by instantiating a new request with specific parameters. Interfaced with Cloud Computing providers, an Earth Observation Exploitation Platform executes the data processing tasks requested by users and retrieves the information produced for delivery back to the processing requester.

After defining new requirement classes brought by the Reproducible FAIR Workflows scenario to the development of the Application and the corresponding Application Package as defined by OGC 20-089, defining how this scenario affects the Exploitation Platform is also necessary.

The Application Package is itself a Common Workflow Language document, a workflow definition standard designed to enable portability, interoperability, and reproducibility of analyses between workflow execution platforms. As such, the Application Package can capture the Workflow software environment by freezing and packaging the runtime environment to encompass all the software components and their dependencies used in an analysis. The CWL object model supports comprehensive recording and capture of information for the EO application design, orchestration, and execution. This can subsequently be published as structured information alongside any resultant analysis using that application.

The reproducibility of the application deployment is intrinsically guaranteed by the draft OGC API — Processes — Part 2: Deploy, Replace, Undeploy (OGC 20-044), but there is no mechanism to make the actual application request reproducible and FAIR. Even if the GET or POST request of an OGC API — Processes could be saved, the full complete provenance tracking of the execution request with the request parameters and results obtained is not guaranteed.

Furthermore, the software environment is not enough to verify results of a computational analysis or reuse the methods in a reproducible FAIR perspective. Having the “retrospective provenance” is necessary. This is the detailed record of the implementation of a computational task including the details of every executed process together with comprehensive information about the execution environment used to derive a given product.

While CWL has emerged as a workflow definition standard designed to enable portability, interoperability, and reproducibility of analyses between workflow platforms, CWLProv is proposed [cwlprov] as a format to represent any workflow-based computational analysis to produce workflow output artifacts that satisfy various levels of provenance.

Starting from the interoperable workflow definitions of CWL and the structured provenance of the W3C PROV Model , the CWLProv defines a resource aggregation and sharing as workflow-centric Research Object (RO) [[Belhajjame et al., 2015]] that stores a given workflow enactment from input to the final outputs as multiple files in their native formats. The folder structure of the CWLProv complies with theBagIt [[RFC 8493]] format such that its content and completeness can be verified with any BagIt tool or library.

BagIt is a set of hierarchical file layout conventions for storage and transfer of arbitrary digital content. A “bag” has just enough structure to enclose descriptive metadata “tags” and a file “payload” but does not require knowledge of the payload’s internal semantics. This BagIt format is suitable for reliable storage and transfer.

In CWLProv the files used and generated by the workflow are considered the data payload and the remaining directories include metadata of how the workflow results were created.

The content of the Bagit CWLProv folder includes all the information to run the application package against the set of input parameters, intermediary files, and output results. All these files have their hash listed in metadata files allowing verification of the results between repeated runs. The official CWL runner, cwltool, implements the CWLProv approach and it is quite straightforward to produce the retrospective provenance folders and files in the Bagit format.

4.6.1. Input EO Data

As defined in OGC 20-089, the Application can take as input an array of Spatio-Temporal Asset Catalog (STAC) Items URLs aggregated into a STAC Collection describing input EO data. The STAC Collection Specification defines a set of common fields to describe a group of Items that share properties and metadata. This definition fits the purposes of aggregating the STAC Items into a single element that can potentially be cited and assigned a digital identifier.

To comply with the Reproducible FAIR Workflow scenario, the STAC collection defining the input EO data must use the STAC Scientific Citation Extension [STAC Citation] to indicate from which publication the data originates and how the data itself should be cited or referenced. The referencing capability of the Scientific Citation Extension is a key element to meet the goals of reproducibility and citation of an application package execution. With the assignment of a DOI to the STAC collection that aggregates the STAC Items used as inputs, the input dataset becomes citable and findable.

```
{
  "type": "Collection",
  "id": "collection-id",
  "stac_version": "1.0.0",
  "description": "this is the collection description",
  "links": [
    {
      "rel": "cite-as",
      "href": "https://doi.org/10.5072/zenodo.1102087"
    },
    {
      "rel": "self",
      "href": "https://sandbox.zenodo.org/api/files/f1cb1db1-1ece-4c1f-858c-
d753e99b36d1/collection.json",
      "type": "application/json"
    }
 ...
```

In the case of an input EO Data from a datacube, the STAC Datacube Extension specifies the datacube related metadata and enriches the input dataset STAC Collection with additional information on the temporal, spatial, and radiometric dimensions. The temporal enrichment provides the list of acquisition dates, the spatial provides the extent of x and y dimensions, and finally the radiometry provides clear and detailed information about the radiometric characteristics of the input data.

```
{
  "type": "Collection",
  "id": "collection-id",
  "stac_version": "1.0.0",
  "description": "this is the collection description",
  "links": [
    {
      "rel": "root",
      "href": "https://sandbox.zenodo.org/api/files/f1cb1db1-1ece-4c1f-858c-
d753e99b36d1/collection.json",
      "type": "application/json"
    },
    {
      "rel": "item",
      "href": "https://earth-search.aws.element84.com/v0/collections/sentinel-
s2-l2a-cogs/items/S2B_10TFK_20210713_0_L2A",
      "type": "application/json"
    },
    {
      "rel": "item",
      "href": "https://earth-search.aws.element84.com/v0/collections/sentinel-
s2-l2a-cogs/items/S2A_10TFK_20220524_0_L2A",
      "type": "application/json"
    },
    {
      "rel": "cite-as",
      "href": "https://doi.org/10.5072/zenodo.1102087"
    },
    {
      "rel": "self",
      "href": "https://sandbox.zenodo.org/api/files/f1cb1db1-1ece-4c1f-858c-
d753e99b36d1/collection.json",
      "type": "application/json"
    }
  ],
  "stac_extensions": [
    "https://stac-extensions.github.io/scientific/v1.0.0/schema.json",
    "https://stac-extensions.github.io/datacube/v2.0.0/schema.json"
  ],
  "cube:dimensions": {
    "x": {
      "axis": "x",
      "extent": [
        -121.8343226741975,
        -120.51956282559038
      ],
      "reference_system": 3857
    },
    "y": {
      "axis": "y",
      "extent": [
        39.63588071728383,
        40.64479052153662
      ],
      "reference_system": 3857
    },
    "time": {
      "extent": [
        "2021-07-13T19:03:24Z",
        "2022-05-24T19:03:29Z"
      ],
      "values": [
        "2021-07-13T19:03:24Z",
        "2022-05-24T19:03:29Z"
      ]
    },
    "spectral": {
      "type": "bands",
      "values": [
        "B1",  "B2", "B3",  "B4",
        "B5",  "B6", "B7",  "B8",
        "B8A", "B9", "B10", "B11",
        "B12", "QA10", "QA20", "QA60"
      ]
    }
  },
  "sci:doi": "10.5072/zenodo.1102087",
  "extent": {
    "spatial": {
      "bbox": [
        [
          -121.8343226741975,
          39.63588071728383,
          -120.51956282559038,
          40.64479052153662
        ]
      ]
    },
    "temporal": {
      "interval": [
        [
          "2021-07-13T19:03:24Z",
          "2022-05-24T19:03:29Z"
        ]
      ]
    }
  },
  "license": "proprietary"
}
```

4.6.2. Output EO Data

As defined in OGC 20-089, the output dataset as generated by the application package execution is a STAC Catalog with links to STAC Items representing and describing the resulting files.

Similarly, to the input EO data, to comply with the Reproducible FAIR Workflow scenario the output EO data must be defined as a STAC Collection with the Scientific Citation Extension to convey the DOI. Additionally, and for extra traceability and reproducibility validation, the output STAC Collection must use the STAC File Extension to provide the file MD5 for each asset.

In the case of an output EO Data that publishes to a GeoDataCube, the STAC Datacube Extension specifies the GeoDataCube related metadata and enriches the output dataset STAC Collection with additional information on the temporal, spatial, and radiometric dimensions. The temporal enrichment provides the list of acquisition dates, the spatial provides the extent of x and y dimensions, and finally the radiometry provides clear and detailed information about the radiometric characteristics of the output data.

```
{
  "type": "Collection",
  "id": "collection-id",
  "stac_version": "1.0.0",
  "description": "this is the collection description",
  "links": [
    {
      "rel": "root",
      "href": "https://sandbox.zenodo.org/api/files/00ba172e-c9f0-4552-91e1-
342a63cf3477/collection.json",
      "type": "application/json"
    },
    {
      "rel": "item",
      "href": "https://sandbox.zenodo.org/api/files/00ba172e-c9f0-4552-91e1-
342a63cf3477/S2B_10TFK_20210713_0_L2A/S2B_10TFK_20210713_0_L2A.json",
      "type": "application/json"
    },
    {
      "rel": "item",
      "href": "https://sandbox.zenodo.org/api/files/00ba172e-c9f0-4552-91e1-
342a63cf3477/S2A_10TFK_20220524_0_L2A/S2A_10TFK_20220524_0_L2A.json",
      "type": "application/json"
    },
    {
      "rel": "cite-as",
      "href": "https://doi.org/10.5072/zenodo.1102089"
    },
    {
      "rel": "self",
      "href": "https://sandbox.zenodo.org/api/files/00ba172e-c9f0-4552-91e1-
342a63cf3477/collection.json",
      "type": "application/json"
    }
  ],
  "stac_extensions": [
    "https://stac-extensions.github.io/scientific/v1.0.0/schema.json",
    "https://stac-extensions.github.io/datacube/v2.0.0/schema.json"
  ],
  "cube:dimensions": {
    "x": {
      "axis": "x",
      "extent": [
        -121.413752588606,
        -120.71922542174708
      ],
      "reference_system": 3857
    },
    "y": {
      "axis": "y",
      "extent": [
        39.83402935827303,
        40.47202226335379
      ],
      "reference_system": 4326
    },
    "time": {
      "extent": [
        "2021-07-13T19:03:24Z",
        "2022-05-24T19:03:29Z"
      ],
      "values": [
        "2021-07-13T19:03:24Z",
        "2022-05-24T19:03:29Z"
      ]
    },
    "spectral": {
      "type": "bands",
      "values": [
        "B1", "B2", "B3", "B4",
        "B5", "B6", "B7", "B8",
        "B8A", "B9", "B10", "B11",
        "B12", "QA10", "QA20", "QA60"
      ]
    }
  },
  "sci:doi": "10.5072/zenodo.1102089",
  "extent": {
    "spatial": {
      "bbox": [
        [
          -121.413752588606,
          39.83402935827303,
          -120.71922542174708,
          40.47202226335379
        ]
      ]
    },
    "temporal": {
      "interval": [
        [
          "2021-07-13T19:03:24Z",
          "2022-05-24T19:03:29Z"
        ]
      ]
    }
  },
  "license": "proprietary"
}
```

5

CONCLUSION

5 CONCLUSION

This Testbed 18 ER addresses the challenge of scientific algorithm portability and reproducibility. The document defines a set of best practices to ensure that the data processing services of Exploitation Platforms follow the FAIR principles and become fully reproducible for large areas and temporal series.

Past testbed activities have shown how platforms for the Exploitation of Earth Observation operate applications and workflows by using a combination of pre-processed data, fusion, and analytical functions. This ER focuses on how to extend the application of the FAIR principles from data to Earth Observation algorithms, tools, and workflows that created the data. From the FAIR principles' perspective, the resources become all platform elements like publications, data, services, products, information, software, or computing environments.

The FAIR Data Principles are a set of guidelines to help make data findable, accessible, interoperable, and reusable. These principles provide the initial guidance for the Exploitation Platform data producers and data publishers to promote maximum use of research data as a framework for fostering and extending the Platform's data services.

The EO Application Package architecture decouples application developers from exploitation platform operators and from application consumers by allowing application developers to make their applications available on a number of platforms with minimal modifications. Taking the FAIR principles as guidelines to enhance the scientific data reusability, the activities performed for this ER presented how the existent model of EO Application Package and corresponding APIs can enhance the reproducibility of results and traceability of application executions and processes.

The ER focuses on identifying the interoperability requirements and respective conformance classes that the Application, Package, and Platform need to address so that all elements essential to the exploitation of data are described and ensure transparency, reproducibility, and reusability. As a result, the best practices ensures that the FAIR principles are applied to all Earth science algorithms used to generate results in an Exploitation Platform that can be referenced in a scientific publication for future reproducibility.

Past activities guaranteed to the communities that their research would be deployable and executed in multiple Cloud providers. The addition of the proposed new conformance classes opens the way to also enable reproducibility in terms of data inputs, algorithm packaging, and data outputs respecting the FAIR principles to support the reusability of these digital assets in Big Data challenges.

Starting from the Application Package encoding as CWL defined in OGC 20-089, the addition of CodeMeta and CWLProv, with a strong commitment in the Common Workflow Language developer community, shows real promise as a format for the methodical representation of EO application enactment, associated resources, and capturing and using retrospective provenance information.

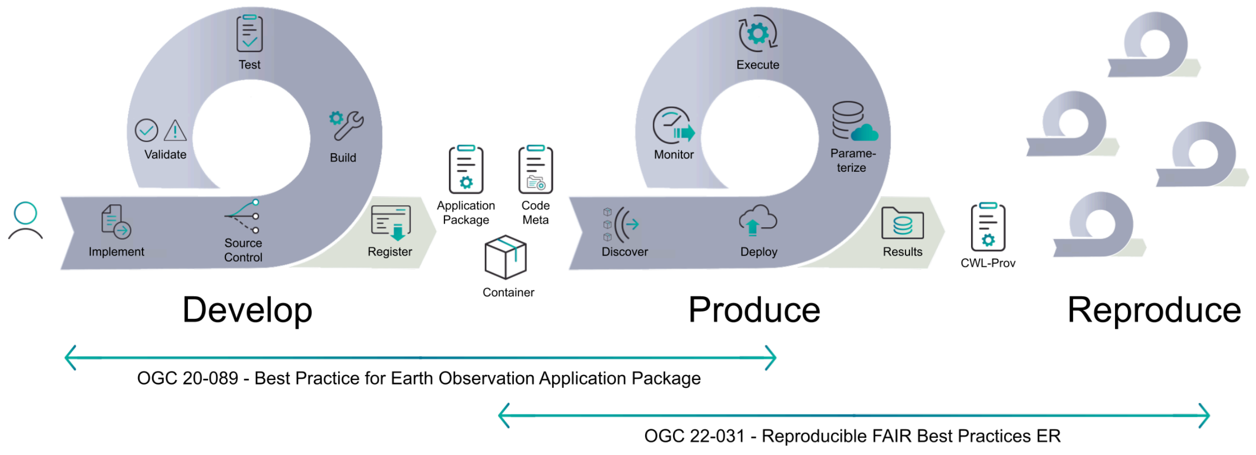


Figure 5 — Full Cycles from Development to Production to Reproducibility

In conclusion, the Testbed participants drafted an initial set of best practices and conformance classes that extends the OGC Best Practices for Application Package. A more enlarged activity taking into consideration a larger set of applications, scenarios, and use cases is necessary to validate and evolve the proposed best practices. In conclusion, an OGC Best Practice document, complementary to the OGC 20-089 and identifying the additional conformance classes should be written and adopted.

A

ANNEX A (INFORMATIVE) REVISION HISTORY

A ANNEX A (INFORMATIVE) REVISION HISTORY

DATE	RELEASE	AUTHOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2022-10-31	1	Pedro Gonçalves	All	Initial Version

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Khan, F.Z., Soiland-Reyes, S., Sinnott, R.O., Lonie, A., Goble, C., and Crusoe, M.R.: Sharing interoperable workflow provenance: A review of best practices and their practical application in CWLProv. GigaScience. 8, giz095 (2019).

[2] Wilkinson, M. D. et al.: The FAIR Guiding Principles for scientific data management and stewardship. Sci. Data 3:160018 doi: 10.1038/sdata.2016.18 (2016).

[3] Belhajjame, K., Zhao, J., Garijo, D., Gamble, M., Hettne, K., Palma, R., Mina, E., Corcho, O., Gómez-Pérez, J.M., Bechhofer, S., et al.: Using a suite of ontologies for preserving workflow-centric research objects. Journal of Web Semantics. 32, 16–42 (2015).