Automation Status and Plans – Accelerator Side

HELMHOLTZ

Frank Mayet – DESY/MXL
For the operation and software teams

Operation Workshop 2025
November 13th, 2025

HLC and MXL Power the Software Behind EuXFEL Operations at DESY

HLC (High Level Controls) team

- A group of physicists and software developers from various groups in the M-division at DESY
- The backbone of the XFEL operations software as well as the majority of the servers and tools come from this group
- Hosts a bi-weekly meetings to discuss ongoing work as well as requests concerning accelerator operations
- Provides guidelines for software development concerning accelerator operation at DESY
- Most colleagues from the HLC-group are also involved in software projects not related to EuXFEL

MXL group software team

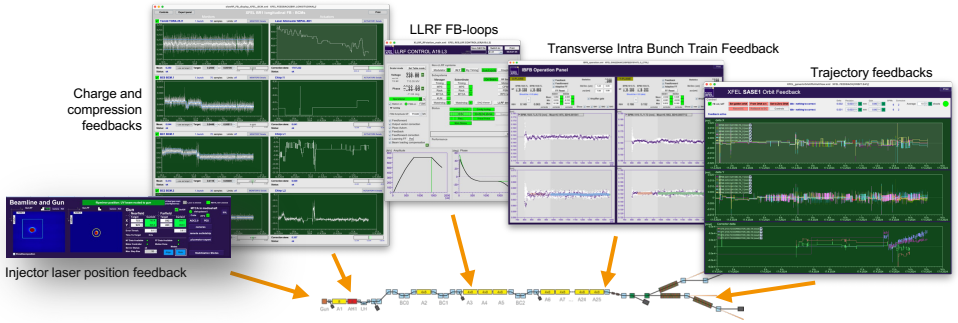
- These colleagues work exclusively for EuXFEL
- Currently one fulltime software developer and one staff scientist (automation / ML/AI deployment)
- Several scientists working on tools and scripts for operation. Partially on final operation versions, partially on test versions that are then handed over to software developers

More Automation Benefits Machine Operation

- **Reduce complexity for the operations team**
  - Parallel operation of three beamlines; weekly changing requirements / regular parameter changes
  - LLM-based tools, smart operation panels, and logbook enhancements
- **Reduce dependence on experts**
  - Much of the expert knowledge can be provided by / modelled in software tools
  - LLM-assisted knowledge retrieval
- **Ensure consistently high performance (intensity and stability)**
  - Standardized software procedures benefit reproducibility
  - ML-based anomaly detection and virtual diagnostics (shot-to-shot monitoring of virtual FOMs)
- **Reduce downtimes after failures**
  - The sequence of recovery steps can be optimized such that the resulting downtime is as short as possible
  - Automated, standardized setup / optimization procedures
- **Explore new operation modes**
  - Leveraging machine learning based optimization, as well as virtual diagnostics and surrogate models

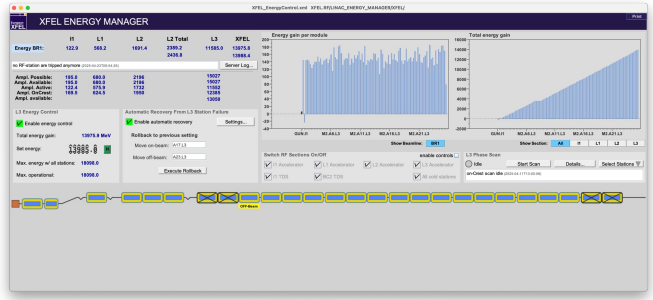
Mature Automation Tools drive Daily Operation

Feedback loops



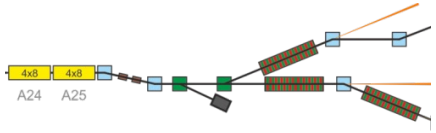
Finite State Machine and Energy Manager

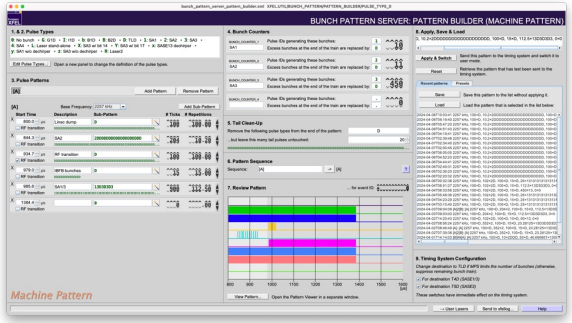
- Automatic RF station recovery
- Automatic RF and magnet parameter scaling
- Spare station handling and parasitic on-crest phase measurements



Bunch Pattern Builder

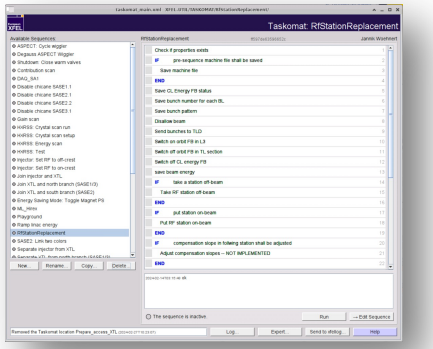
- Highly customized bunch patterns for all three beamlines





Taskomat

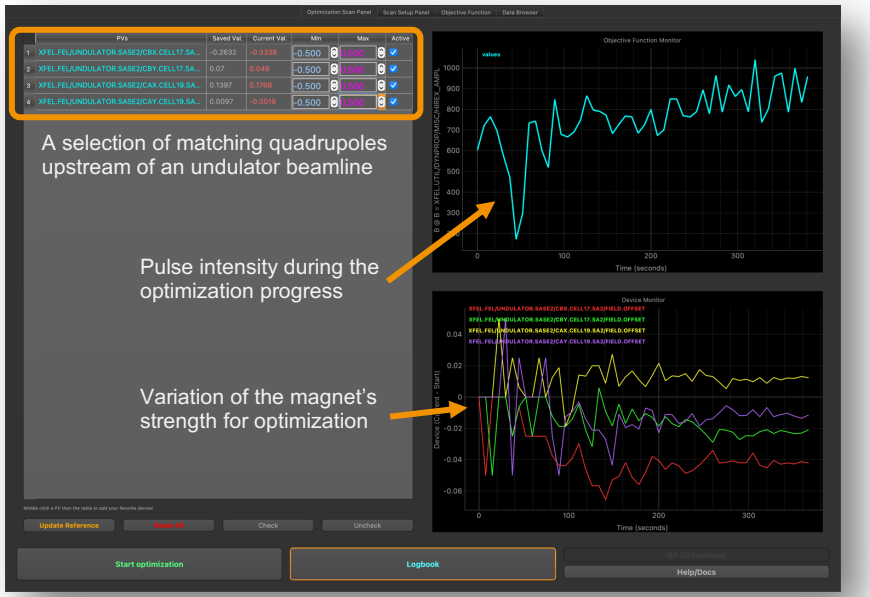
- Powerful generic sequencer implemented as a server
- Very easy to use and to develop new sequences using LUA
- Not only serial procedures, but also more exotic (coupling of two colors, etc..)



Mature Automation Tools drive Daily Operation

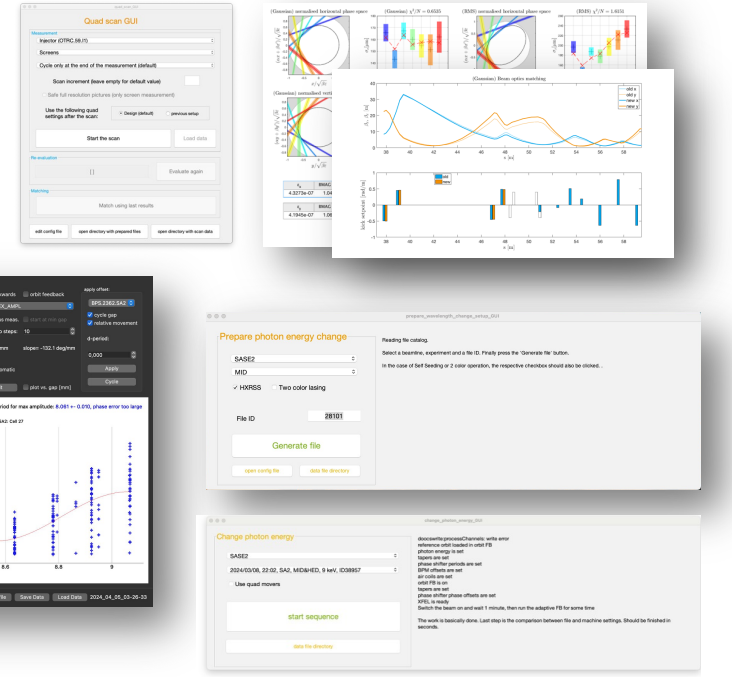
Ocelot Optimizer

- Drag and drop DOOCS properties for actuators and monitors
- Many presets are available for standard tuning
- Optimization routines can be freely programmed via Python and started e.g. for dispersion minimization



Many more…

- Emittance measurement and beam optics matching
- Parasitic longitudinal phase space characterization
- Photon energy change
- Phase shifter scans
- Etc.



Software Developments in Service of Automation

- **doocs4py**
  - Python bindings to the C++ DOOCS server and client libraries
  - Feature complete DOOCS servers written in Python
  - Speeds up transfer of (mostly Python-based) ML/AI projects into operation
  - Easy conversion of Python GUI tools into servers
  - Install via pip

A complete DOOCS server written with doocs4py

```python
#!/usr/bin/env python
from doocs4py import Server, Location, location_class
from doocs4py.utils import print_to_stderr
from doocs4py.properties import PropertyFloat, PropertyText

@location_class(42)
class ServerLocation(Location): #EqFct
    def __init__(self):
        super(ServerLocation, self).__init__("NAME")

        # float property
        self.float_prop = PropertyFloat(self, "FLOAT")

        # text property
        self.text_prop = PropertyText(self, "TEXT")

        print_to_stderr(self.get_name(), message="ServerLocation constructor")

    def init(self):
        print_to_stderr(self.get_name(), message="ServerLocation.init()")

    def update(self):
        print_to_stderr(self.get_name(), message="ServerLocation.update()")


server = Server("simple_server")
server.register_location_class(ServerLocation)
server.run()
```

Software Developments in Service of Automation

Example: Badger Server

- **DOOCS server implementation of the Python-based Badger Optimizer 1.4.3**
  - Uses doocs4py
  - Runs pre-defined optimization routines, which are created using the Badger GUI
  - Exposes key scan parameters and control properties to the control system
  - Can be controlled via a standard control room panel, or from other servers/tools
  - Ocelot optimizer still relevant for ad-hoc optimization
  - Will also be able to run RL agents in the future

In collaboration with SLAC Xopt

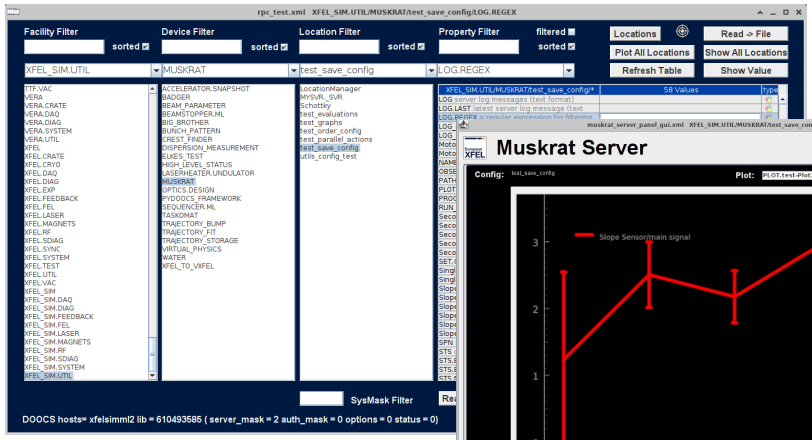
The control room panel can be used to control and monitoring

Badger GUI used as an optimization routine editor

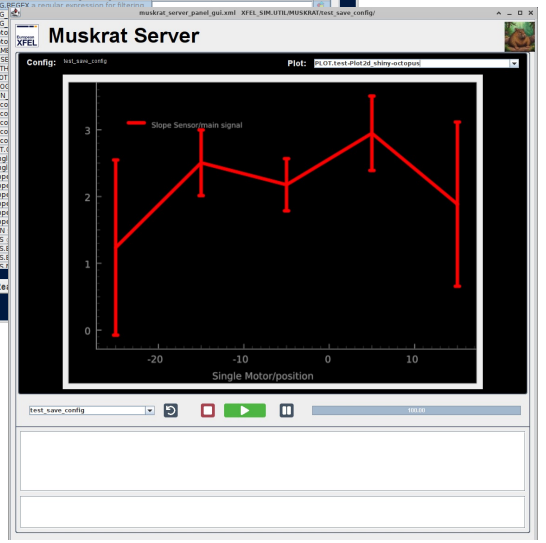
Software Developments in Service of Automation

Example: Muskrat Scan Tool

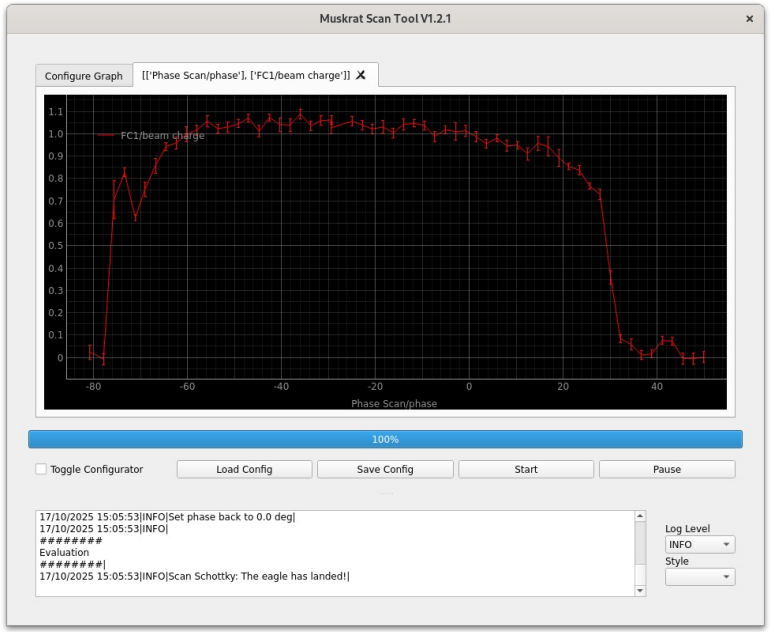
- **Powerful parameter scan tool developed in Python**
  - Create complex presets, which can replace existing single-purpose scan tools
    - ► …using the Python GUI
    - ► …by directly writing a .yaml file
  - Also available as a server via `doocs4py`
  - Workflow analogous to the Badger Server
    - ► GUI and server use the same *engine* and *preset data base*
  - Replacing single purpose tools with Muskrat presets improves maintainability and makes the procedures available to other servers/tools
  - Run scans via GUI, command line, or DOOCS



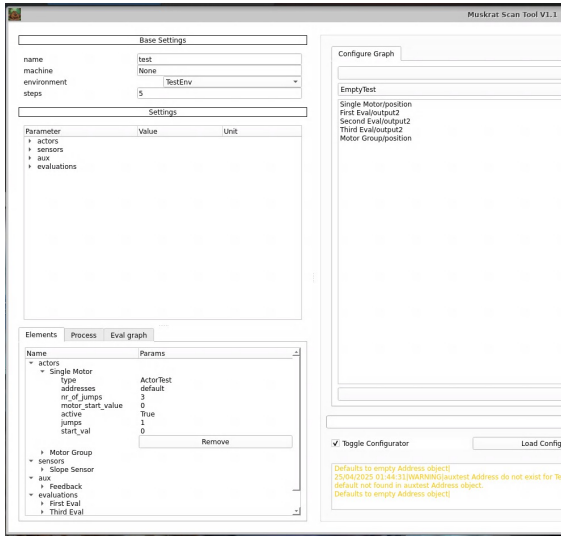
Muskrat doocs4py server



JDDD panel with debug scan



Example gun phase scan performed at ARES using the Python GUI



Scan editor

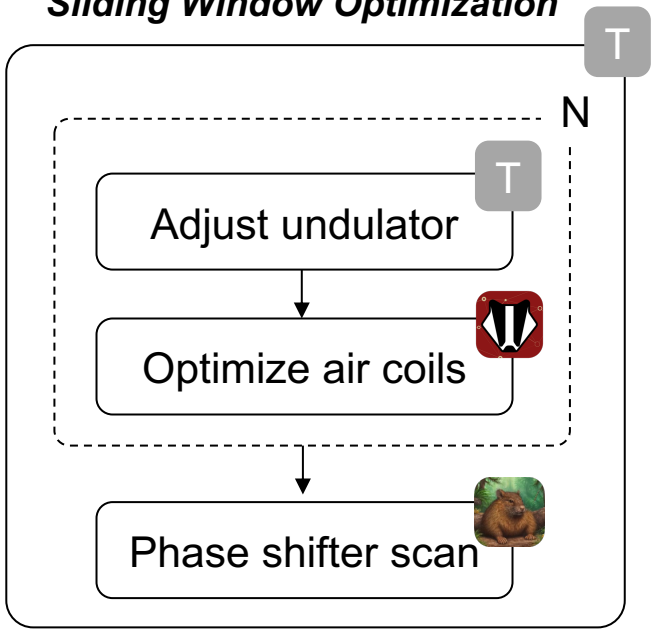
Software Developments in Service of Automation

Example: Combining tools to tackle complex optimization scenarios (*work in progress*)

- **Combination of**
  - Taskomat → Arbitrary control scripts
  - Badger Server → Optimization algorithms (`xopt`)
  - Muskrat Server → Parameter scans

- Enables automation of time-consuming, complex optimization tasks
  - Extremely flexible
- Visualization is key
  - Possible e.g. using server-defined control room panels
- Everything is server-based
  - Laying the groundwork for tool calling agents
  - Toolbox for RL, or LLM-based orchestrators

Automatic Tuning of different Undulator Sections:
Sliding Window Optimization



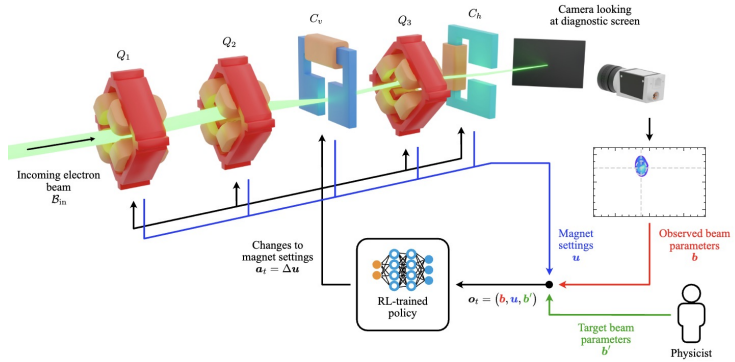
Reinforcement Learning for FEL Tuning (XFEL R&D Project)

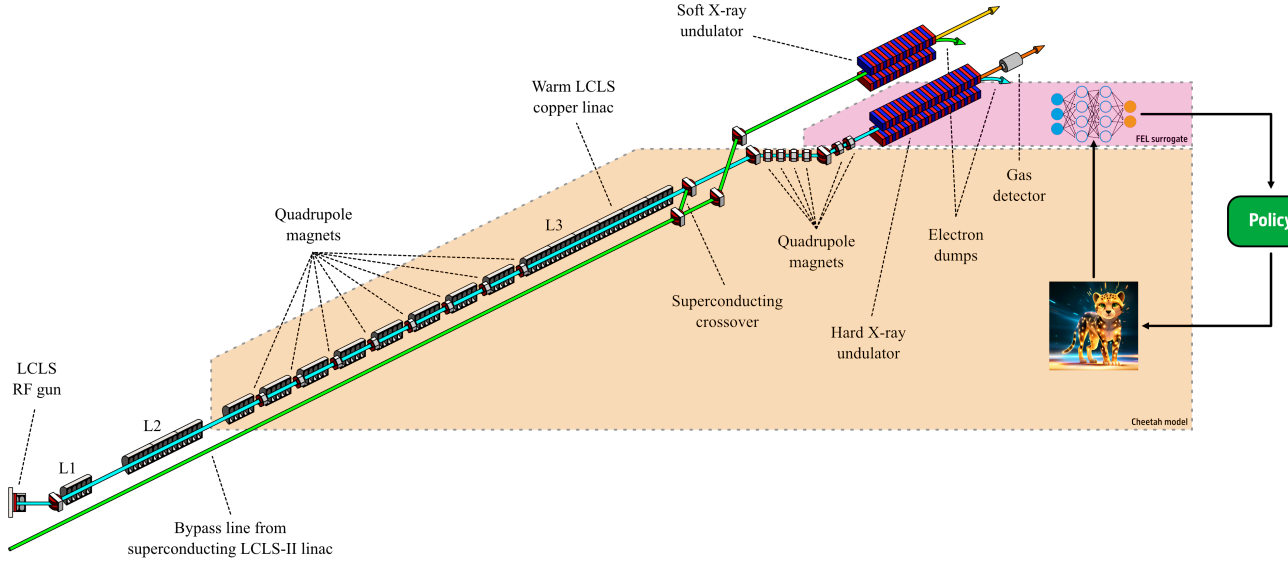
Status

- RL-based tuning methods developed at other facilities
  - ► Transverse beam parameter tuning at ARES
  - ► FEL intensity tuning at LCLS
- Extensive performance investigations
  - ► Outperforms alternative algorithms like Bayesian Optimization (BO) in speed and result
  - ► Achieves comparable working points to expert human operators significantly faster

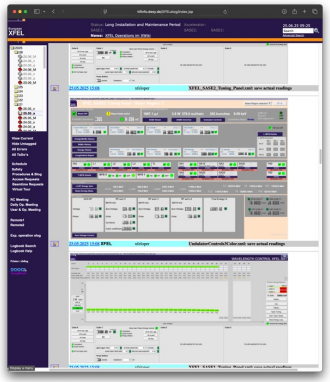
Next steps

- Transfer to EuXFEL
- Hierarchical approach combining tuning methods towards more automated operation



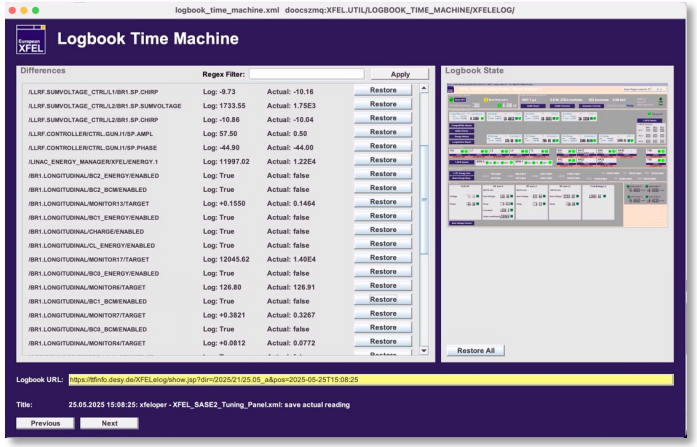


Enhancing Existing Tools

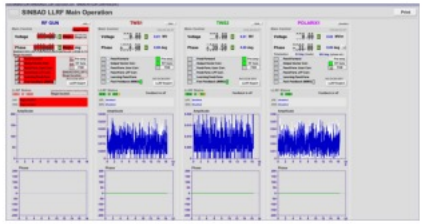


- **jddd** → GUIs for accessing control system servers
  - Server defined panel components
  - Panel metadata for better documentation
    - ► Static data: Author, Description, Purpose, Plot bounding boxes, etc.
    - ► Dynamic data: Widget data, Usage history
- **eLog**
  - Image metadata (automatic for jddd panels)
  - LLM-based documentation suggestions (work in progress)
    - ► Automatic enhancement of metadata with LLM-based plot descriptions

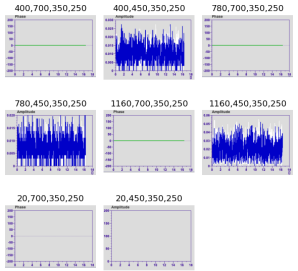
Use logbook metadata to restore panel settings
Panel layout generated by the server

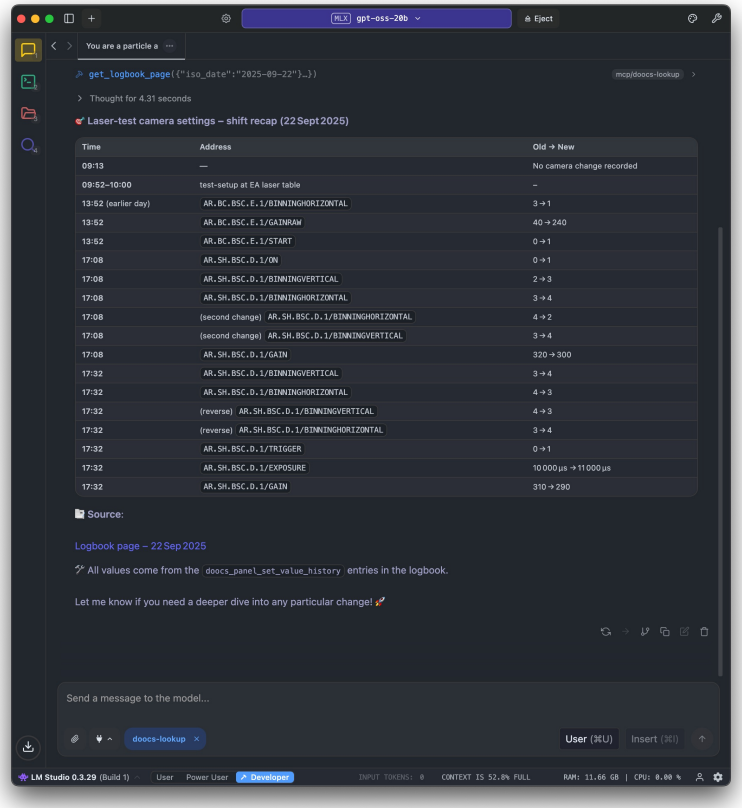


Easily extract plots from panel screenshots for further analysis



```
entry = LogbookEntry.from_url(url)
entry.load_image()
plots = entry.get_jddd_panel_plot_images()
```

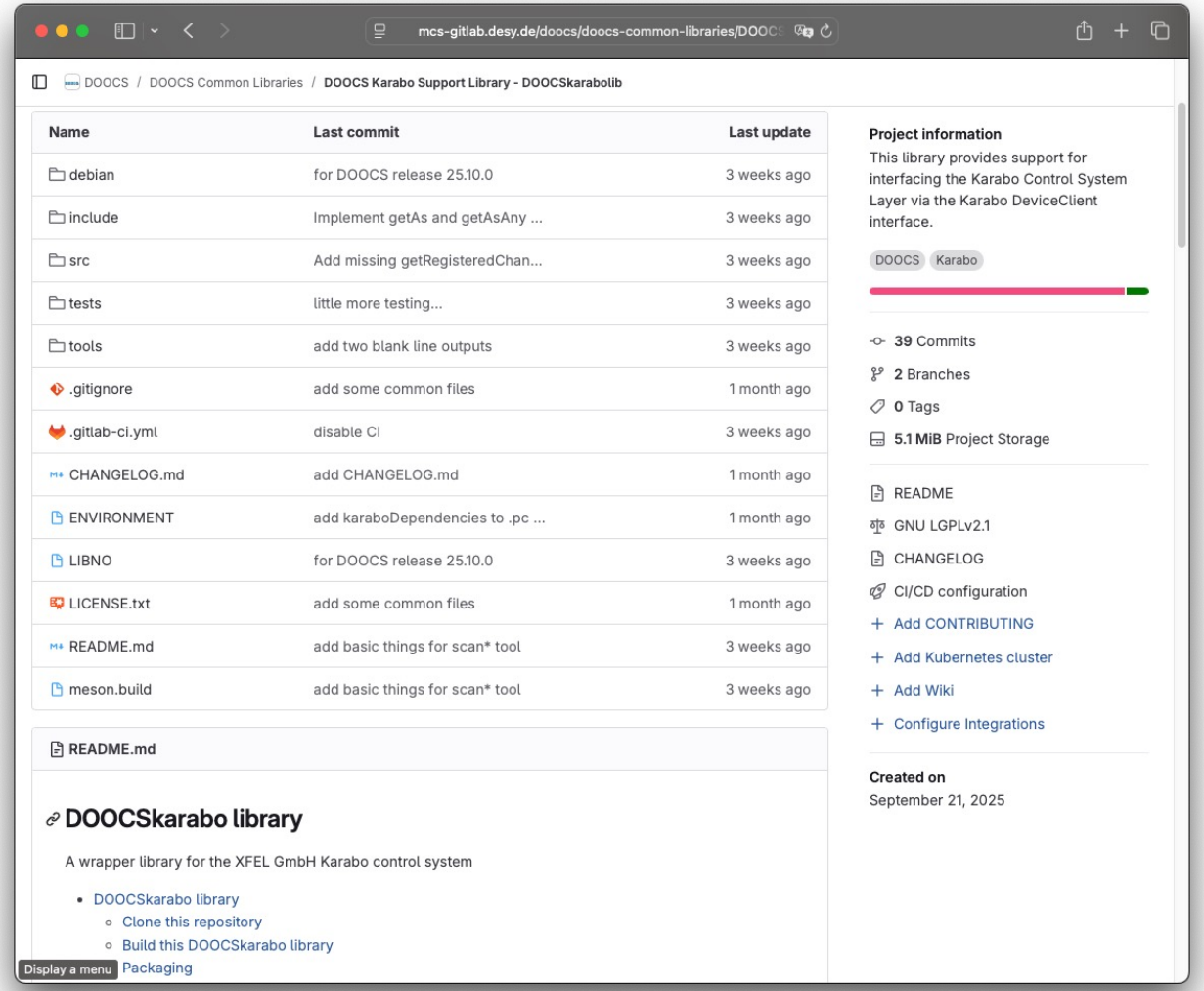




LLM-based logbook search benefits from metadata

Enhancing Existing Tools

- **New DOOCSKarabolib**
  - Ready for KARABO 3
  - Based on experience from the existing KARABO bridges and the HIREX Data ML server
  - Simplifies interfacing with KARABO within DOOCS servers
  - Used by the new versions of the KARABO bridges, as well as the HIREX Data and Control ML servers



Automation Plans

- **e-beam based undulator alignment → Sliding window optimization**
  - Time consuming procedure during initial setup of a working point
  - Has proven to be useful if performed correctly
  - Combination of Taskomat (undulator setup), Badger (aircoil optimization), Muskrat (phase shifters)
- **Photon beam pointing correction using quad movers and K-mono**
  - Cell-by-cell procedure, which is tedious by hand
  - Planned automation builds on manual safe setup of K-mono and undulator line
  - Expert procedure
- **Enhanced phase shifter scan** using two devices at the same time
- **On-crest phase setup**

Automation (and other) Plans

- **Muskrat presets**
  - Implementation of Tuesday e-beam characterization procedures → Run via server
    - ► Example: Fast QE map
  - Fast gain curve (was actually the starting point for Muskrat)

- **Streamlining of existing tools**
  - Example: Move the adaptive orbit feedback from the Ocelot GUI to the DOOCS feedback server

- **In collaboration with the photon side: HIREX setup automation**
  - HIREX Control ML DOOCS server
    - ► Calls safe KARABO MDLs

Agentic AI – The Future?

- **Current AI work mostly focused on knowledge retrieval (eLog, Xwiki, Procedures, etc.)**
  - Implementation of knowledge retrieval tools as Model Context Protocol (MCP) servers / tools
  - Benefits a lot from e.g. logbook metadata
- **Automation tools shown before organize setup / optimization / beam characterization tasks into high level blocks, which lend themselves to be called (or suggested) by agentic systems**
  - Reduce complexity and ensure safety!
- **Close collaboration with Berkeley Lab** → Osprey Framework **https://github.com/als-apg/osprey**
  - Framework for agentic AI systems
  - Plan-first orchestration with human oversight (in contrast to simple ReAct agents studied before)
  - Feasibility to be evaluated – First tests first using our knowledge retrieval MCP servers (harmless)

F. Mayet, ***Building an intelligent accelerator operations assistant using advanced prompt engineering techniques and a high-level control system toolkit***, LIPS Symposium, Feb. 2024
F. Mayet, ***GAIA: A general ai assistant for intelligent accelerator operations***, arXiv:2405.01359, May 2024.
A. Sulc et al., **Towards Agentic AI on Particle Accelerators,** arXiv:2409.06336, September 2024
A. Sulc, et al., ***Towards Unlocking Insights from Logbooks Using AI***., Proc. IPAC'24, Nashville, USA
T. Hellert et al., **Osprey: A scalable framework for the orchestration of agentic systems**, arXiv:2508.15066v2, 2025

Summary

- We continue to tackle repetitive tasks with our mature automation tools
- Current software developments are in service of automation, ML/AI deployment, and maintainability
- The combination of Taskomat, Badger Server, and Muskrat enables us to combine individual stand-alone solutions into larger units, which are then available to be used by the operators or any other tools/servers

- Work on software infrastructure during LIMP
- Implementation and test of automation scenarios as much as possible when the machine is back

- Feasibility studies on agentic AI on-going

Thank you for your attention!