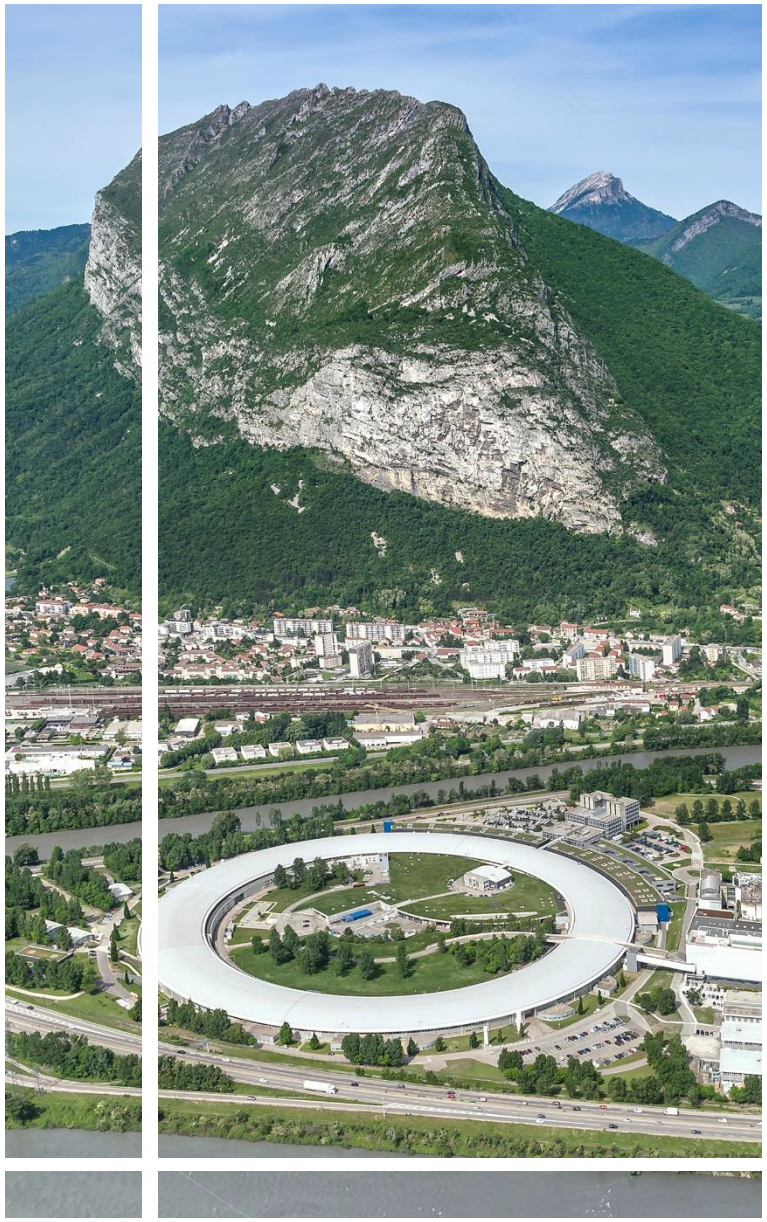
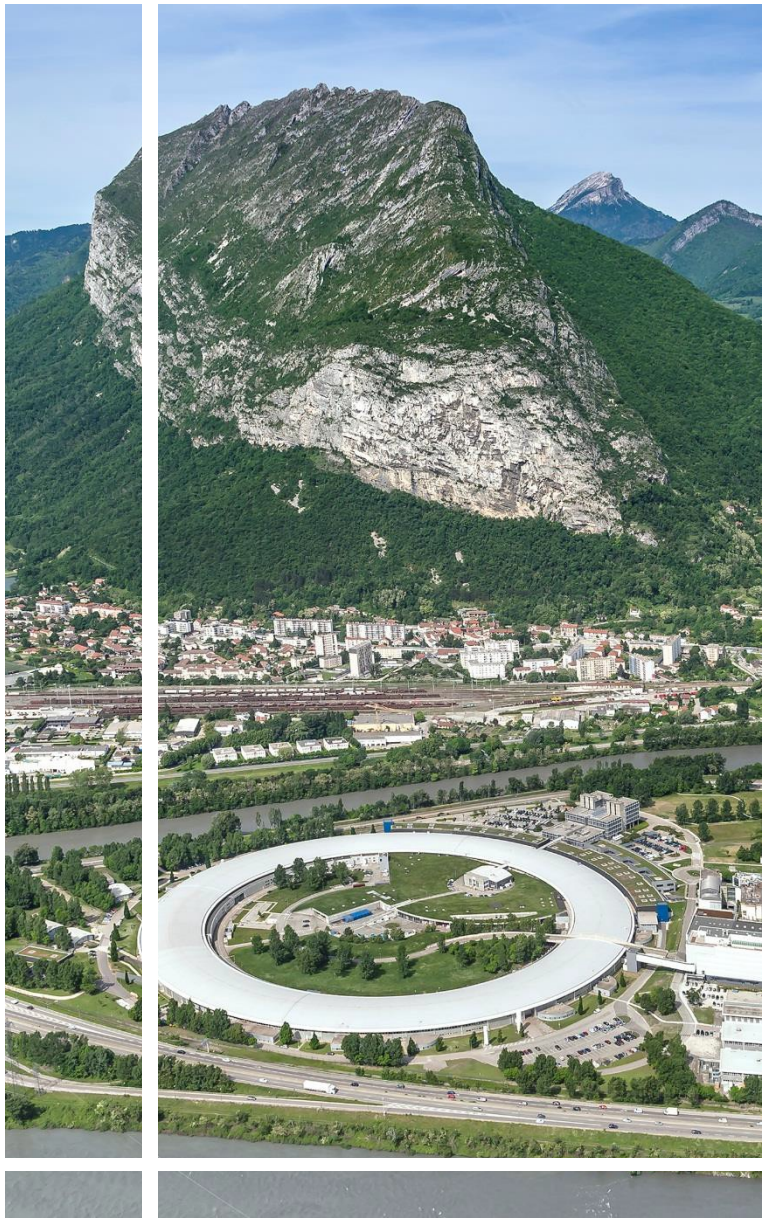

ESRF
The European Synchrotron



EWOKS (Extensible Workflow System)
Automation and FAIR data analysis

Wout De Nolf
ESRF (Data Automation Unit)

STREAMLINE has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 870313

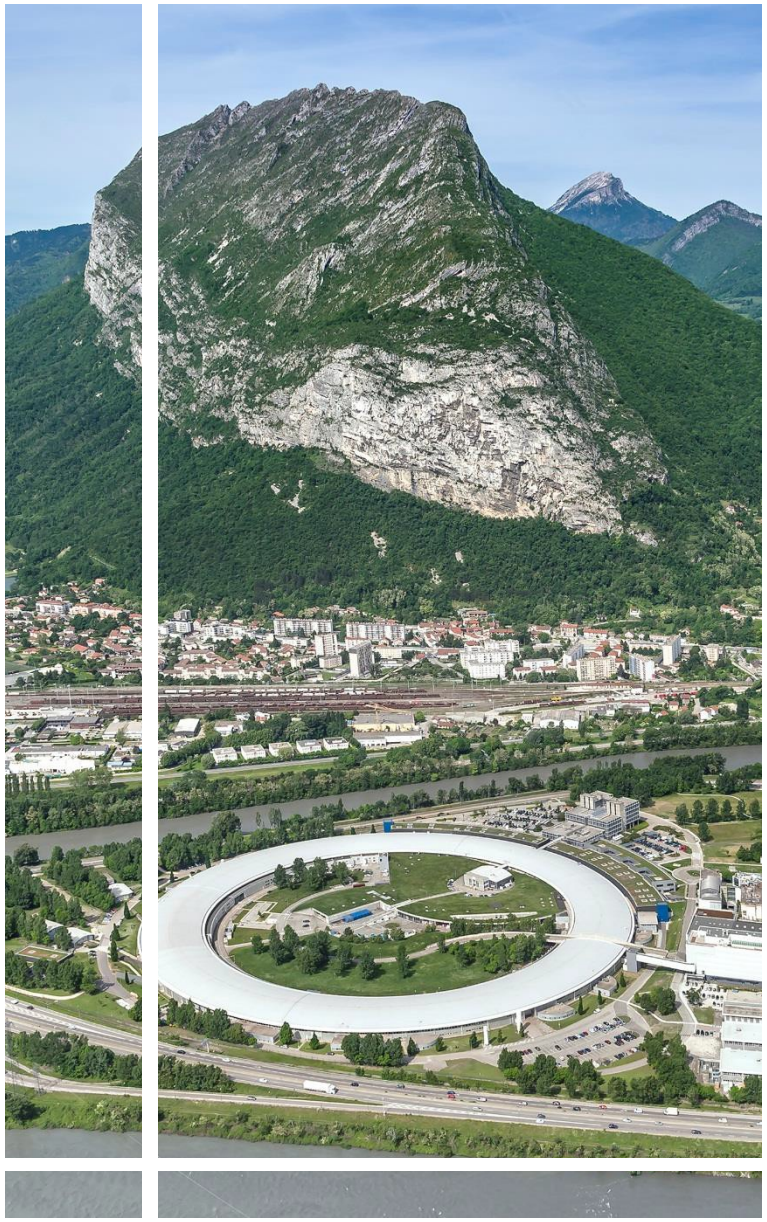


The goal of EWOKS is NOT to provide a workflow system.

The goal of EWOKS is to provide a solution to

1. **automate** data processing and beamline operation
2. and make data processing results **FAIR** (the traceable and reproducible aspect).

A workflow-based approach was chosen (as opposed to e.g. a plugin-based approach) but it is not the goal in and of itself.

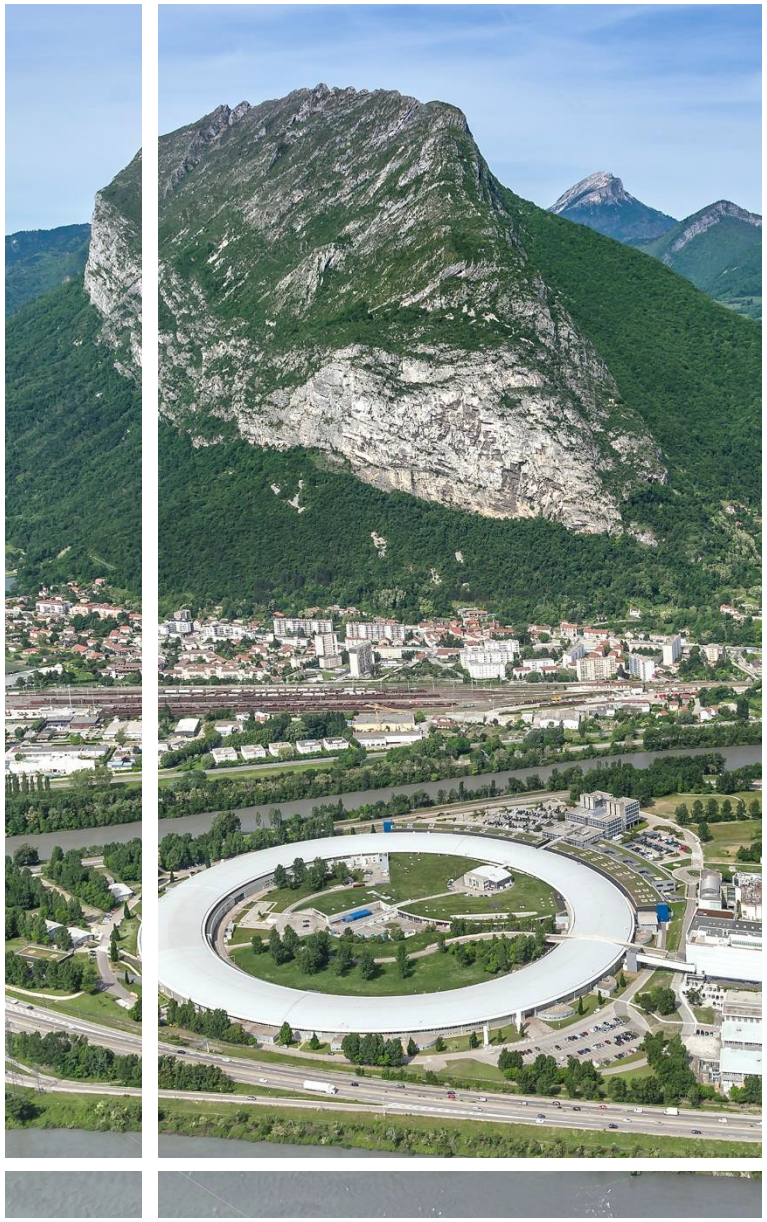


Goal of this presentation

Why did the ESRF develop a “meta workflow system”?

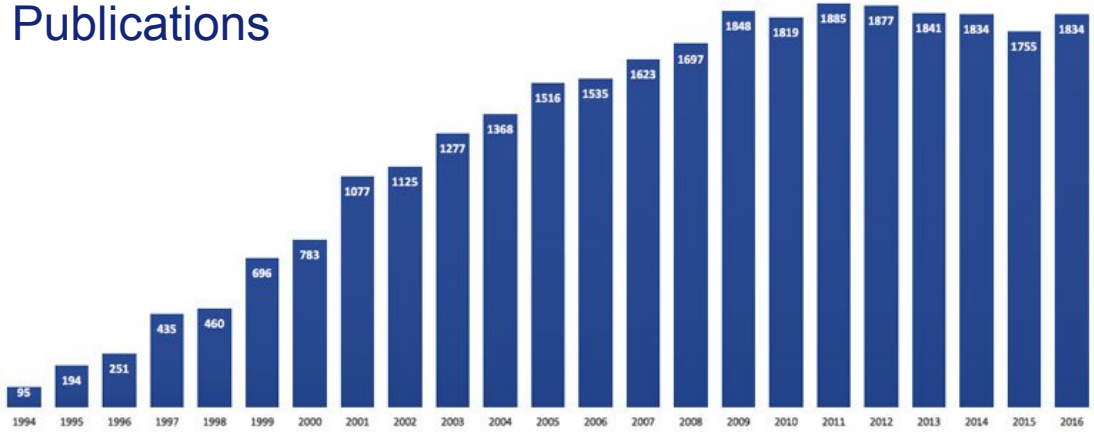
Meta workflow system: **decouple** workflow representation and implementation from the workflow management system that executes, visualizes and manages workflows.

WHAT PROBLEM IS EWOKS SOLVING?

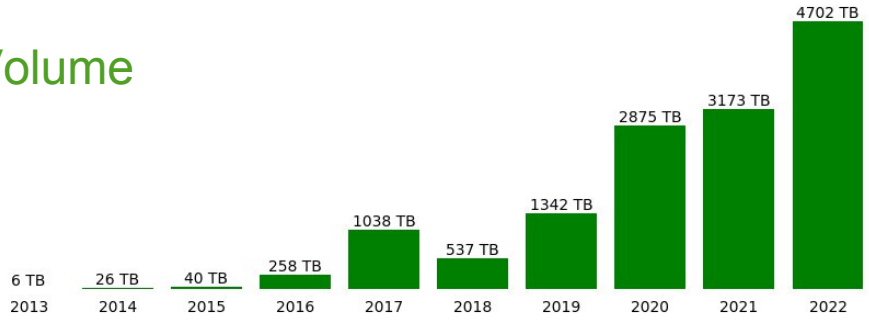


⚠ Plateau in scientific output while data volume increases ⚠

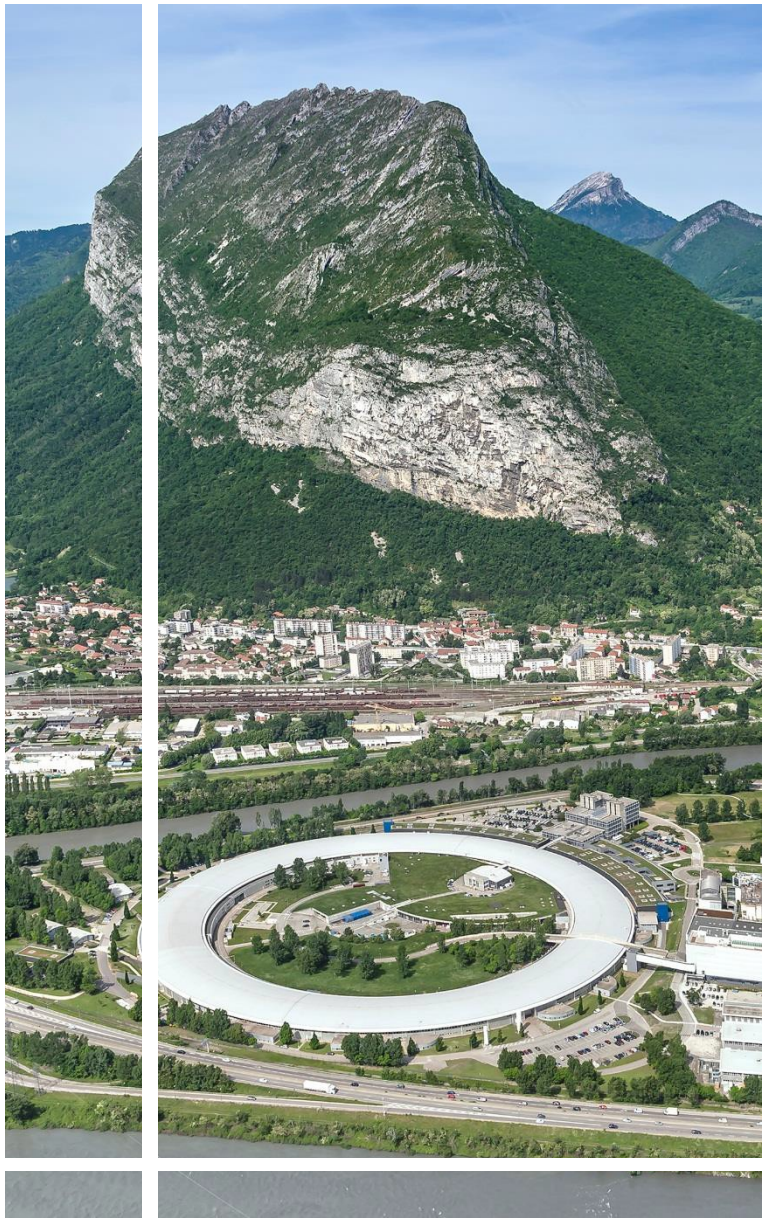
Publications



Data Volume



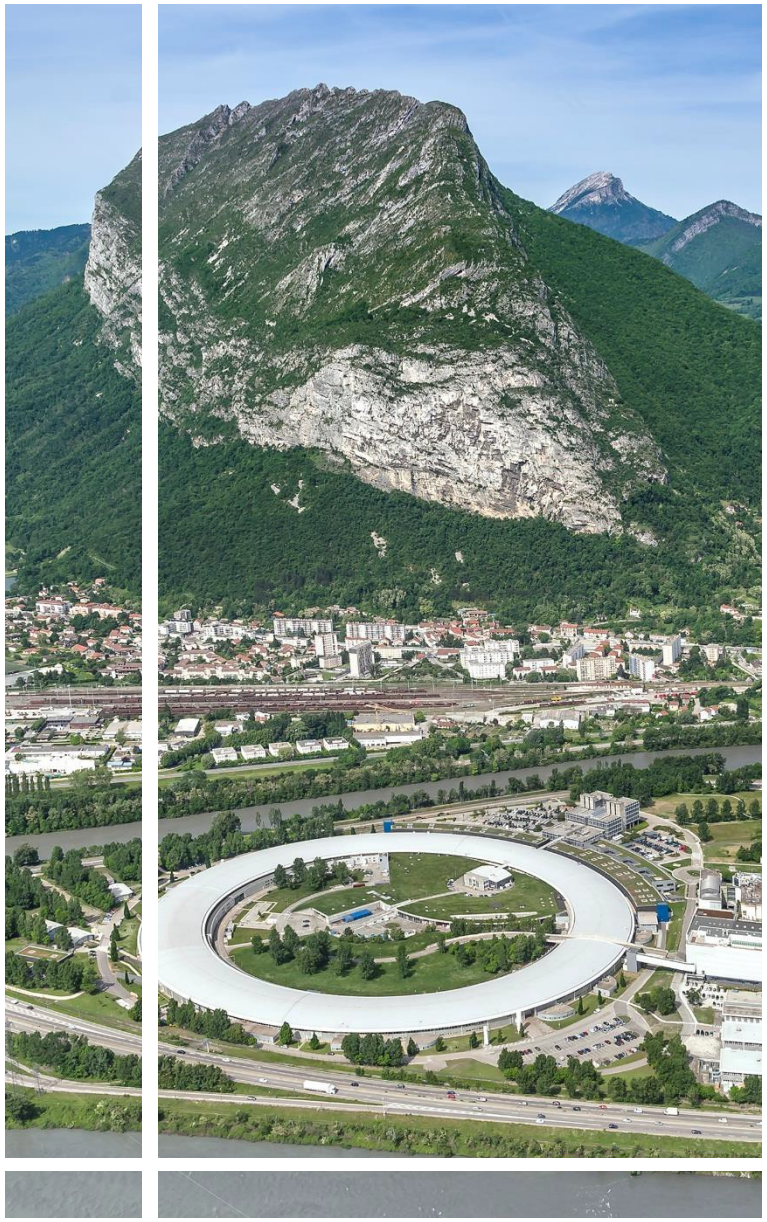
WHAT PROBLEM IS EWOKS SOLVING?



Bottleneck is no longer X-ray flux or detectors, it is beamline operation and data processing.

- Replace beamline operators with expert systems
- Automate data processing or at least persist the expert usage of scientific software in the case of interactive workflows

WHAT PROBLEM IS EWOKS SOLVING?



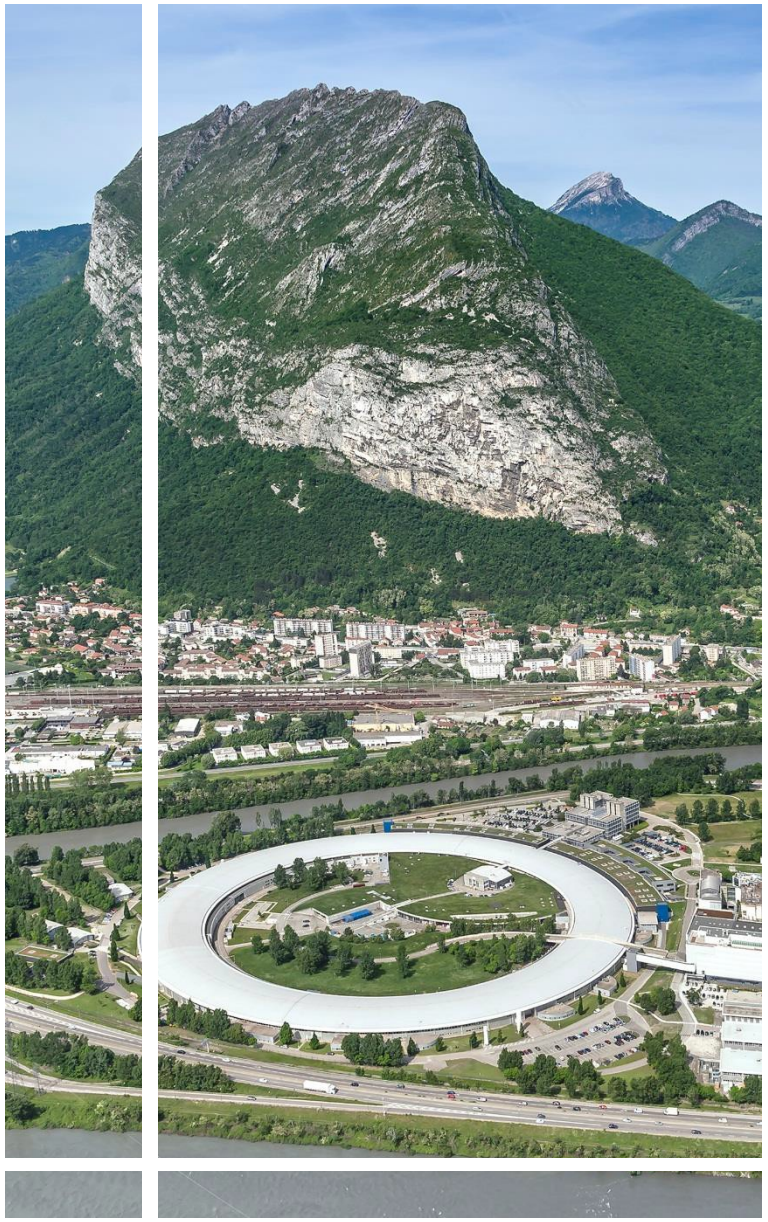
Several workflow-based solutions have been used in the past at the ESRF

⚠ requirements change over time

⚠ complexity in usage and/or maintenance

⚠ different workflow solutions are needed because of their different features but this is inefficient for users and maintainers

WHAT PROBLEM IS EWOKS SOLVING?



⚠ Plateau in scientific output while data volume increases ⚠

We needed a domain and beamline agnostic solution with the following properties:

- **easy to install** (runs everywhere from laptops to clusters: *bring the workflows to the data or the instruments and not the other way around*)
- **interactive** vs. non-interactive workflows
- **offline data processing** (DAGs) vs. **online data processing or expert systems** (loops and conditions)
- scientific software (often in python) needs to be **integrated**, not re-written
- **in-memory data transfer** between workflow nodes (no copying or serialization)
- the same workflow needs to run under **different circumstances** (on a laptop with a desktop GUI, headless on an HPC cluster, from the ESRF data portal, from the acquisition control system, ...)
- **reproducible and traceable** data processing (*installable and executable data provenance document* saved with the results)

WHAT PROBLEM IS EWOKS SOLVING?

Workflow Systems

Curated list by the CWL community
(~350 systems)

https://s.apache.org/existing-workflow-systems

Side note: *“Common Workflow Language (CWL) is an open standard for describing how to run command line tools and connect them to create workflows.”*

Doesn’t match the large python-based eco-system of scientific software.

WHAT PROBLEM IS EWOKS SOLVING?

None have all features we need

- Graphical interface for desktop
- Interactive execution
- Parallel execution
- Distribution on a compute cluster
- Support for loops
- Python can be easily integrated
- Web service to manage and execute workflows
- Workflow execution without the need for infrastructure

The list of requirements will change over time

EWOKS: A META WORKFLOW SYSTEM

Extensible Workflow System
https://ewoks.esrf.fr

- Project started in July 2021. In production since January 2023.
- Deployed at 25 ESRF beamlines
- 6 core developers in 2024
- MIT License
- Ewoks workflows are published under the FAIR principles so the ESRF is committed to long term support.
- Workflow systems currently supports:
  - Orange: desktop GUI
  - Dask: parallel + cluster
  - Pypushflow: parallel + loops

Apache Airflow

Luigi

orange
Desktop GUI

APACHE Spark

Reflow

RAPIDS

COMMON WORKFLOW LANGUAGE

DASK
Parallel + cluster

RAY

Galaxy

DAGSTER

snakemake

PREFECT

PYPUSHFLOW
Loops and conditions

nextflow

EWOKS: A META WORKFLOW SYSTEM

Meta Workflow System: workflow representation and implementation are independent from the workflow systems.

EWOKS provides these component

- workflow definition language: defines the meaning of workflow nodes and their links
- workflow representation: in-memory, file-based, serialized representation of workflows for execution or storage
- workflow task representation: interface required by the execution engines to execute the underlying software that does the actual computations, allowing to add common features of tasks independent from the WFMS like execution events and caching of task output
- workflow events: mechanism for following workflow progress for logging or other workflow management purposes

A binding between EWOKS and any workflow management system (WFM) can be created by mapping the workflow and task representation of EWOKS to the equivalent in the WFMS.

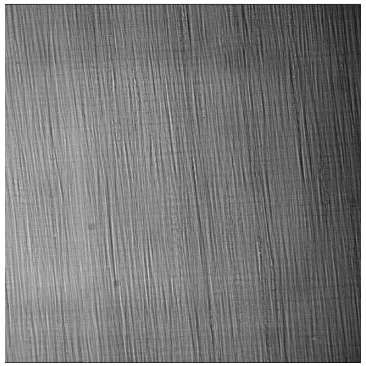
Workflow system provides these component (EWOKS provides a default for each)

- workflow engine: job scheduling of workflow tasks as defined in the workflow definition language
- workflow scheduling: job scheduling of workflows on a workflow basis, as opposed to what a workflow engine does on a workflow node basis
- interface for workflow definition and execution: this refers to an API or service (e.g. python API, REST) or a graphical interface (web or desktop)

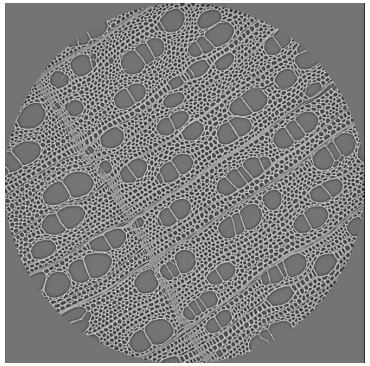
EWOKS: A META WORKFLOW SYSTEM

Automation of existing scientific software in different X-ray domains at 25 ESRF beamlines

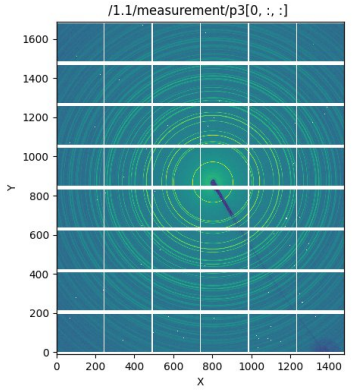
Tomography



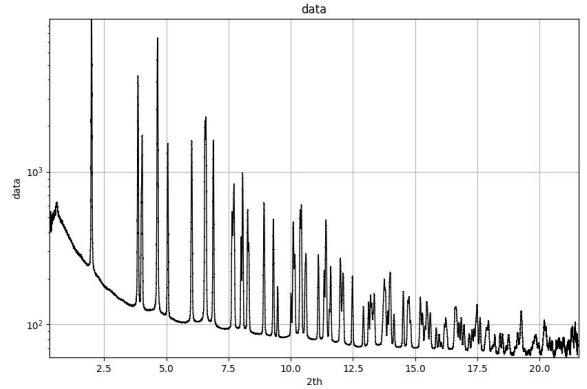
nabu



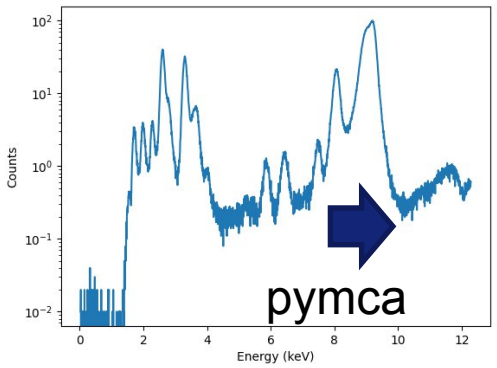
Diffraction



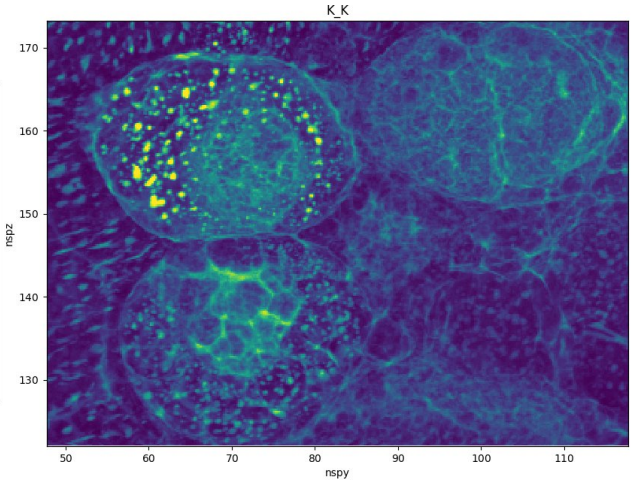
pyFAI



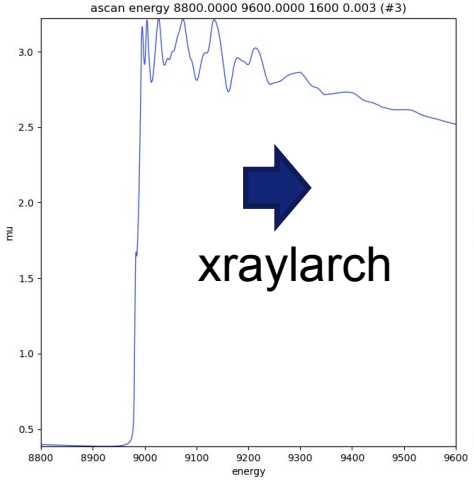
Fluorescence



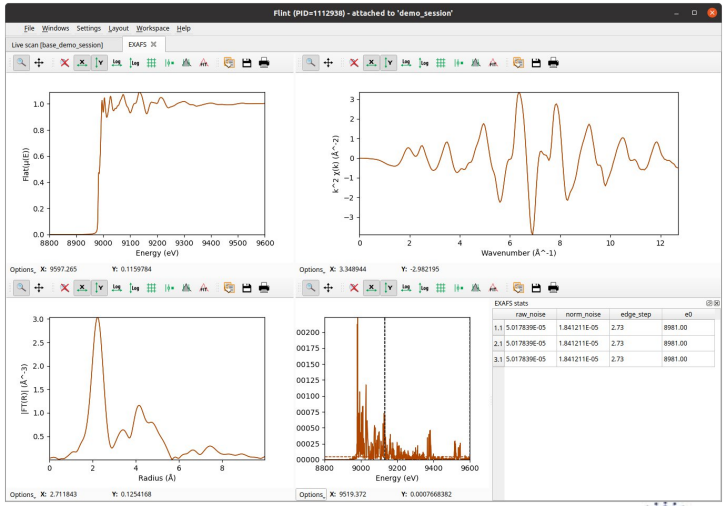
pymca



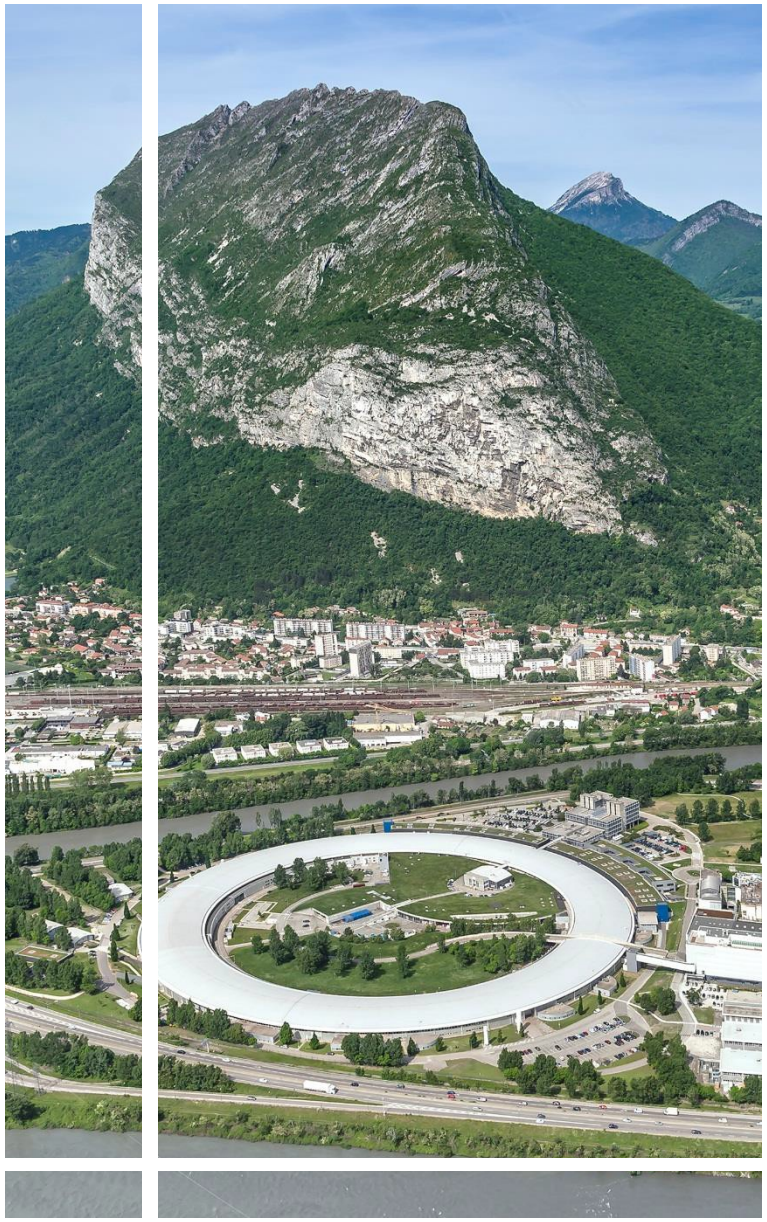
Spectroscopy



xraylarch



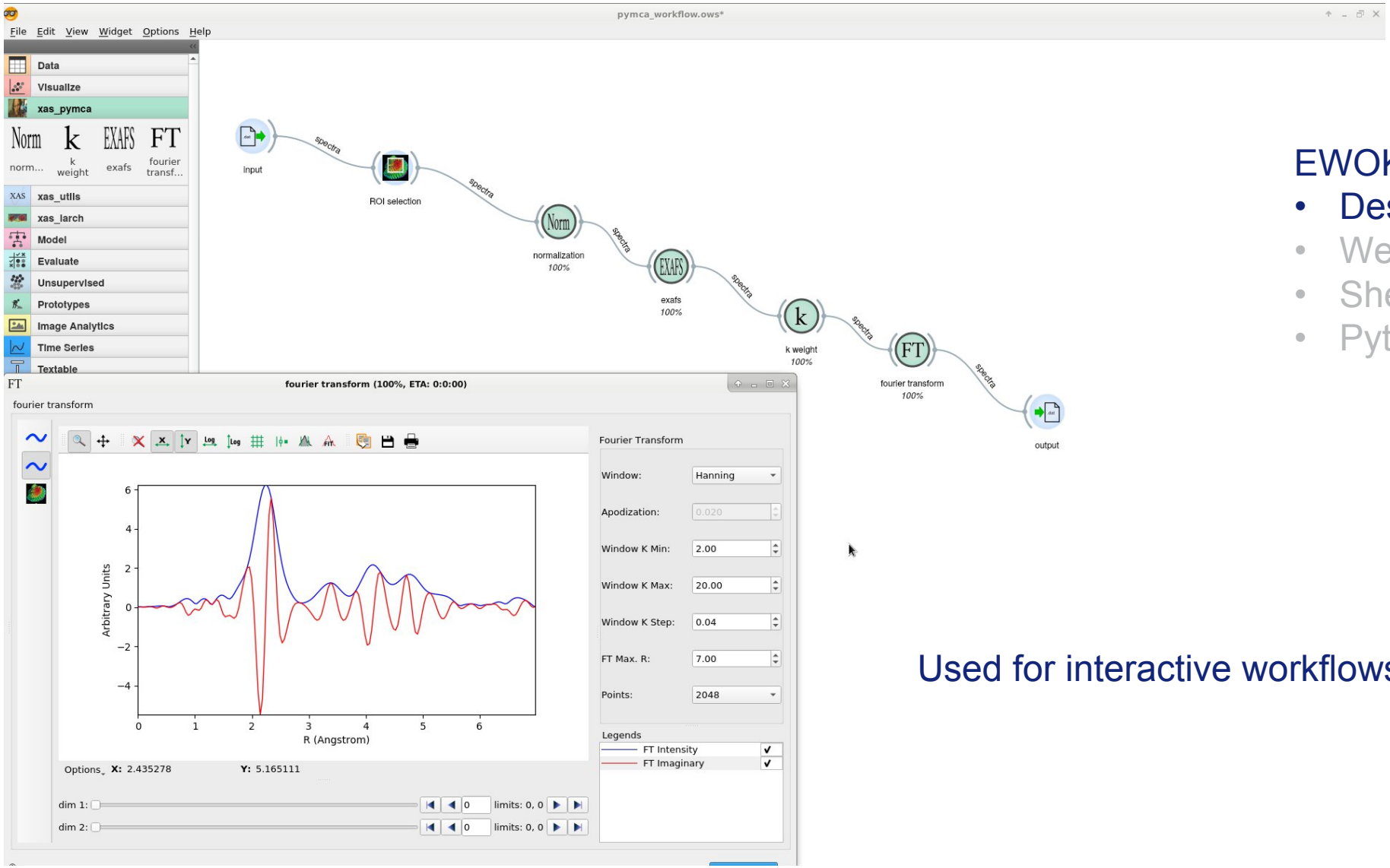
EWOKS INTERFACES



To conclude an overview of all EWOKS interfaces and why they exist

- Desktop (interactive workflows)
- Web (workflows as a service)
- Shell (headless execution)
- Python (integration for developers)

EWOKS INTERFACES



EWOKS interfaces

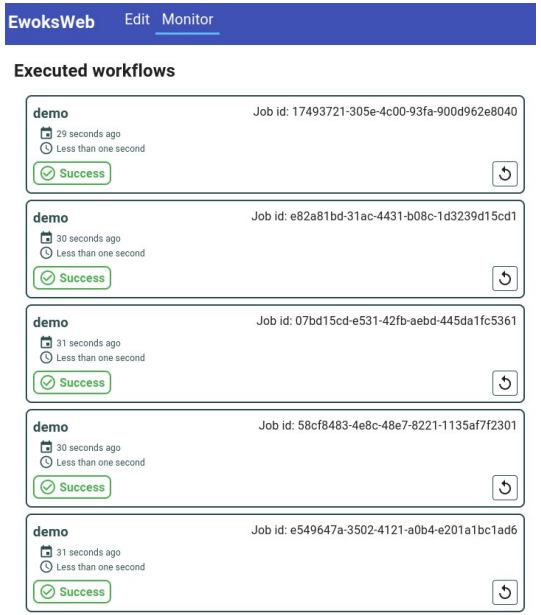
- Desktop
- Web
- Shell
- Python

Used for interactive workflows

EWOKS INTERFACES

EWOKS interfaces

- Desktop
- Web
- Shell
- Python

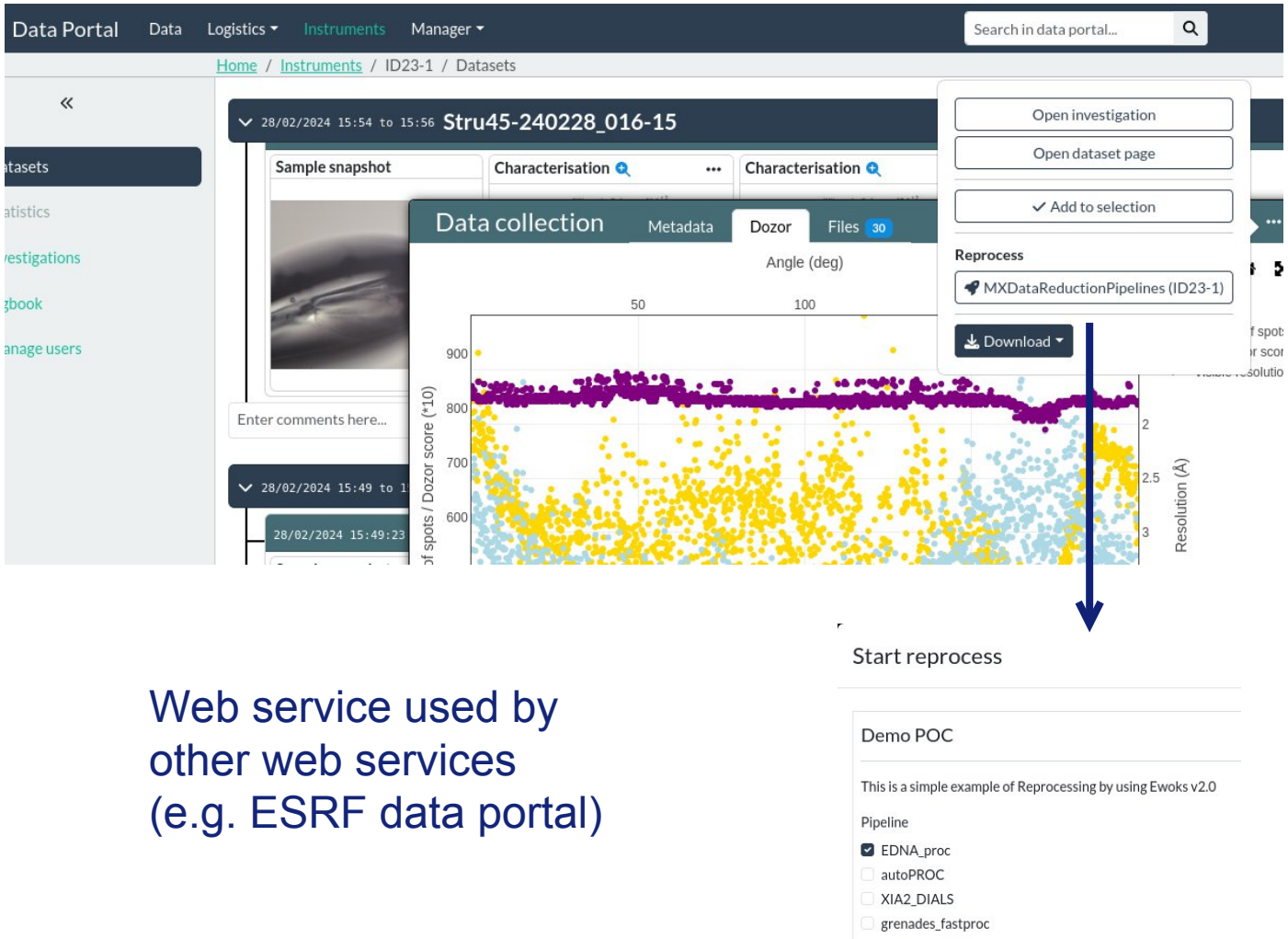


Used to visualize workflows that don't have graphical components

Standalone + frontend (similar to Jupyter notebooks)

EWOKS INTERFACES

https://data2.esrf.fr/

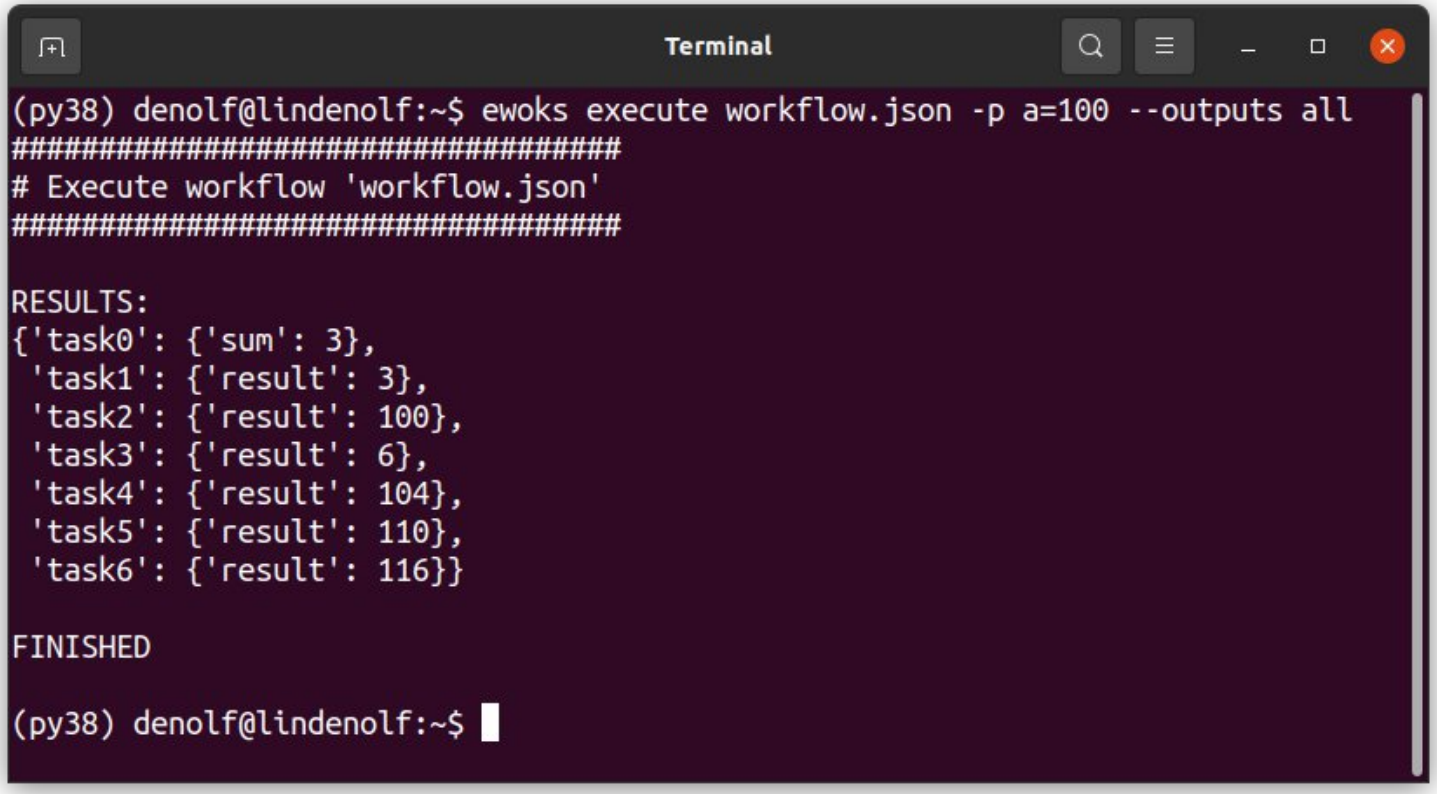


Web service used by other web services (e.g. ESRF data portal)

EWOKS interfaces

- Desktop
- Web
- Shell
- Python

EWOKS INTERFACES



EWOKS interfaces

- Desktop
- Web
- Shell
- Python

Used for headless execution

EWOKS INTERFACES

```python
from ewokscore import Task
from ewoks import execute_graph

# Implement a workflow task
class SumTask(
    Task, input_names=["a"], optional_input_names=["b"],
output_names=["result"]
):
    def run(self):
        result = self.inputs.a
        if self.inputs.b:
            result += self.inputs.b
        self.outputs.result = result


# Define a workflow with default inputs
nodes = [
    {
        "id": "task1",
        "task_type": "class",
        "task_identifier": "__main__.SumTask",
        "default_inputs": [{"name": "a", "value": 1}],
    },
    {
        "id": "task2",
        "task_type": "class",
        "task_identifier": "__main__.SumTask",
        "default_inputs": [{"name": "b", "value": 1}],
    },
    {
        "id": "task3",
        "task_type": "class",
        "task_identifier": "__main__.SumTask",
        "default_inputs": [{"name": "b", "value": 1}],
    },
]
links = [
    {
        "source": "task1",
        "target": "task2",
        "data_mapping": [{"source_output": "result", "target_input": "a"}],
    },
    {
        "source": "task2",
        "target": "task3",
        "data_mapping": [{"source_output": "result", "target_input": "a"}],
    },
]
workflow = {"graph": {"id": "testworkflow"}, "nodes": nodes, "links": links}

# Define task inputs
inputs = [{"id": "task1", "name": "a", "value": 10}]

# Execute a workflow (use a proper Ewoks task scheduler in production)
varinfo = {"root_uri": "/tmp/myresults"}  # optionally save all task outputs
result = execute_graph(workflow, varinfo=varinfo, inputs=inputs)
print(result)
```

Developer usage like triggering workflows from the acquisition control system.

EWOKS interfaces

- Desktop
- Web
- Shell
- Python

Hello world example of EWOKS:

https://ewoks.readthedocs.io/en/latest/tutorials/hello_world.html

- Define a task (could also be a python function, shell command, jupyter notebook or another workflow)
- Workflow representation (dict in python) of nodes and links
- Workflow execution from python

Usage

```
pip install ewoks
python myscript.py
```

or

```
ewoks execute demo --test --outputs all
```

EWOKS @ NOBUGS

Tuesday 24 September (BLISS and EWOKS as collaborative platform)

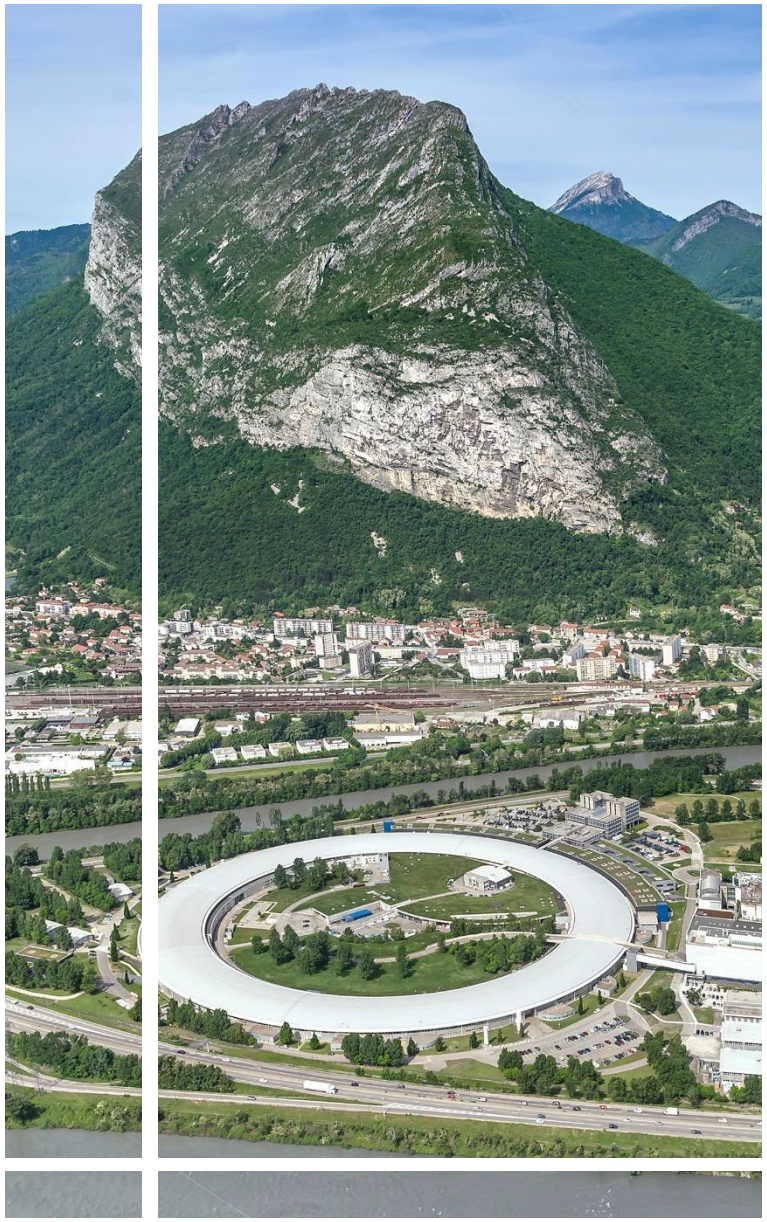
10:45 → 11:15 (Visitor Center)

EWOKS demo (Wout De Nolf - ESRF)

Thursday 26 September (Session: workflow engines)

16:40 → 16:55 (ESRF Auditorium)

Automated data processing with EWOKS for ESRF beamlines and users (Loic Huder - ESRF)



Questions ?

https://ewoks.esrf.fr

STREAMLINE has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 870313

https://streamline.esrf.fr/