Handling different analysis workflows in a modular framework

Malte Storm

Helmholtz-Zentrum Hereon, Institute of Materials Physics

deRSE 2024, Würzburg

Helmholtz-Zentrum hereon

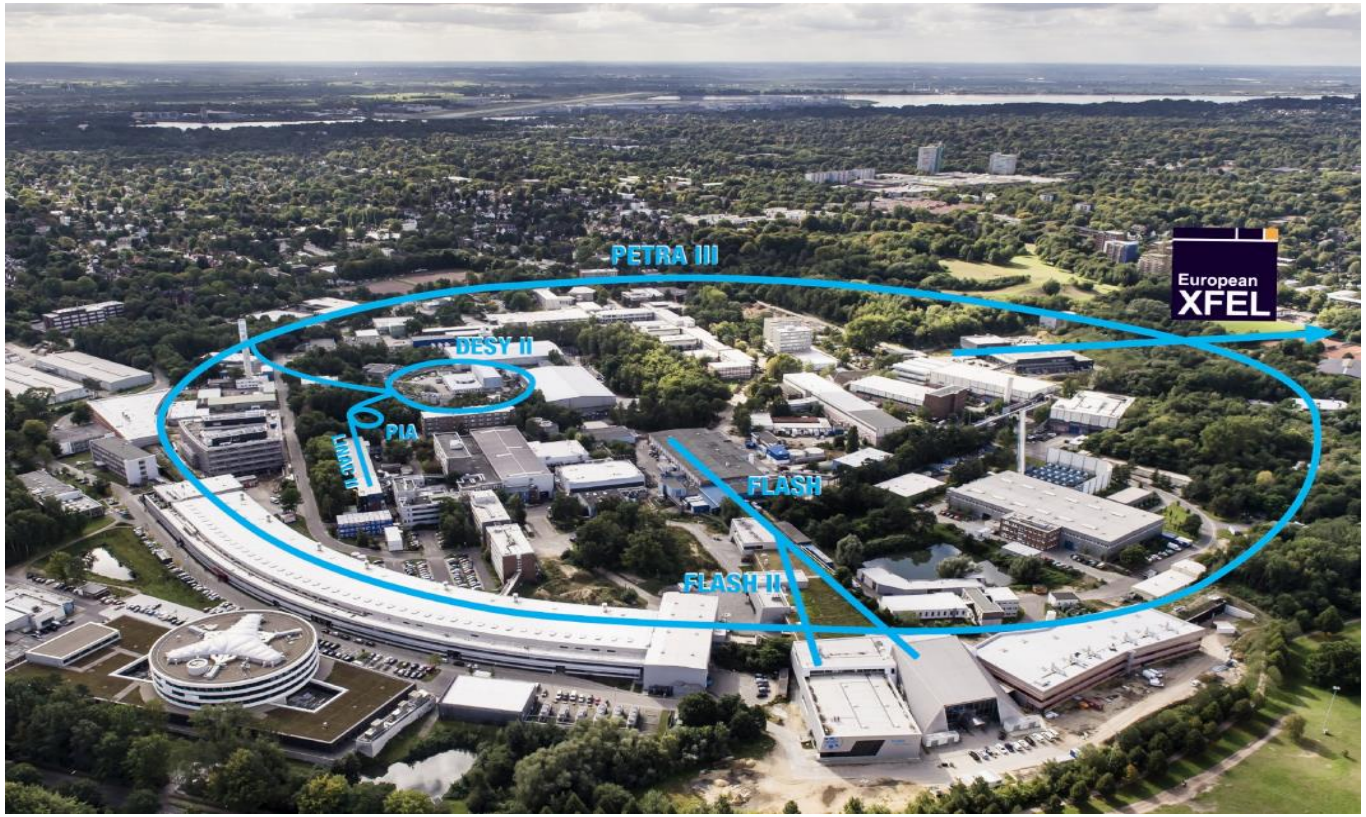
gems
German Engineering Materials Science Centre
Helmholtz-Zentrum hereon

Outline

- Motivation of the scientific background
- Individual components of the solution
- Interaction of components and global solution architecture
- Conclusion

gems
German Engineering Materials Science Centre
Helmholtz-Zentrum hereon

Motivation



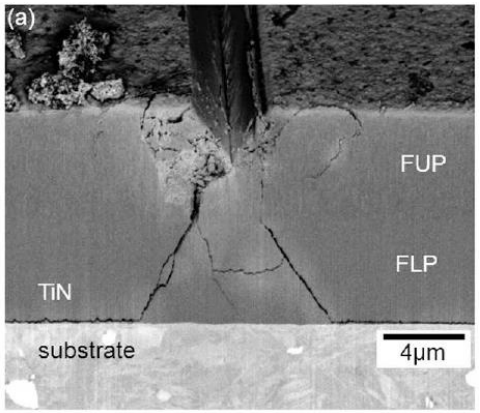
Helmholtz-Zentrum Hereon operates multiple X-ray experiments at the PETRA III storage ring (DESY, Hamburg)

- **In situ nano diffraction (P03)**
- **Nano tomography (P05 EH1)**
- **Micro tomography (P05 EH2)**
- **High energy diffraction (P07 EH3)**
- **High energy micro tomography (P07 EH4)**
- **White beam diffraction (P61A)**

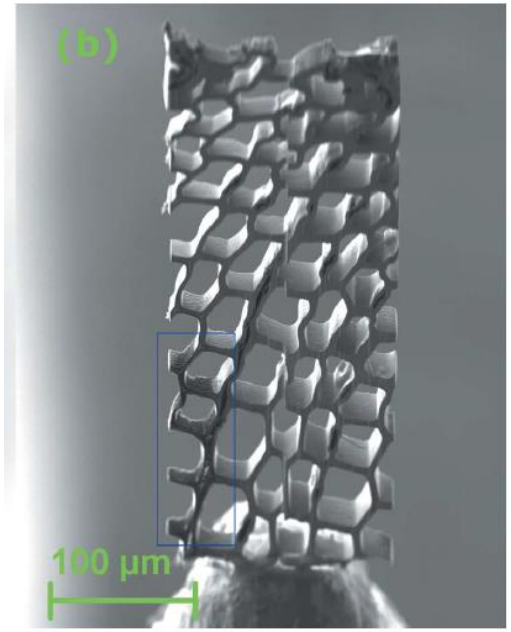
In this talk: Focus on analysis solution for diffraction experiments

gems
German Engineering Materials Science Centre
Helmholtz-Zentrum hereon

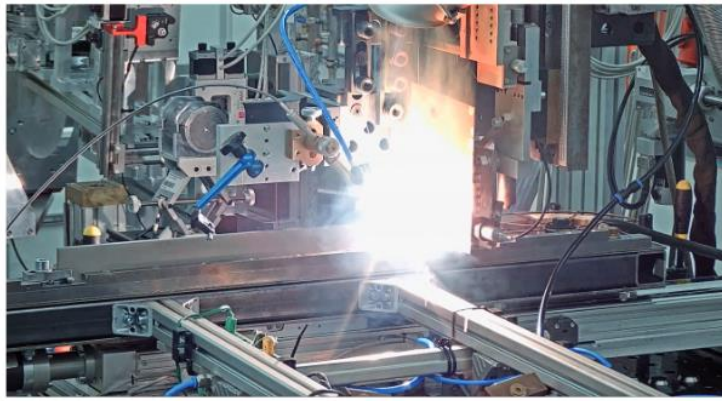
Exemplary samples



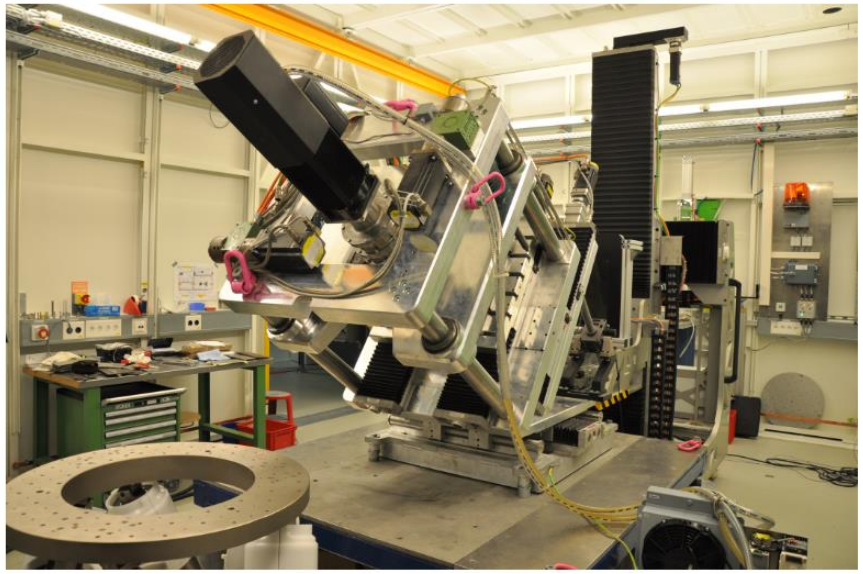
TiN steel coating for tools
Zeilinger et al.
Scientific Reports 6 (2016), 22670



Tracheid wood cells
Storm et al.
J. Synchrotron Rad. 22 (2015), 267



In situ laser beam welding
RWTH Aachen, 2022



FlexiStir in situ friction stir welding
Blankenburg, PhD thesis, Kiel 2016

gems
German Engineering Materials Science Centre
Helmholtz-Zentrum hereon

Motivation for new software solution

Political mandate to make experiments and analysis accessible for all domain scientists, not just experts in diffraction analysis

Old analysis pipeline example:

Raw data inspection (e.g. hdf5): python script / gui

Data calibration: pyFAI-calib2

Data processing: python script, Excel, …

Data visualization: python script / gui

Motivation for new software solution

Political mandate to make experiments and analysis accessible for all domain scientists, not just experts in diffraction analysis

Old analysis pipeline example:

Raw data inspection (e.g. hdf5): python script / gui

Data calibration: pyFAI-calib2

Data processing: python script, Excel, …

Data visualization: python script / gui

pydidas

gems
German Engineering Materials Science Centre
Helmholtz-Zentrum hereon

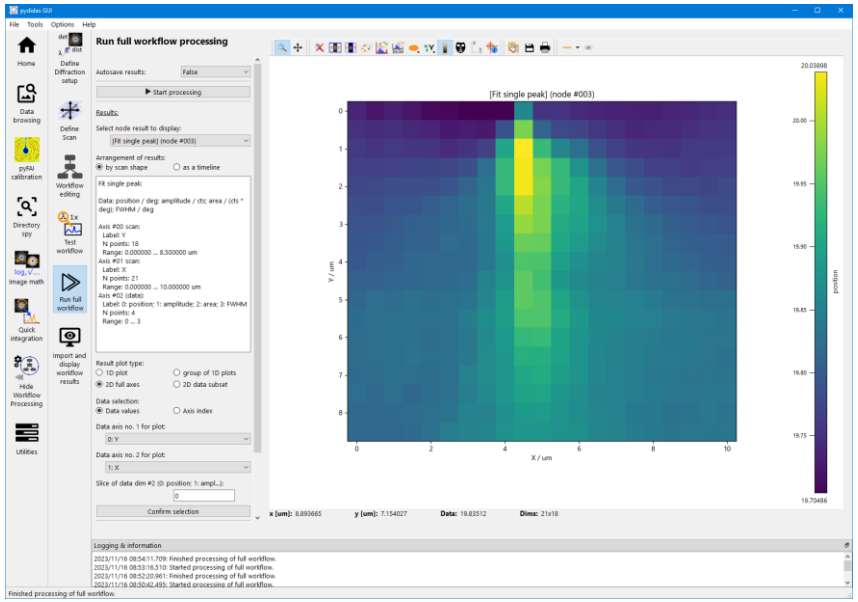
pydidas - Python Diffraction Data Analysis Suite

Key requirements:

- Easy to use and with graphical user interface
- Enable fast analysis during experiments
- Have a tool which is widely deployable and scalable

pydidas Python package with

- Graphical user interface
- Low-level access to all functionalities
  (e.g. for advanced scripting)
- Uses Qt for GUI, persistent configuration storage and communication



Pydidas workflow requirements

High flexibility required

- different processing strategies required for
  - different analysis goals
  - different materials
  - different experimental conditions
- Generic functionality sufficient for routine experiments but specialist analysis methods required in some cases
  → implemention to extend generic functionality required

→ Workflow implemented based on a tree and plugins.

gems
German Engineering Materials Science Centre
Helmholtz-Zentrum hereon

Plugins

Plugins structure must be simple to allow non-experts to extend functionality

Minimum requirements:

- Classes must inherit from base classes for automatic discovery
- Some basic class attributes (plugin metadata)
- `pre_execute` method
- `execute` method

Can be extended with e.g.

- Custom configuration widgets
- Storage of detailed intermediate results

```python
class CropAndBinImage(ProcPlugin):
    plugin_type = PROC_PLUGIN
    plugin_subtype = PROC_PLUGIN_IMAGE
    basic_plugin = False
    plugin_name = "Crop and bin image"
    output_data_label = "Image intensity"
    output_data_unit = "counts"
    input_data_dim = 2
    default_params = get_generic_param_collection(
        *param_keys: "use_roi",
        "roi_xlow",
        "roi_xhigh",
        "roi_ylow",
        "roi_yhigh",
        "binning",
    )

    def __init__(self, *args: tuple, **kwargs: dict):
        super().__init__( *args: *args, **kwargs)
        _params = self.get_params(
            *param_keys: "use_roi", "roi_xlow", "roi_xhigh", "roi_ylow", "roi_yhigh", "binning"
        )
        self._image_metadata = ImageMetadataManager(*_params)

    def calculate_result_shape(self):
        if self._config["input_shape"] is None:
            raise UserConfigError(
                "No input shape has been given for the *CropAndBinData* plugin."
            )
        self._image_metadata.store_image_data(self._config["input_shape"], int, n_image: 1)
        self._image_metadata.update_final_image()
        self._config["result_shape"] = self._image_metadata.final_shape

    def execute(self, data: Dataset, **kwargs: dict) -> tuple[Dataset, dict]:
        _roi = self._image_metadata.roi
        _binning = self.get_param_value("binning")
        if _roi is not None:
            data = data[_roi]
        if _binning > 1:
            data = rebin2d(data, _binning)
        return data, kwargs
```

gems
German Engineering Materials Science Centre
Helmholtz-Zentrum hereon

Plugins access

Plugins are made accessible through a registry

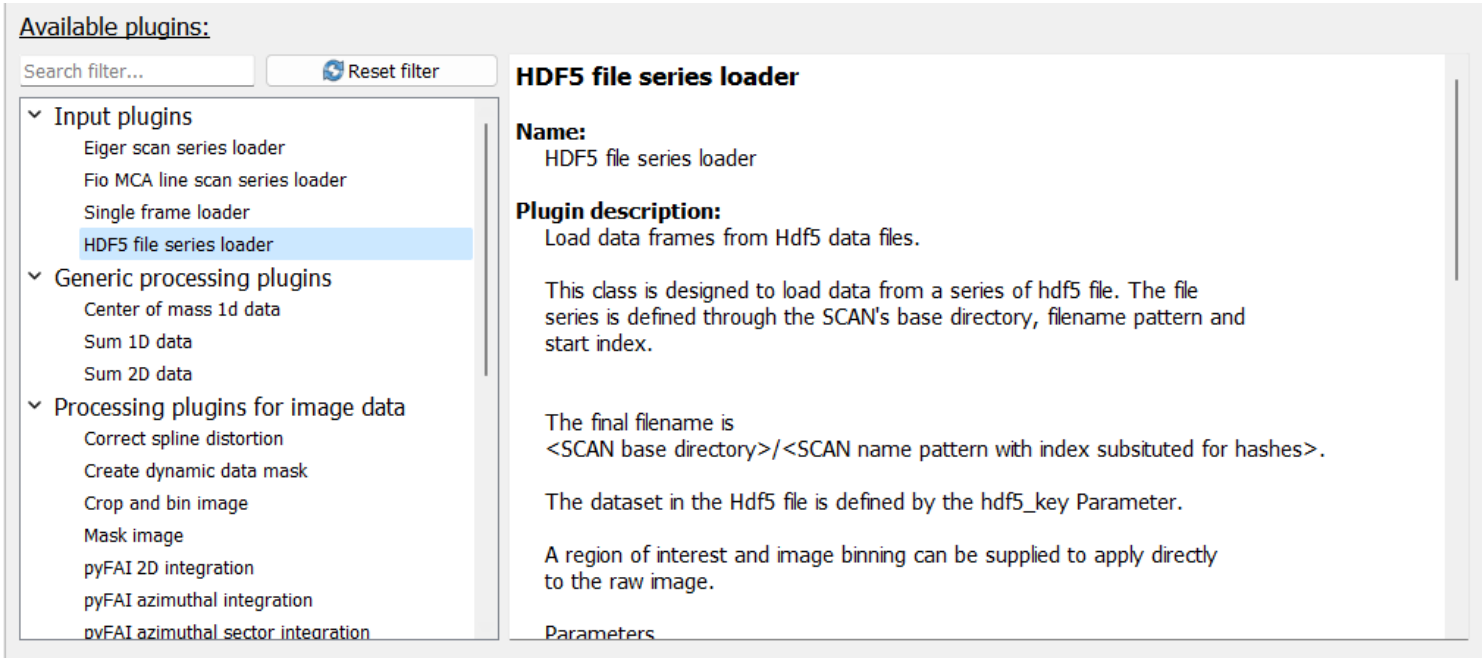
- **Plugin paths are managed through a Qsettings variable**
- **Registry scans all python files in plugin paths and stores classes which inherited from `BasePlugin`**
- **Registry offers access to**
  - Plugin classes
  - Plugin names (also filtered by type)
  - Stored paths

gems
German Engineering Materials Science Centre
Helmholtz-Zentrum hereon

Plugins access in the UI

Widget for browsing plugins

- Display all available plugins
- Search for plugins
- Display additional information about plugins



ProcessingTree and WorkflowNode

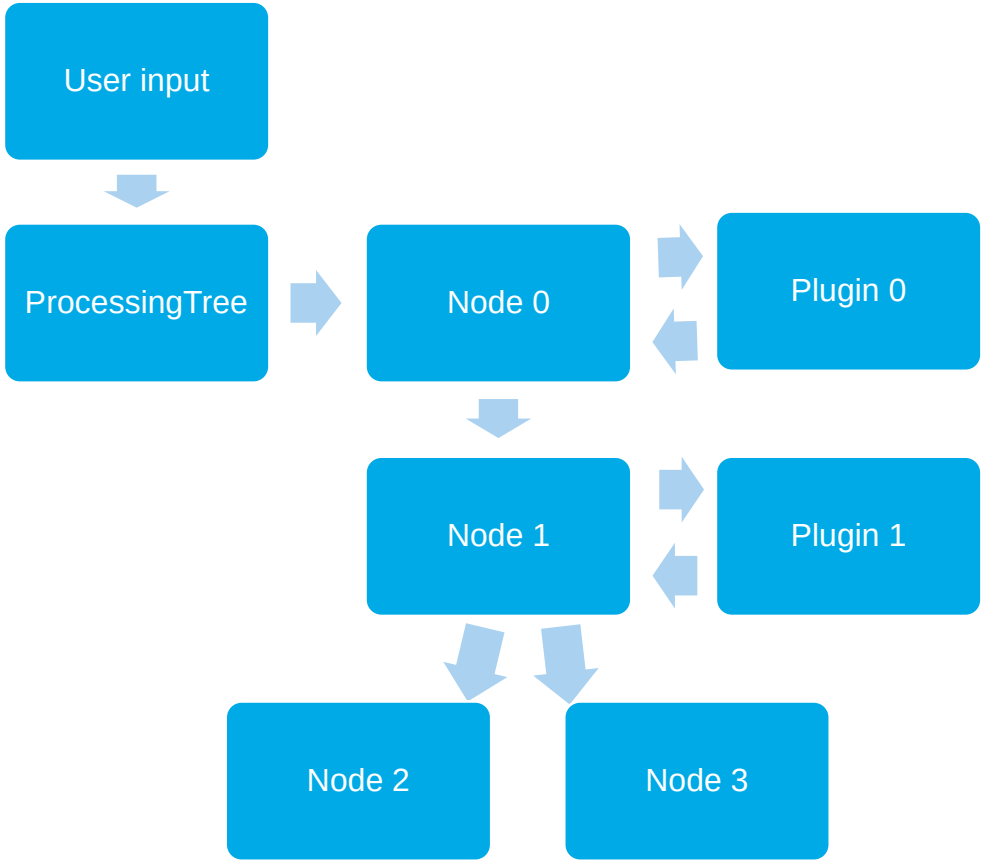
- **ProcessingTree: Controller class for nodes**
- **WorkflowNodes: Nodes to populate the ProcessingTree and which include a plugin**

ProcessingTree responsibilities:

- **Import and export**
- **Managing nodes (incl. addition / removal)**
- **Start of processing**

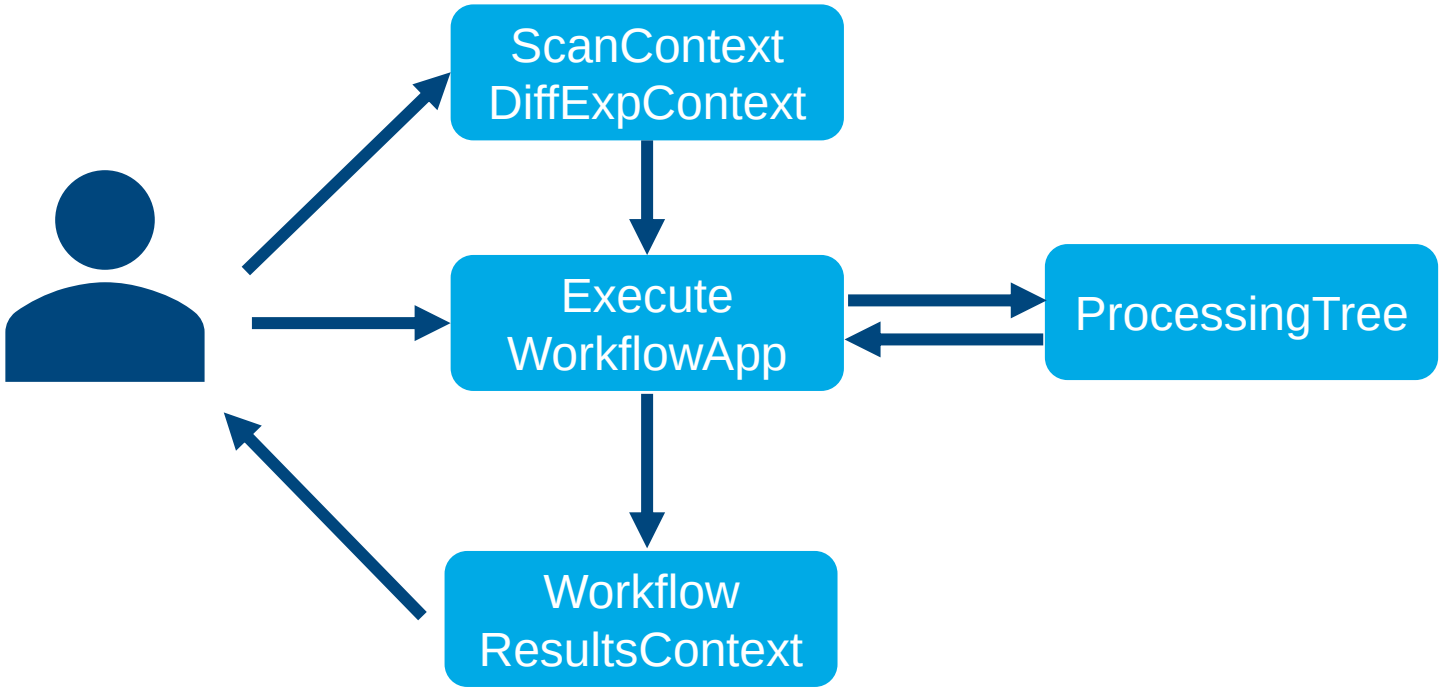
WorkflowNode responsibilities:

- **Handling of data transport**
- **Plugin execution**
- **Knows parent and children**



Running Workflows

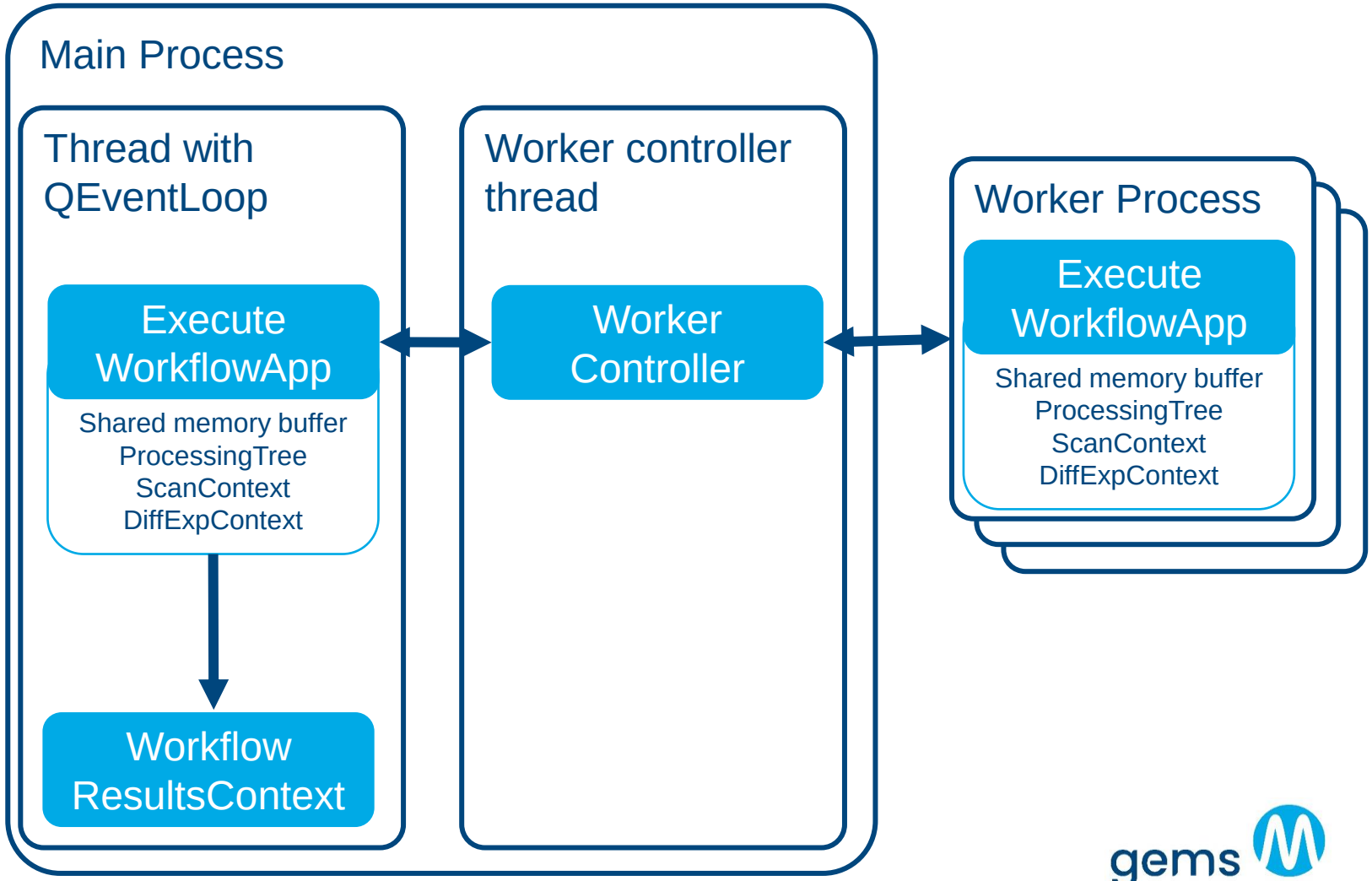
- **ScanContext and DiffractionExperimentContext handle experimental metadata**
- **WorkflowResultsContext stores results**
- **ExecuteWorkflowApp handles batch processing and result storage**



Parallelization

- **Communications with worker processes through queues**
- **Communication between WorkerController and main ExecuteWorkflowApp with Qt‘s signals / slots**

- **Only memory pointers communicated, data shared through shared memory buffer**



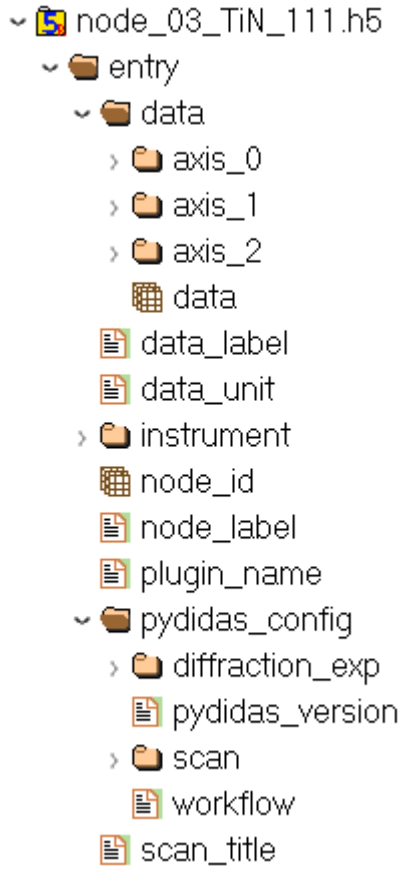
Pydidas results: Metadata

Internal data handling:

- **Data stored in subclassed np.ndarray to have metadata attached to raw data**

Exporting data:

- **Exports only in hdf5 file format (for now)**
- **All data axes are labelled and ranges are stored**
- **All processing metadata (including contexts) is stored in results file**



FAIR principles

- Pydidas is open source (GPL 3 license)
- Full architecture is designed for reusability:
  - Components designed with separation of concern in mind
  - Processing logic separated from UI
  - Dynamic UI: Composed of frames;
    frames usage can be modified in startup script

- To improve: Separate generic components from XRD-specific components to allow re-use with only necessary dependencies

Conclusion

- **Pydidas was develop with X-ray diffraction analysis in mind but framework versatile enough for other applications**
- **In production and used by users, feedback very positive**
- **Hereon is committed to maintaining and improving pydidas**
- **Goal is to build a larger community to extend functionality further**

gems
German Engineering Materials Science Centre
Helmholtz-Zentrum hereon

Thank you for your attention

Any questions?

gems
German Engineering Materials Science Centre
Helmholtz-Zentrum hereon