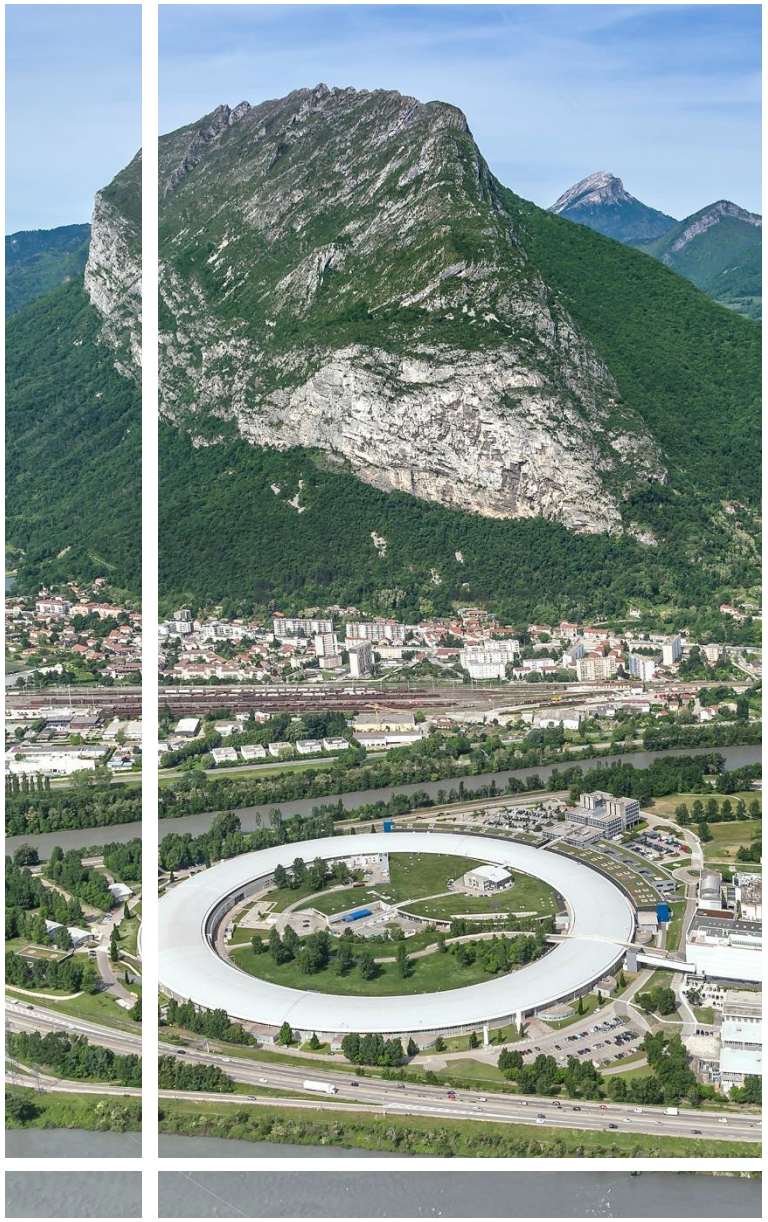
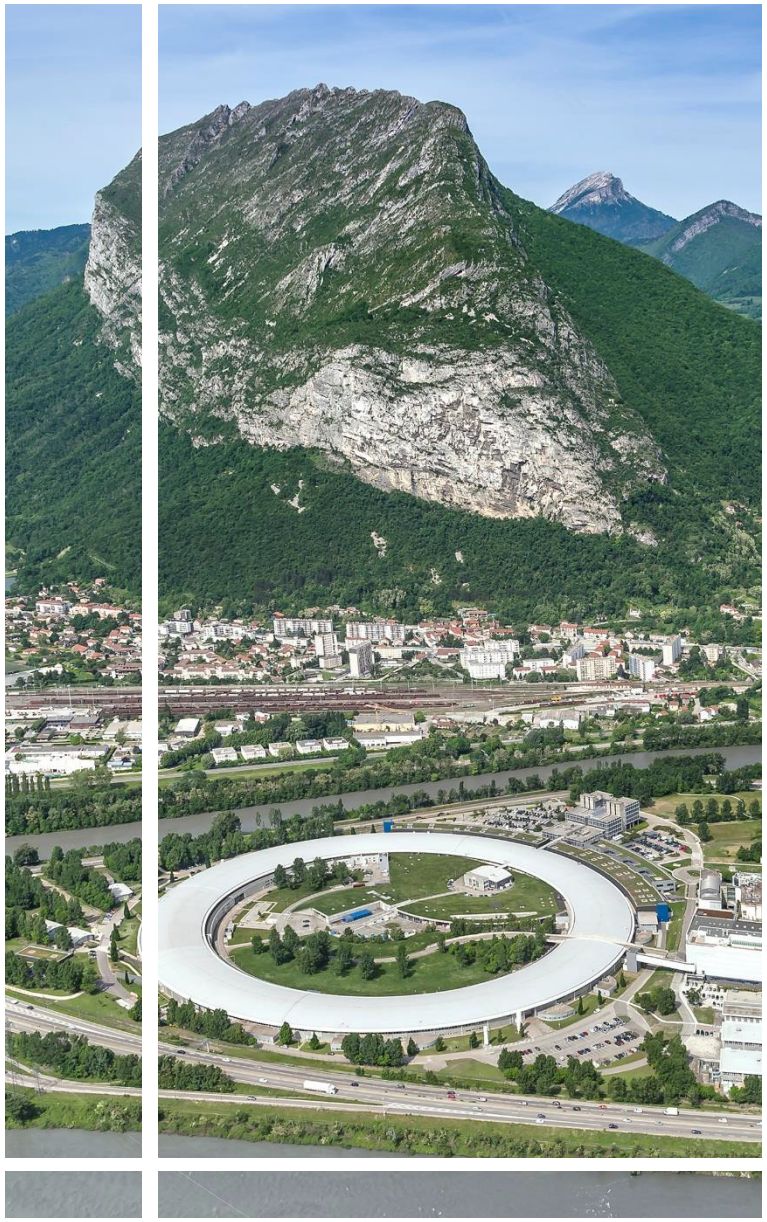

ESRF
The European Synchrotron



EWOKS (ESRF Workflow System)
A meta workflow system

Wout De Nolf
ESRF (Data Automation Unit)

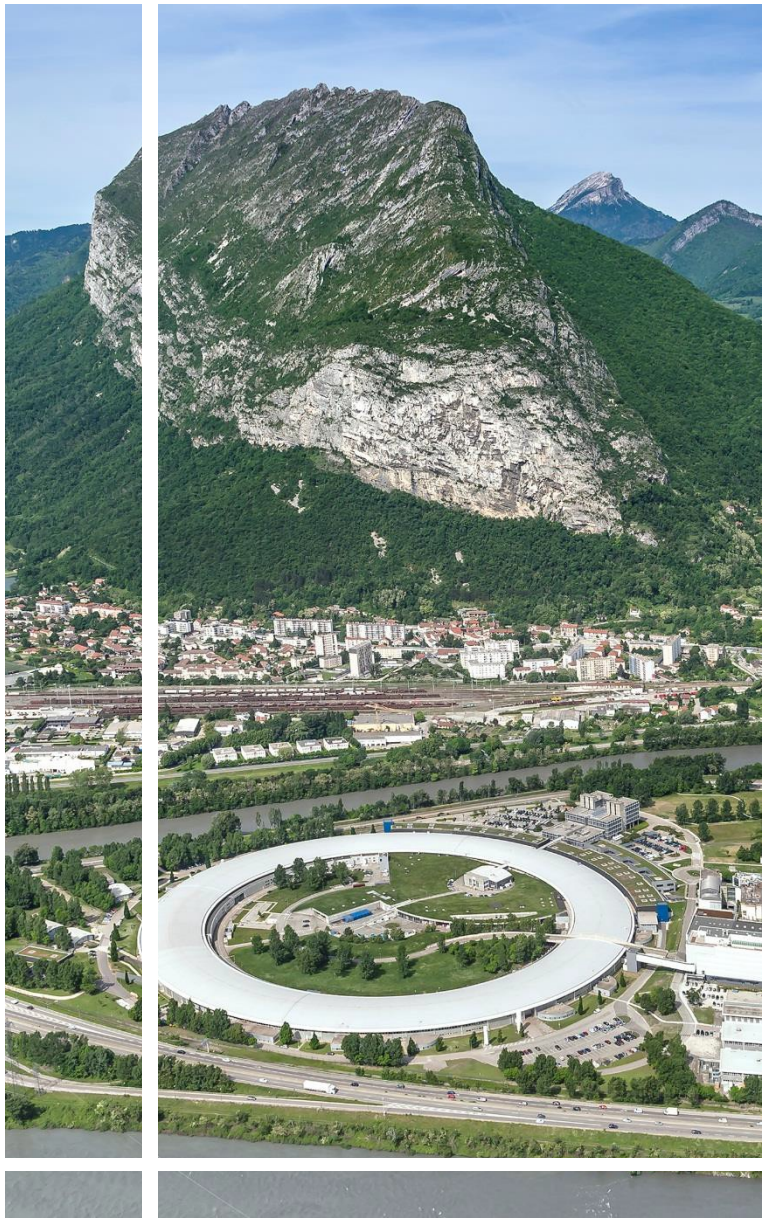
STREAMLINE has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 870313



Goal of this presentation

Demonstrate how the ESRF handles increasing data processing challenges with a “meta workflow” approach.

DATA PROCESSING CHALLENGES



What are the data processing challenges?

1992-2018

ESRF

Produce Unique Data

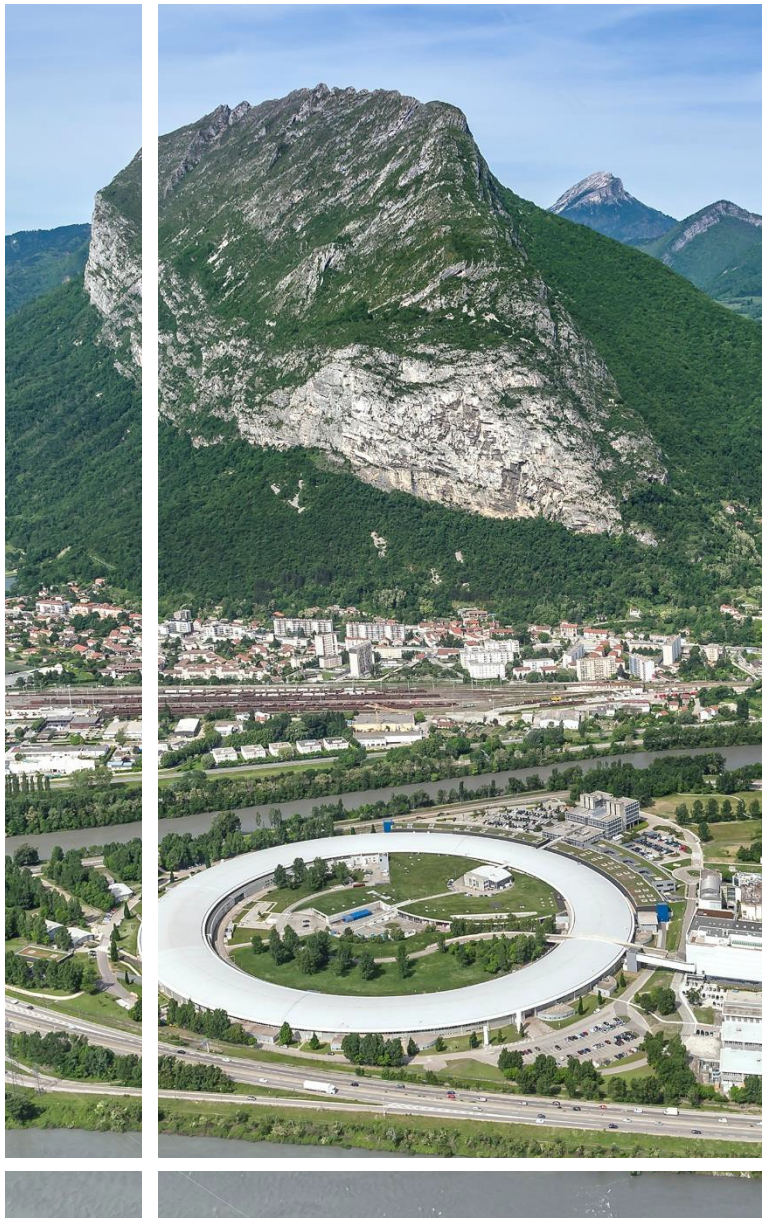
2020-present

ESRF-EBS
Extremely Brilliant Source

Produce Unique Results

Findable Accessible Interoperable Reusable

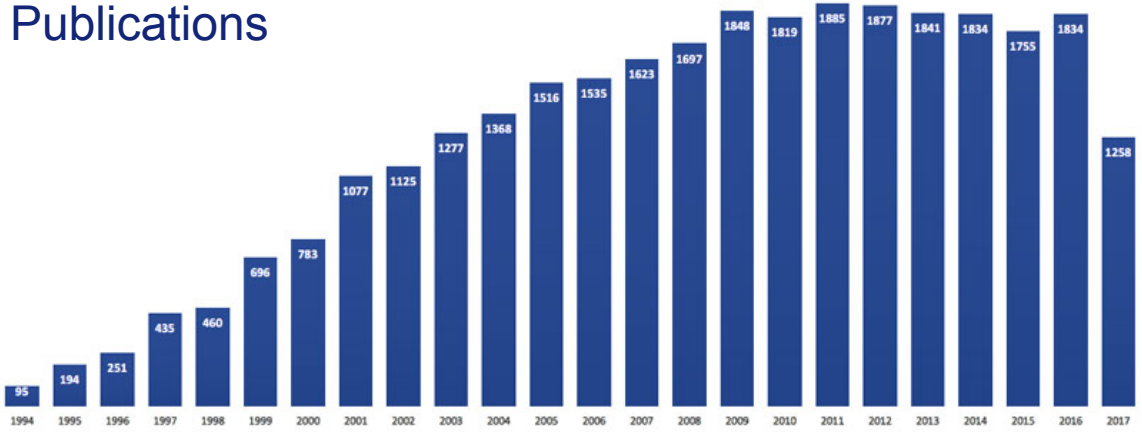
DATA PROCESSING CHALLENGES



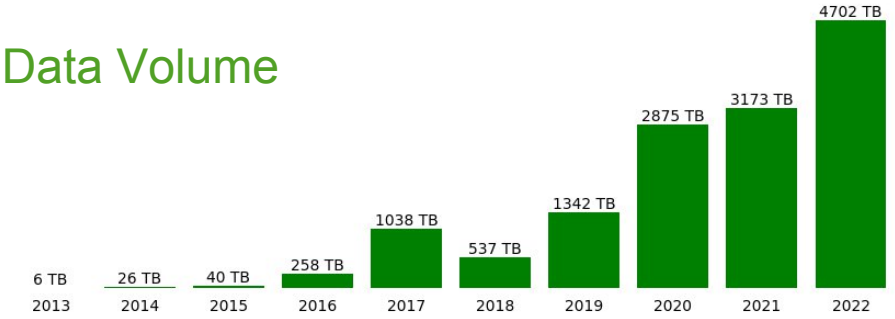
What are the data processing challenges?

⚠ Plateau in scientific output while data volume increases ⚠

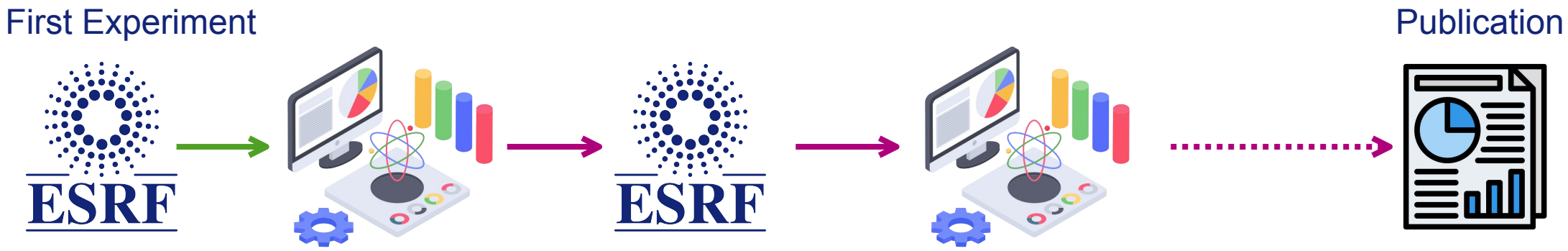
Publications



FAIR Data Volume



DATA PROCESSING CHALLENGES



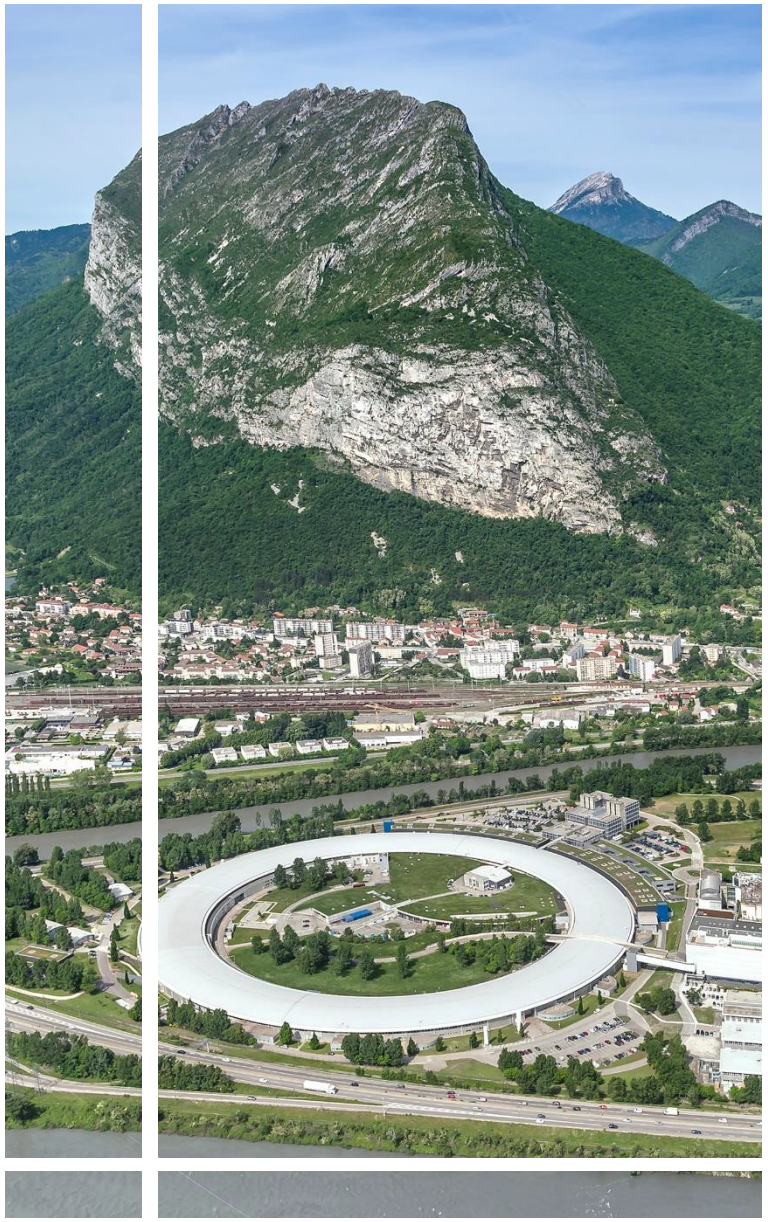
Timescale of months at best, often one year or more

Steep learning curve for new users from a wide range of scientific domains
Growing the synchrotron community has reached its limits

DATA PROCESSING CHALLENGES

- **Visualize the experiment** for humans, not the way X-ray detectors see it
- Produce **domain specific results** when possible, X-ray results when necessary
- Integrate scientific software in the experiment (acquisition control system and data portal)

DATA PROCESSING CHALLENGES



- 🗝 **Visualize the experiment** for humans
- 🗝 Produce **domain specific results** when possible
- ➥ Integrate scientific software in the experiment

Attempts at the ESRF to do this are domain or beamline specific.

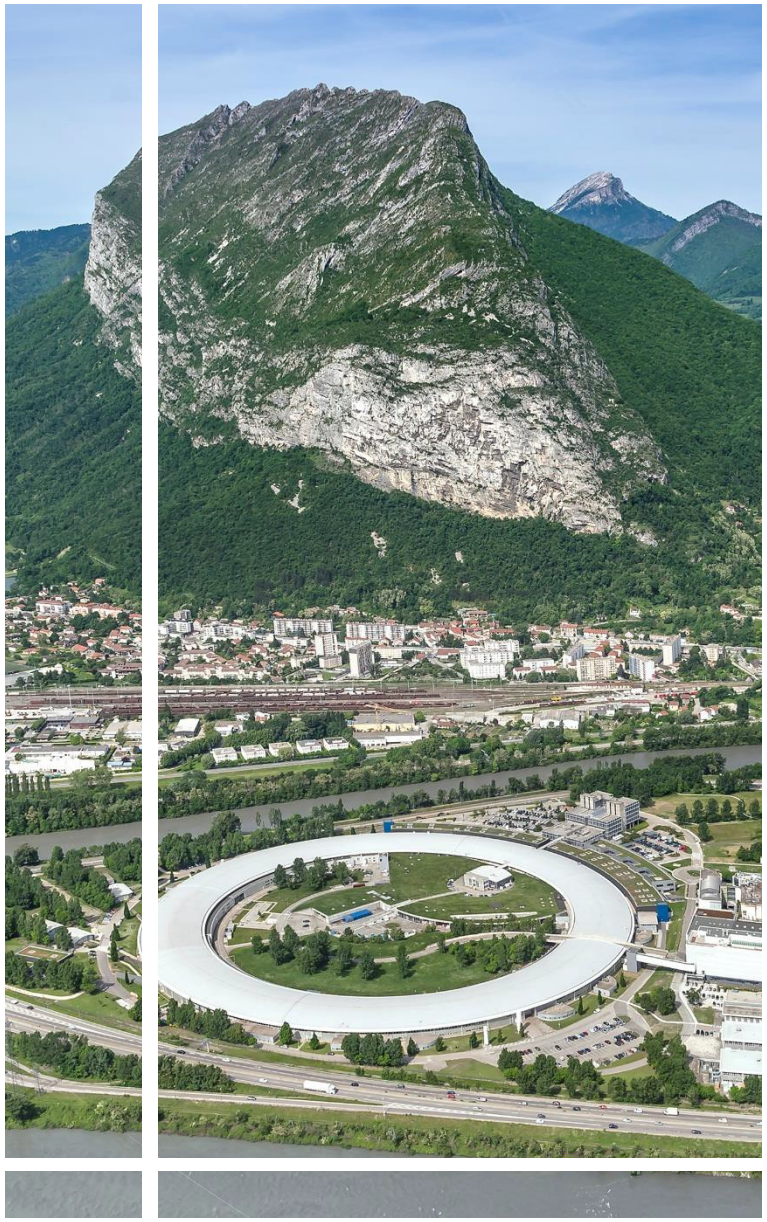
For example the *Structural Biology Services* at the ESRF (e.g. macromolecular crytallography) allow for completely automated experiments.

- custom designed workflow system that orchestrates the experiment and data analysis
- ISPyB based Laboratory Information Management System (LIMS) combining sample tracking and experiment reporting

Other attempts are all beamline specific with custom software.

- ➥ reinvent the wheel for every beamline
- ➥ no shared tools or technologies
- ➥ very labor intensive
- ➥ relies on a single expert per project (knowledge leaves when they leave)
- ➥ no professional devops (users cannot run the software at home)

DATA PROCESSING CHALLENGES

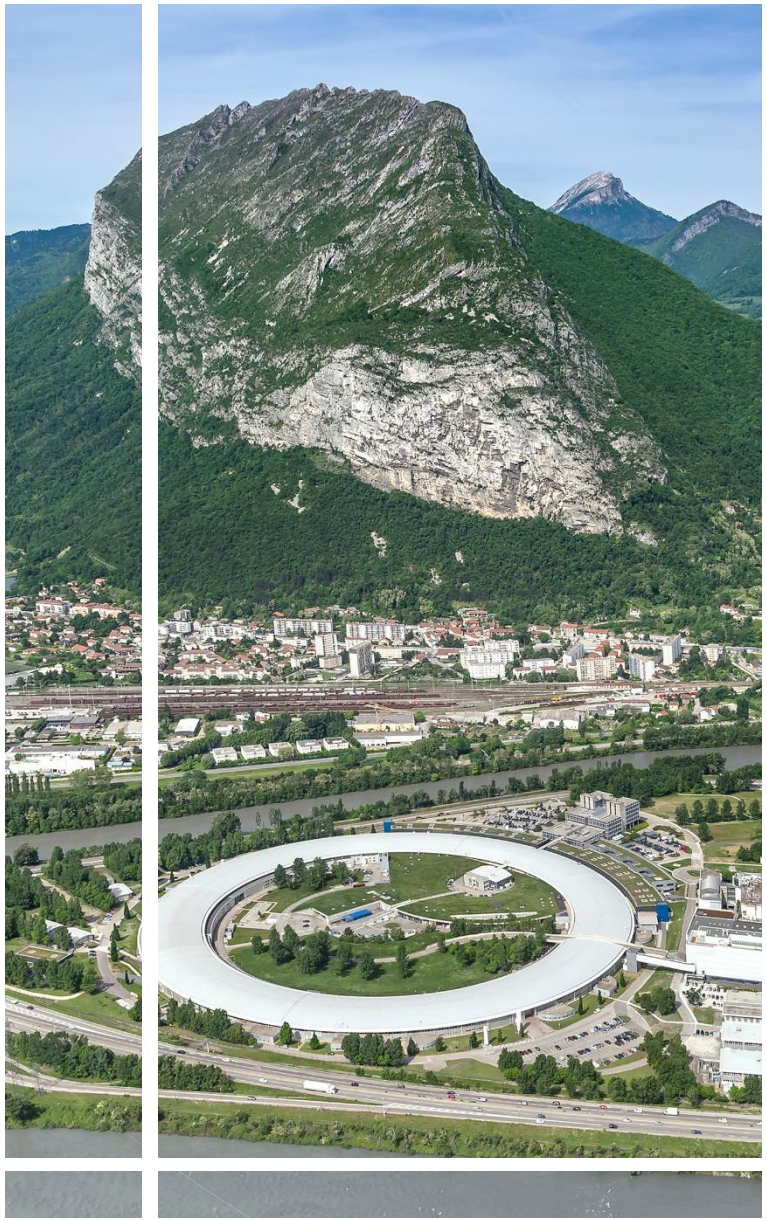


- 🗝 **Visualize the experiment** for humans
- 🗝 Produce **domain specific results** when possible
- ➥ Integrate scientific software in the experiment

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

- **easy to install** (runs everywhere from laptops to clusters)
- **reproducible** data processing (*executable data provenance document* saved with the results)
- **modular** (share common data processing steps, e.g. SAXS - WAXS, XRF mapping - XRF tomography, needs to evolve with the state-of-the-art)
- **interactive** vs. non-interactive
- different types of **interfaces** for humans and machines (GUI with plots and buttons, job scheduling on a cluster, web service)
- scientific software (often in python) needs to be **integrated**, not re-written
- can be provided as **service** of the facility maintained by a group, not reliant on individual experts

DATA PROCESSING CHALLENGES



- 🗝 **Visualize the experiment** for humans
- 🗝 Produce **domain specific results** when possible
- ➥ Integrate scientific software in the experiment

A **workflow-based solution** was chosen for the ESRF because

- A workflow is an **executable data provenance document**.
- It encapsulated the decisions taken by an **expert** so the learning curve for non-experts is less steep: “how do I use this workflow” not “how do I process this diffraction data”.
- A workflow can be **reused** to process other data (the “R” in FAIR).
- Workflows can be developed by experts and **maintained and deployed** by the facility.

META WORKFLOW SYSTEM

There are hundreds (yes, hundreds) of workflow systems

None have all features

- 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

META WORKFLOW SYSTEM

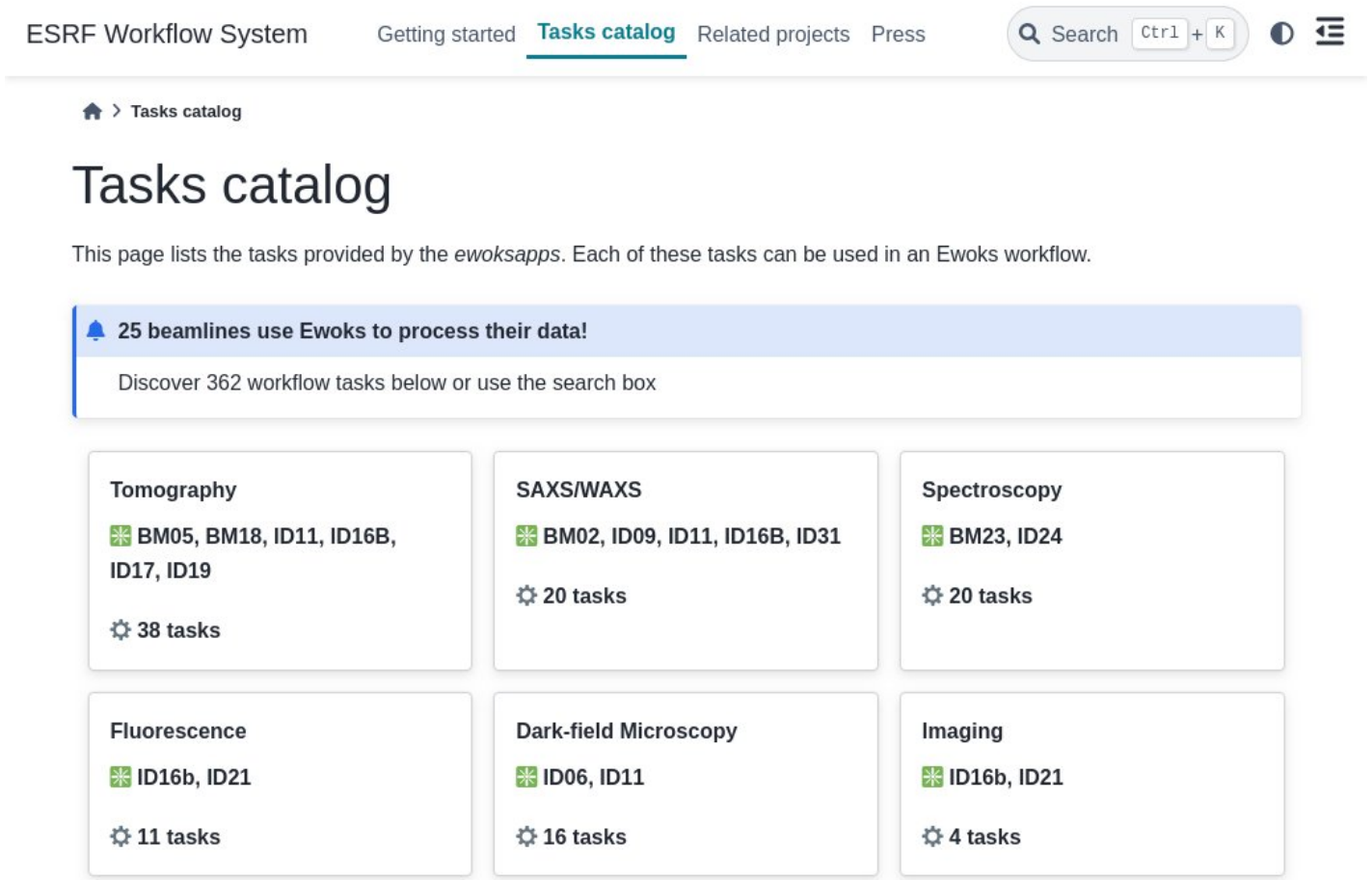
Meta workflow system: isolate the workflow definition and implementation from the workflow system

EWOKS
ESRF Workflow System
https://ewoks.esrf.fr

- Project started in July 2021. Stable since January 2023.
- 6 core developers in 2024
- Ewoks workflows are published under the FAIR principles so the ESRF is committed to long term support.
- Despite the name, nothing in the design makes it ESRF specific.
- MIT License
- Workflow systems currently supports:
  - Orange: desktop GUI
  - Dask: parallel + cluster
  - Pypushflow: parallel + loops

EWOKS tasks to be used in workflows are reusable and searchable

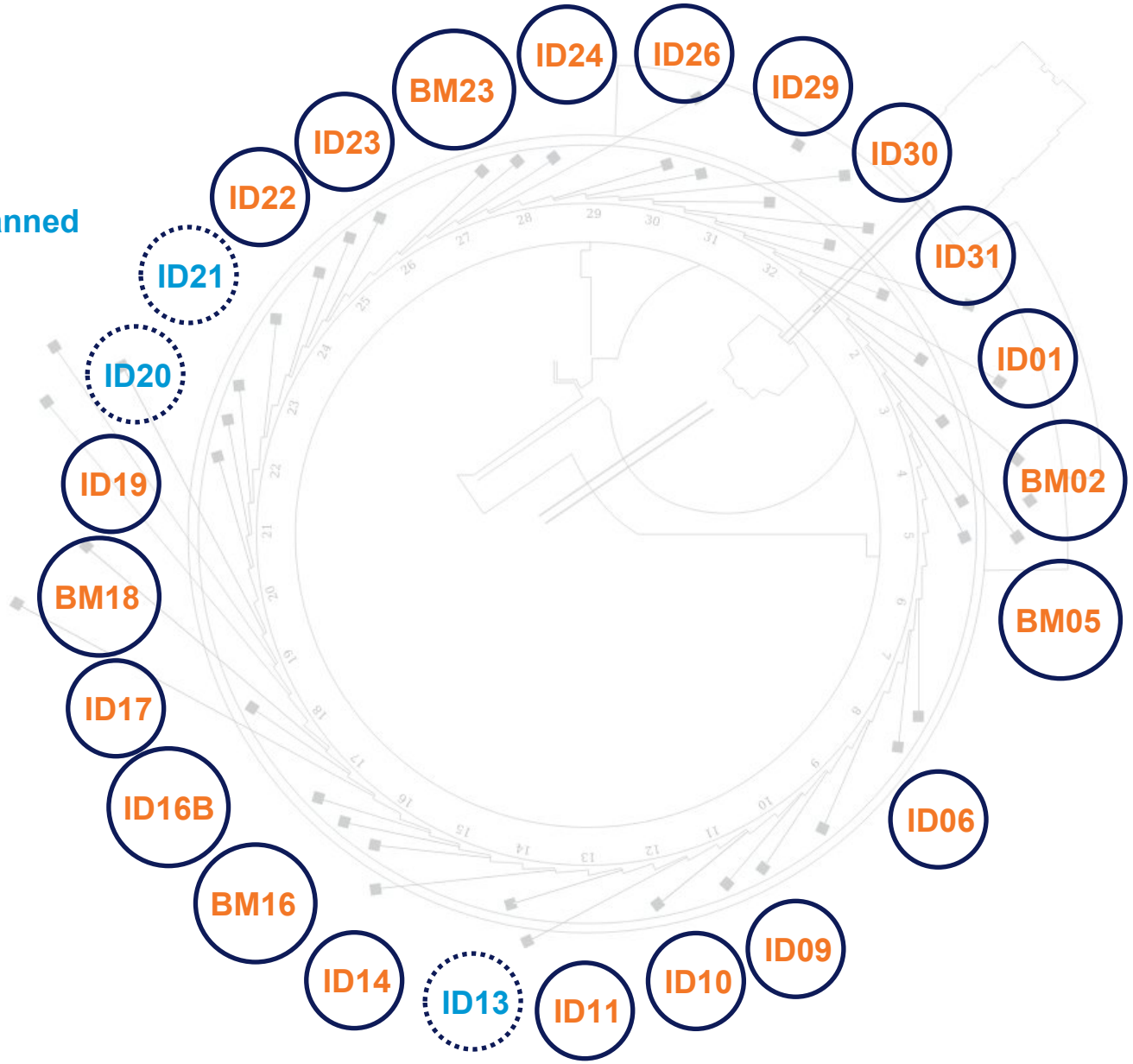
https://ewoks.esrf.fr



EWOKS

BM/ID In production

BM/ID In development/planned

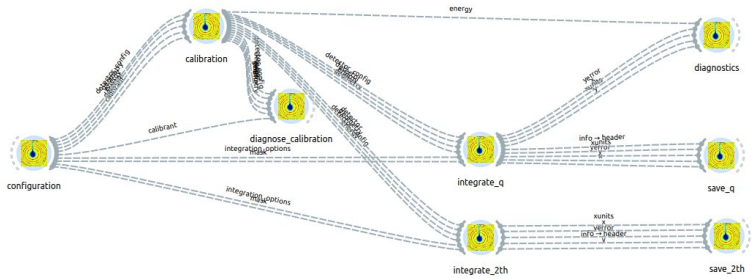


EWOKS

Typical setup for online data processing at the ESRF

Acquisition control system
(Bliss, Daiquiri, MxCube)

Execute EWOKS workflow on
- local machines (immediate feed)
- compute cluster



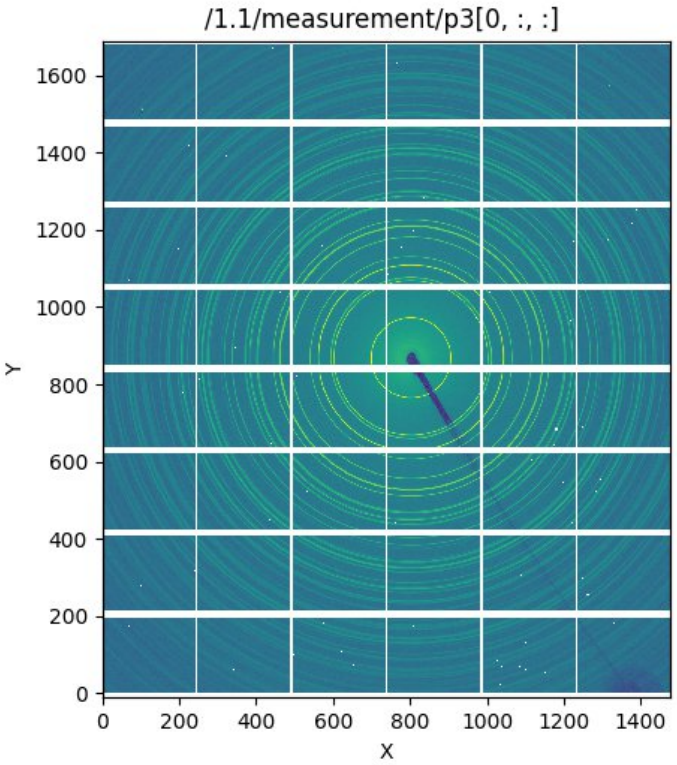
Visualization

Persist result for further analysis

Data portal
(FAIR results including the workflows with DOI for publication)

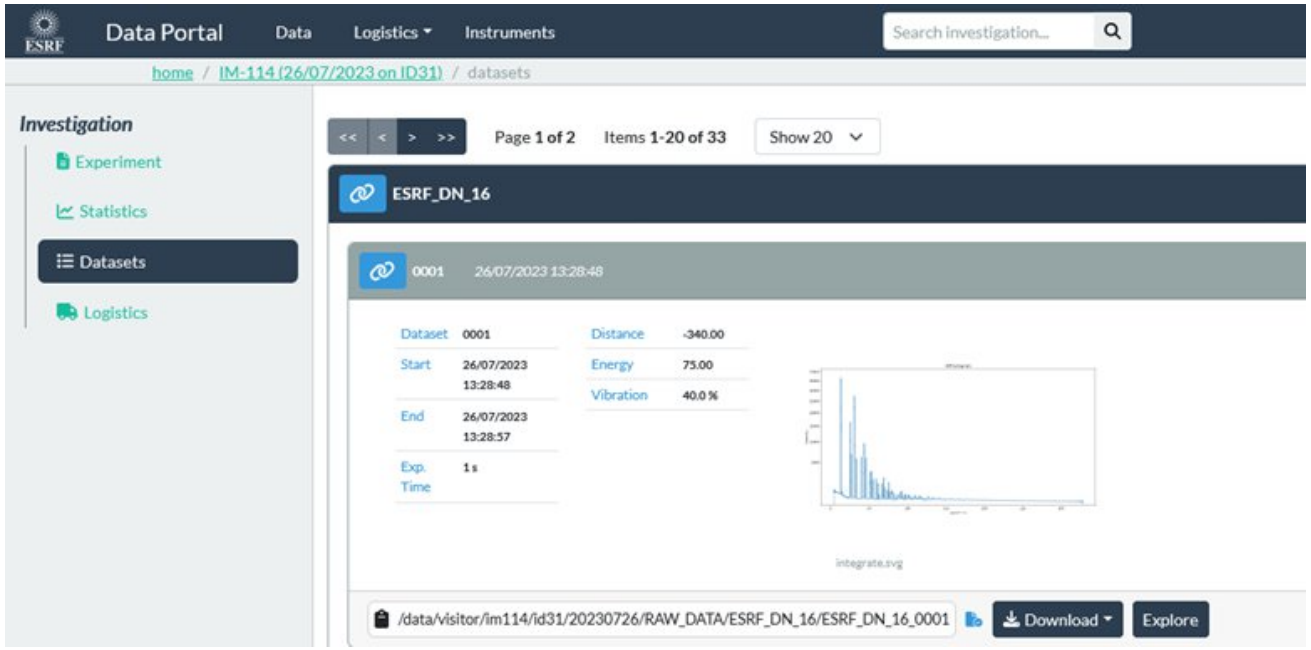
EWOKS

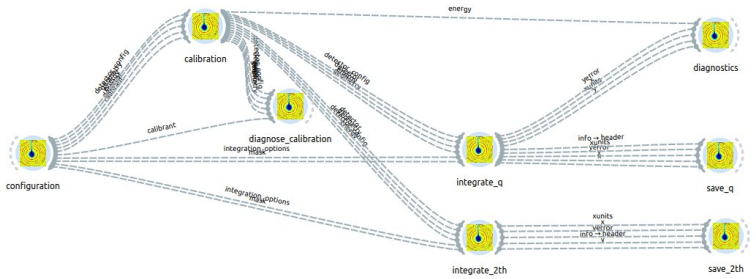
Diffraction (SAXS/WAXS): convert synchrotron data to traditional power diffractograms



- images + metadata (flux, energy, geometry)
- calibration (pyFAI)
- integration (pyFAI)
- save (HDF5)

EWOKS

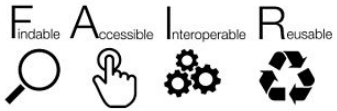


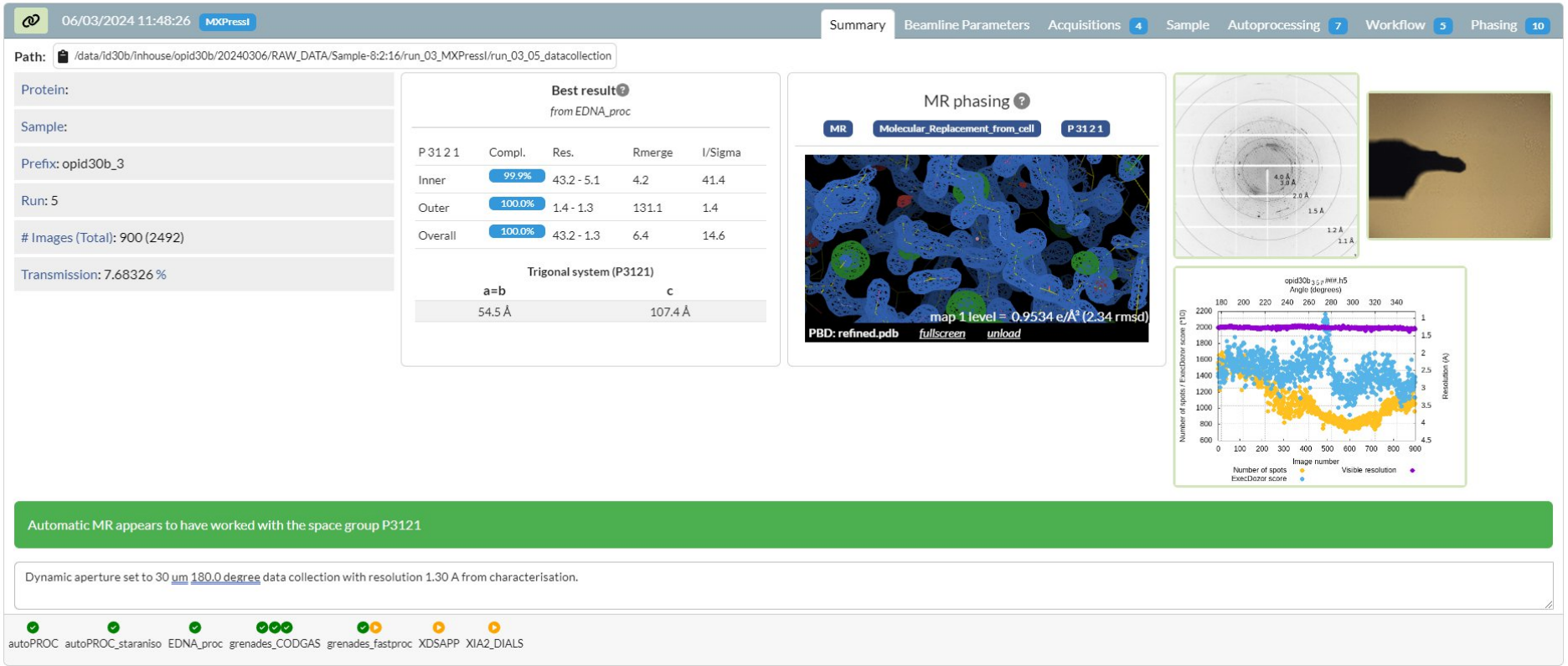


Further analysis to be brought online in the future

EWOKS

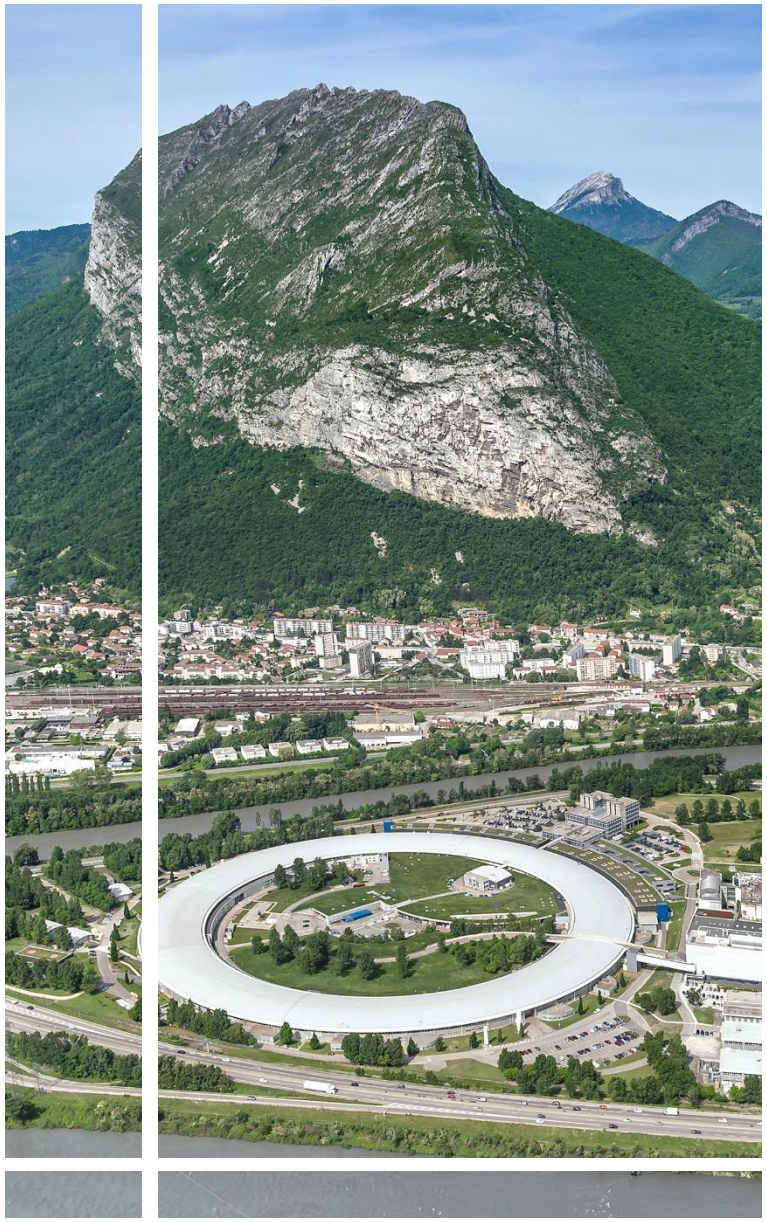
Diffraction (macromolecular crystallography): Automatic structure solution from single-crystal diffraction





Example where the complete analysis was brought online (completely automated with diagnostics for validation)

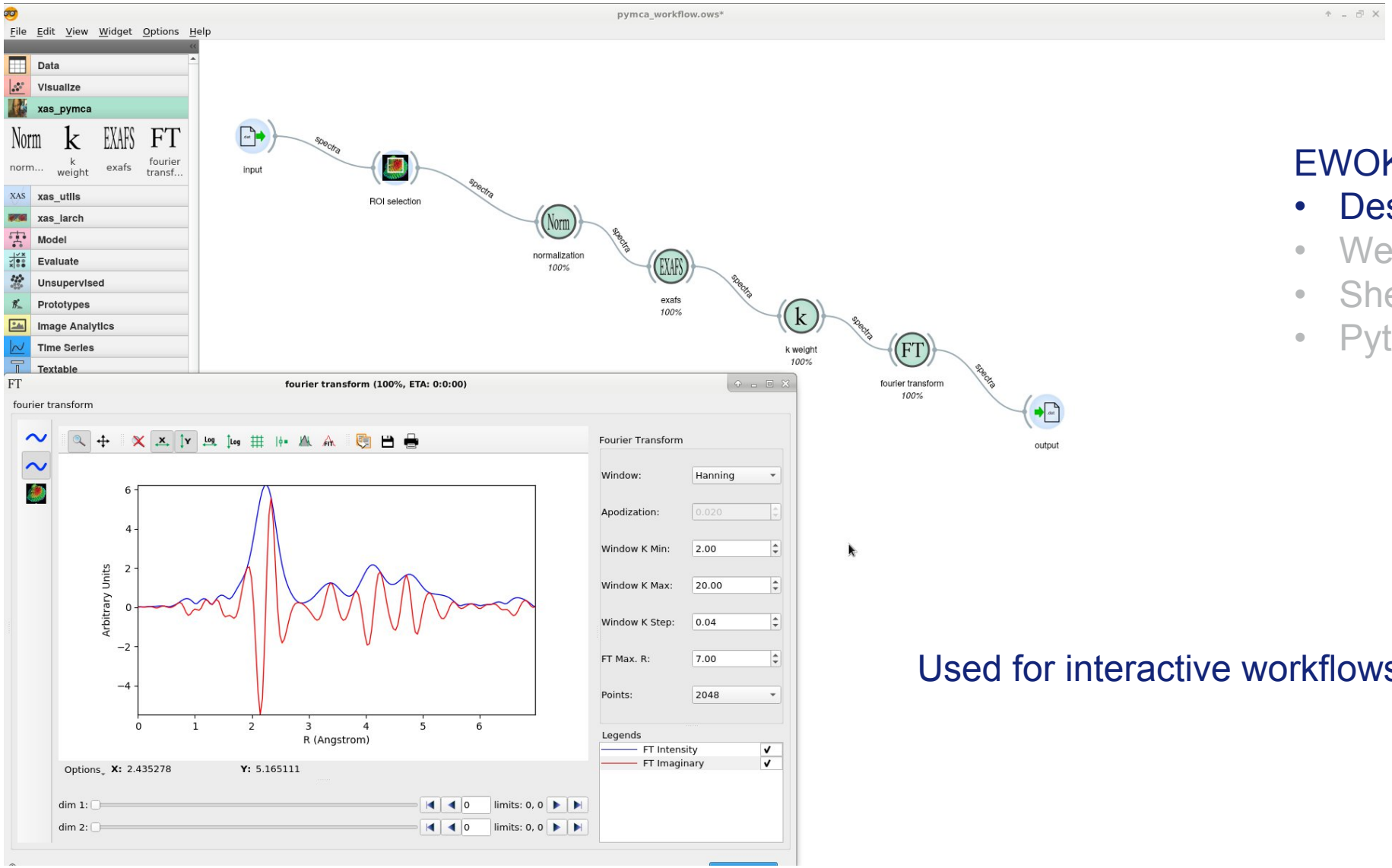
EWOKS



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



EWOKS interfaces

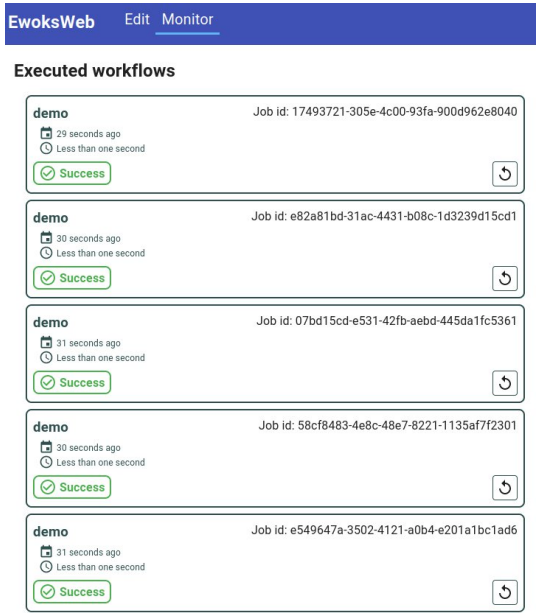
- Desktop
- Web
- Shell
- Python

Used for interactive workflows

EWOKS

EWOKS interfaces

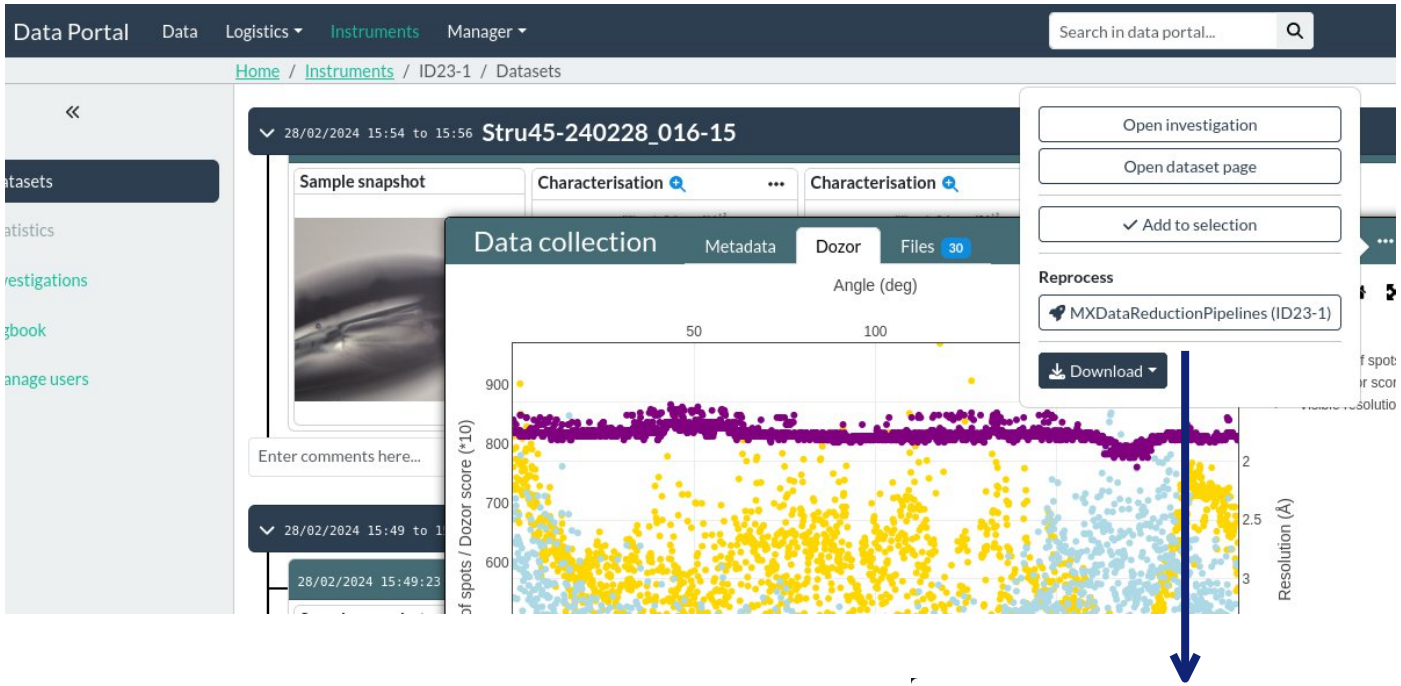
- Desktop
- Web
- Shell
- Python

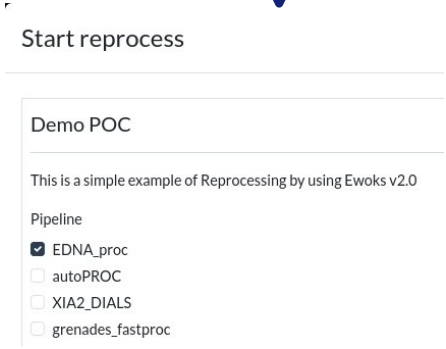


Used to visualize workflows that don't have graphical components

Standalone + frontend (similar to Jupyter notebooks)

EWOKS



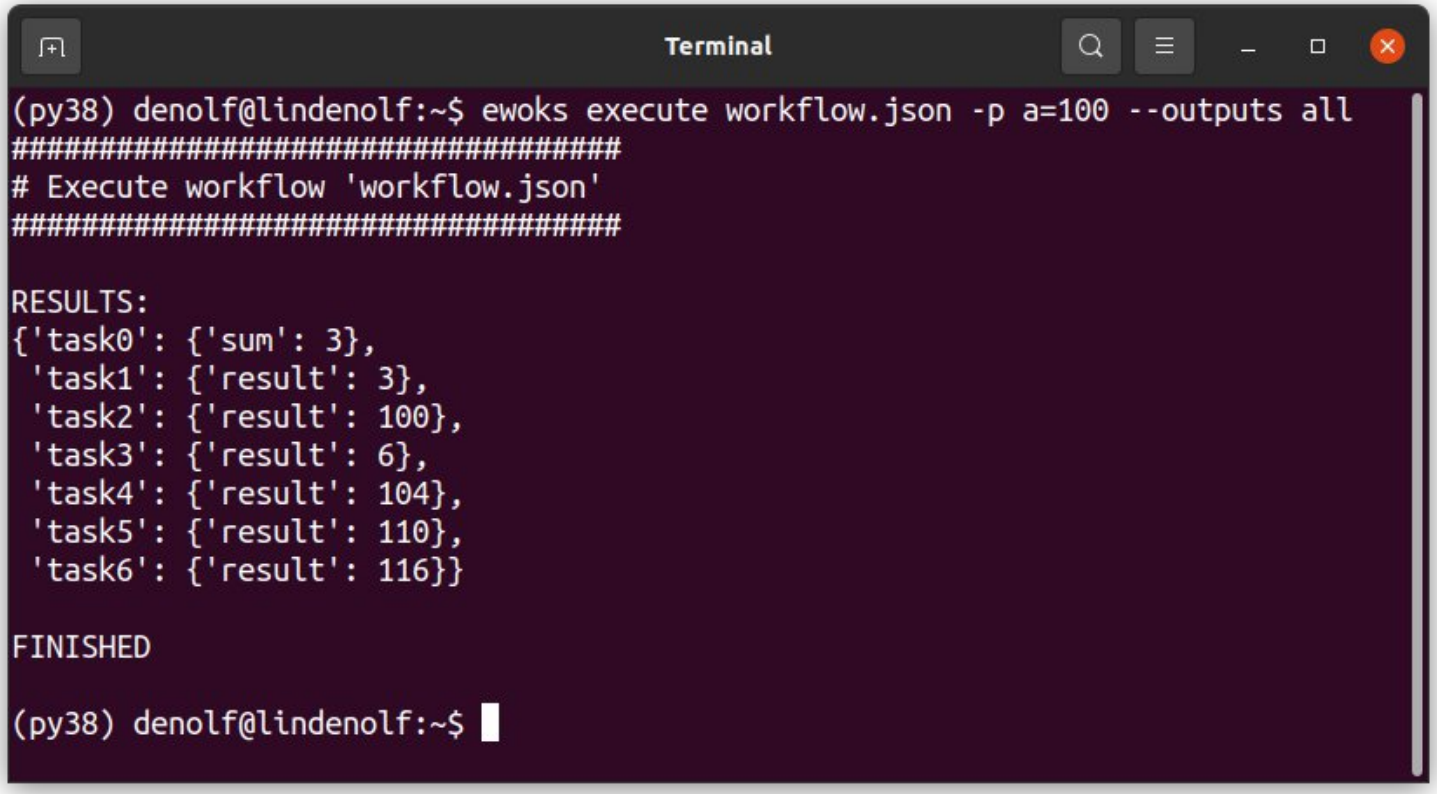


Web service used by other web services (e.g. ESRF data portal)

EWOKS interfaces

- Desktop
- Web
- Shell
- Python

EWOKS



EWOKS interfaces

- Desktop
- Web
- Shell
- Python

Used for headless execution

EWOKS

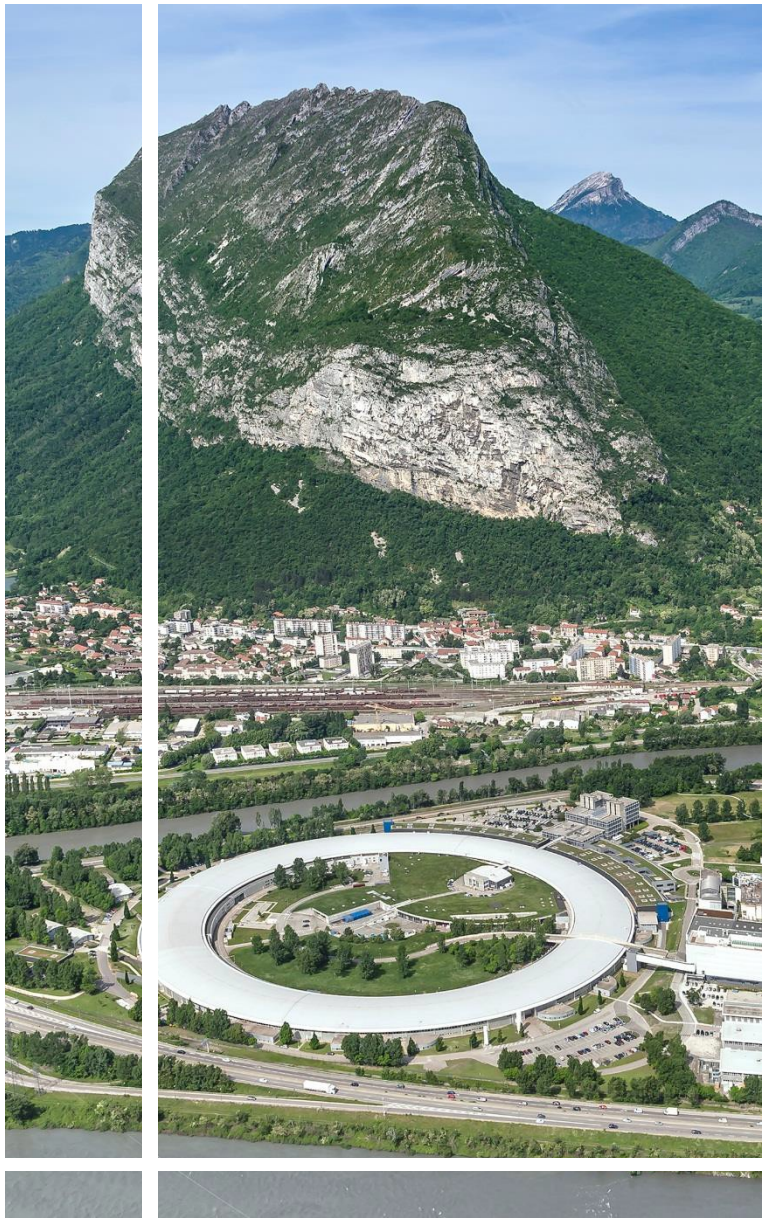
```python
from ewokscore import Task
from ewokscore import execute_graph

# 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)
```

EWOKS interfaces

- Desktop
- Web
- Shell
- Python

Developer usage like triggering workflows from the acquisition control system.

CONCLUSION



Increase the scientific output of the ESRF

- 🗝 **Visualize the experiment** for humans
- 🗝 Produce **domain specific results** when possible, X-ray specific otherwise
- ➥ Integrate scientific software in the experiment
- ➥ Workflow based solution (**executable data provenance**)
- ➥ **Meta approach**: decouple workflows and implementation from systems

Questions ?

https://ewoks.esrf.fr

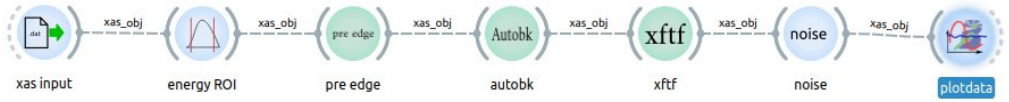
STREAMLINE has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 870313

https://streamline.esrf.fr/

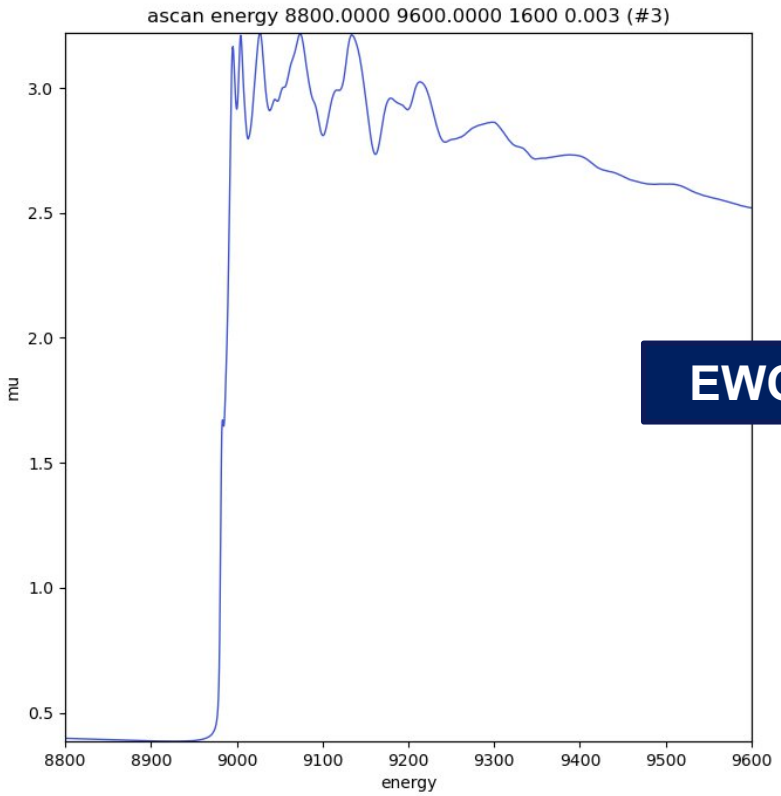
EWOKS

Spectroscopy: EXAFS visualization

workflow based on xraylarch

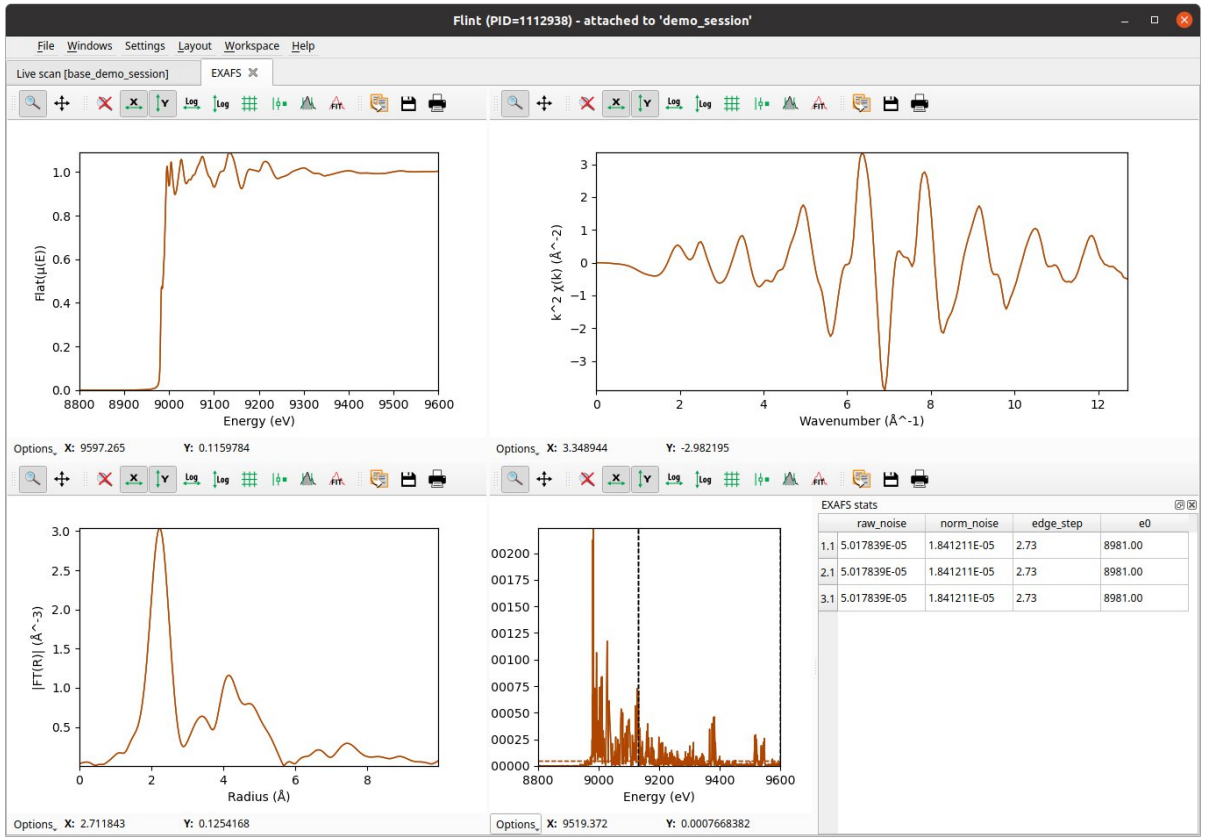


Raw Parameter Space



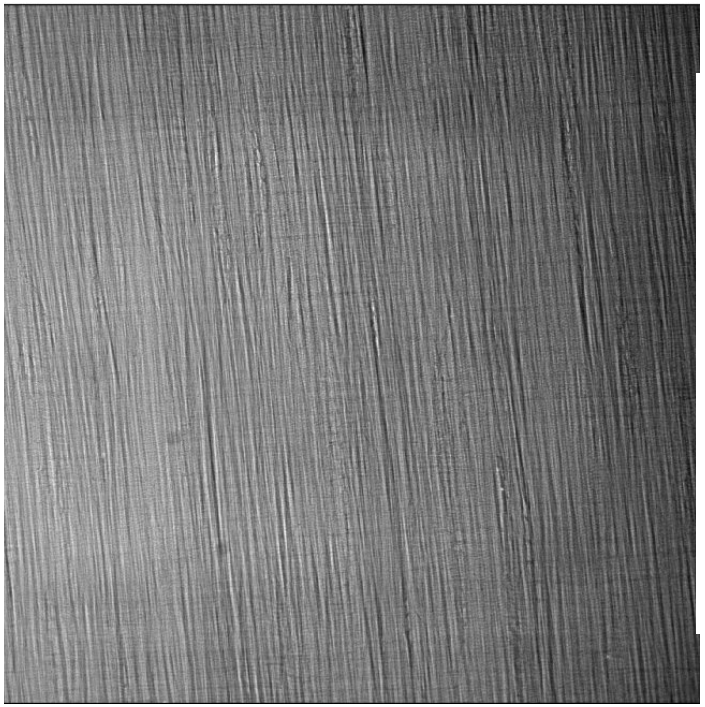
EWOKS

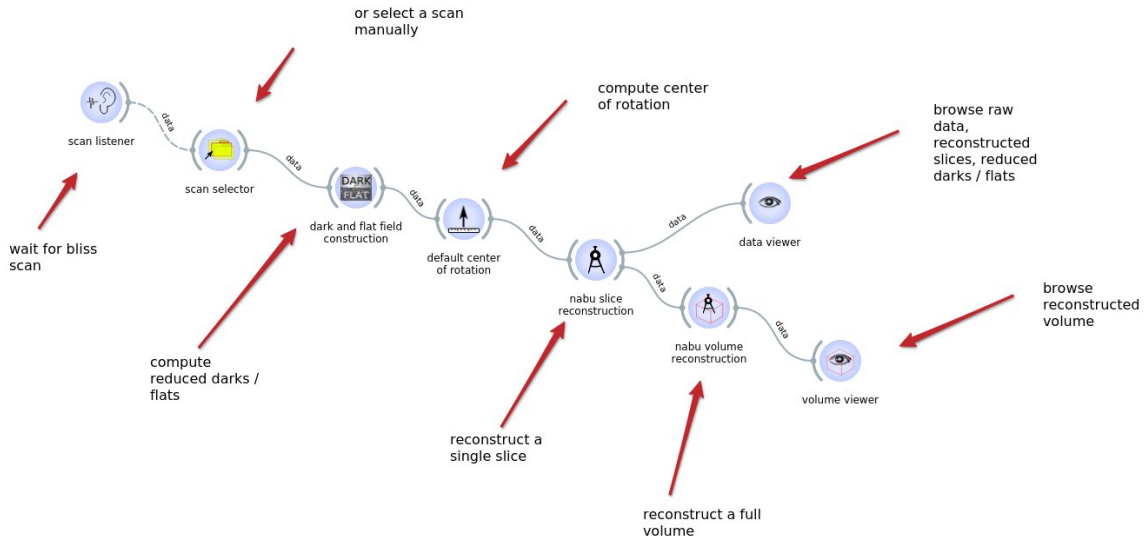
Parameter space in which scientific decisions are made

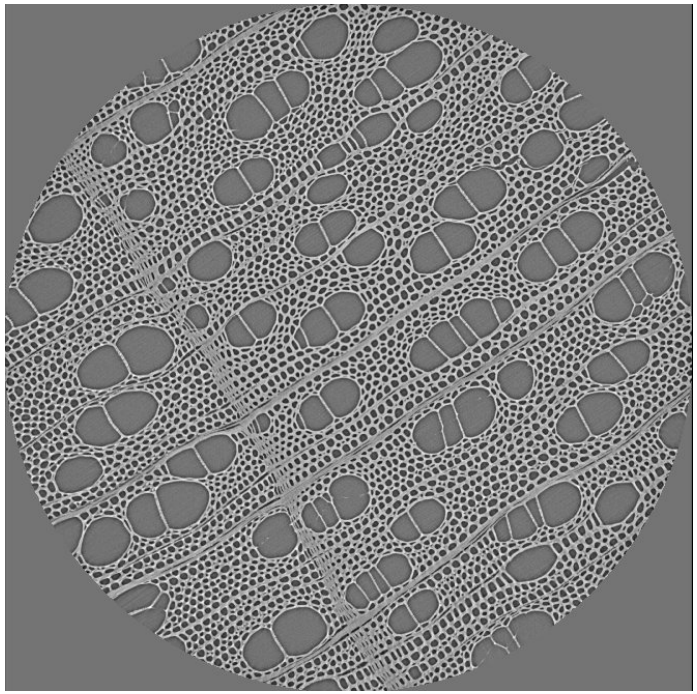


EWOKS

Tomography: 3D volume reconstruction of the X-ray density from projections







Inputs
Images of the sample in transmission + dark field images + flat field images + metadata

EWOKS

Outputs
Reconstructed volumes