Karabo Overview

Dr. Gero Flucke
for the
Control and Analysis Software group

Satellite Workshop on
Karabo Control and Data Analysis at European XFEL
January 24th, 2019

Outline

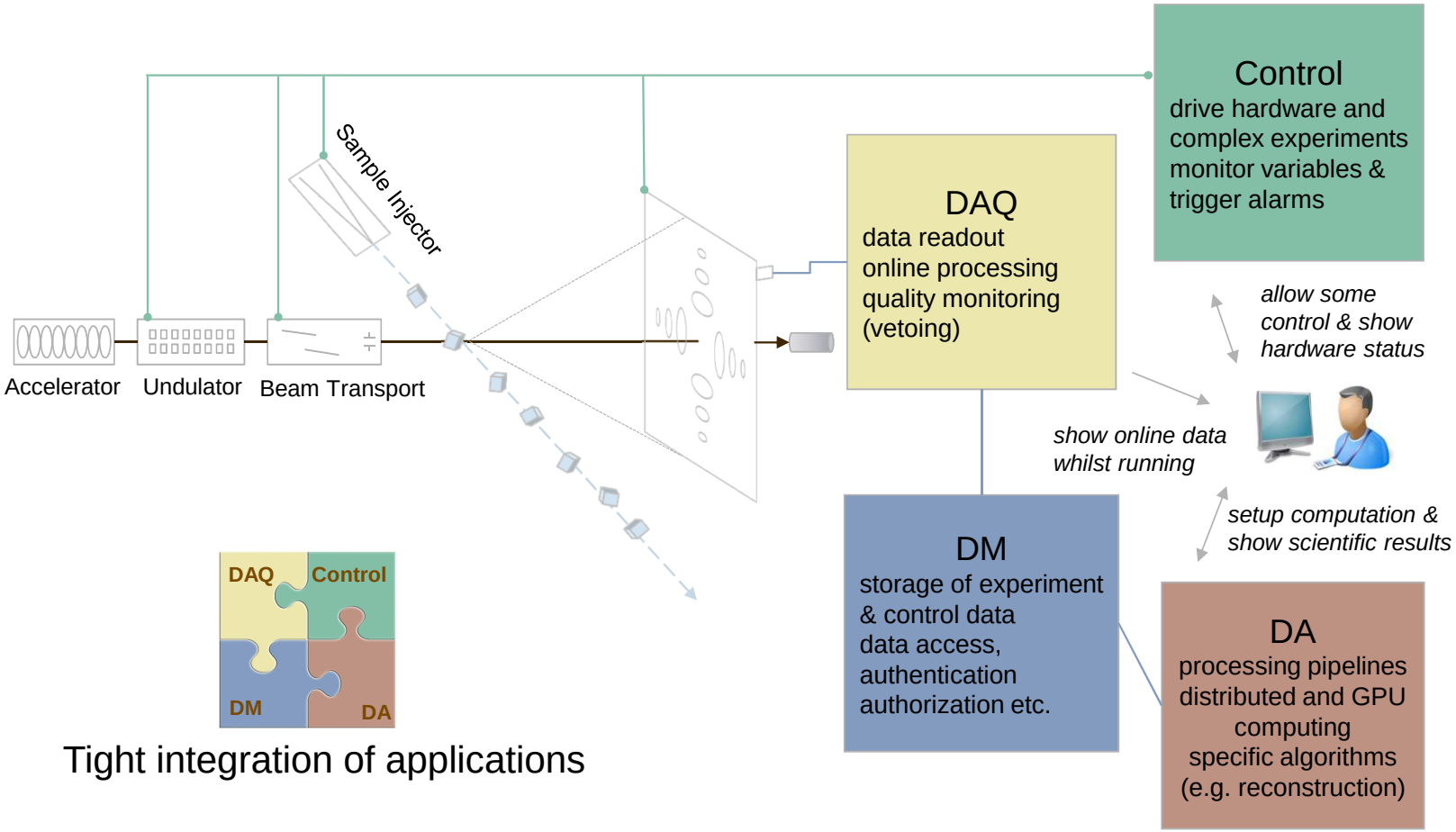
- Introduction to the Karabo Framework
- Graphical User Interface
- Data Acquisition and Scanning
- Data Analysis Capabilities

Introduction

- Karabo is custom made European XFEL's
  - **Control** System: controls a wide-range of light source instruments
  - **Data acquisition** system: GB/s scientific data collection
  - **Data processing** framework: Native support of distributed process chaining using pipelines.
- Under active development
- Drives European XFEL commissioning and user experiments

Unified approach required for:

- Instrument Control
- Data Acquisition (DAQ)
- Data Management (DM)
- Data Analysis (DA)

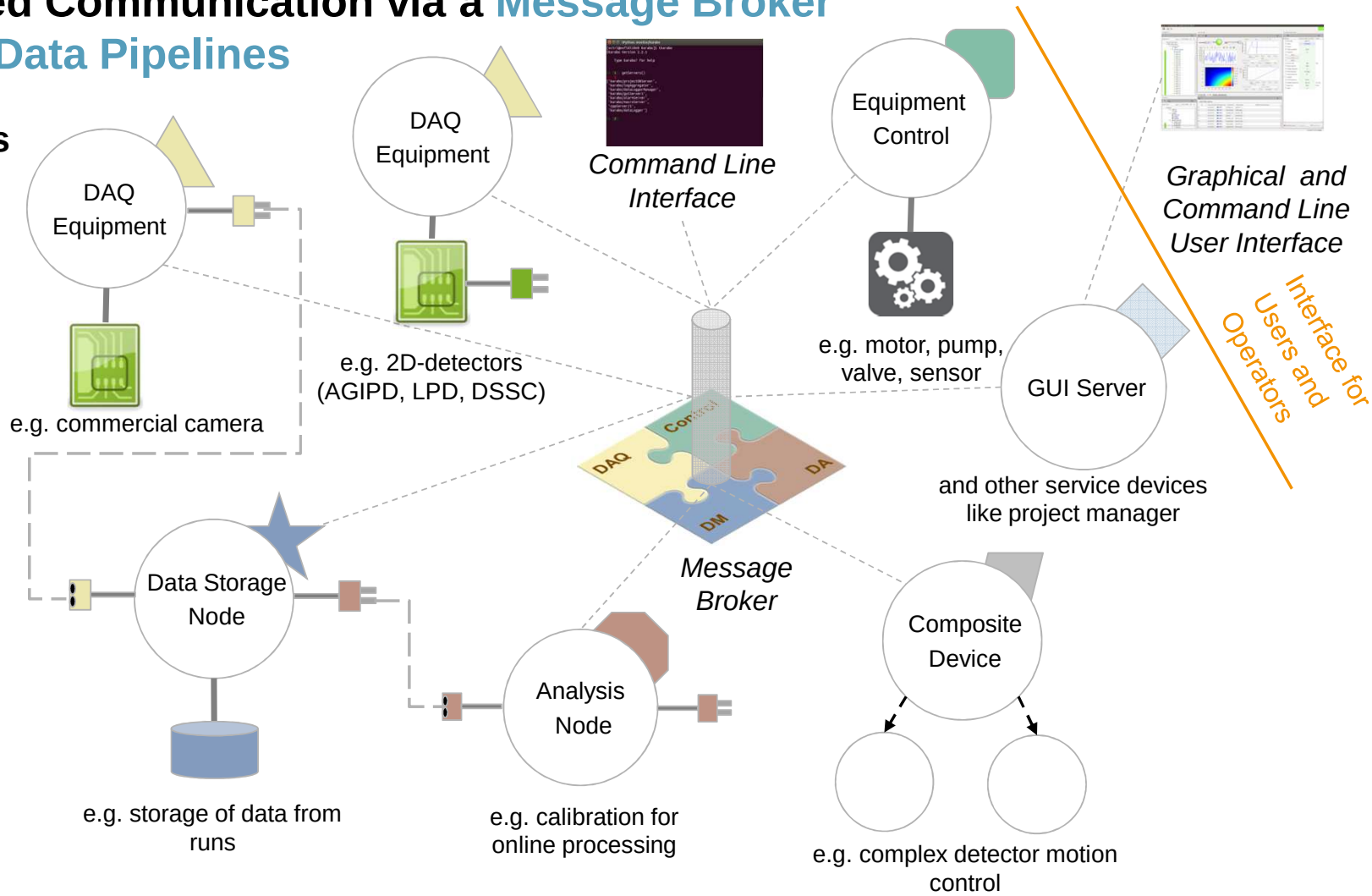


Tight integration of applications

Karabo: Device Based Communication via a Message Broker and TCP/IP Data Pipelines

Individual Karabo Devices

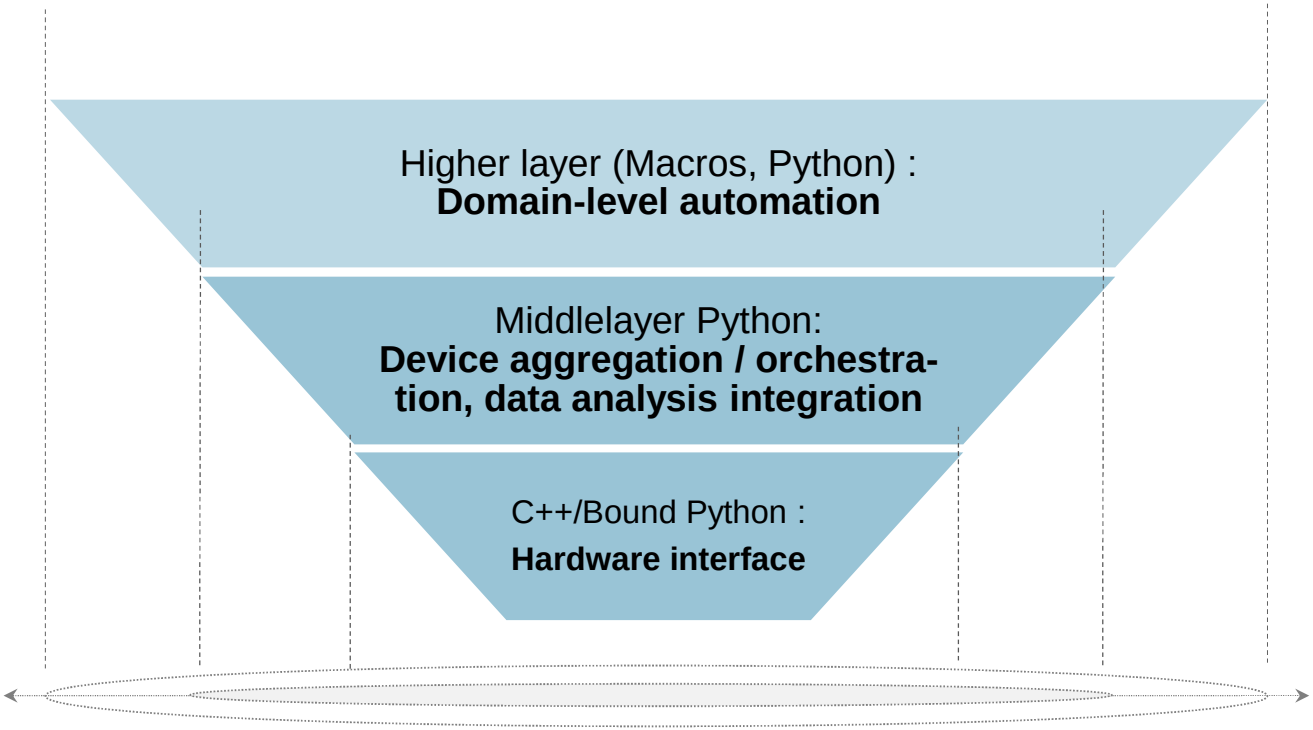
- Equipment control, e.g. motors, valves,…
- Detectors like
  - big 2D detectors,
  - cameras
- Data storage
- System services
- Online data analysis, e.g. calibration



Four Application Programming Interfaces

…and whom they address:

- Macros:
  instrument scientists,
  possibly later users
- Middlelayer:
  Control experts with feedback
  from instruments,
  maybe instrument scientists
- Hardware interface:
  Control experts with hardware
  knowledge

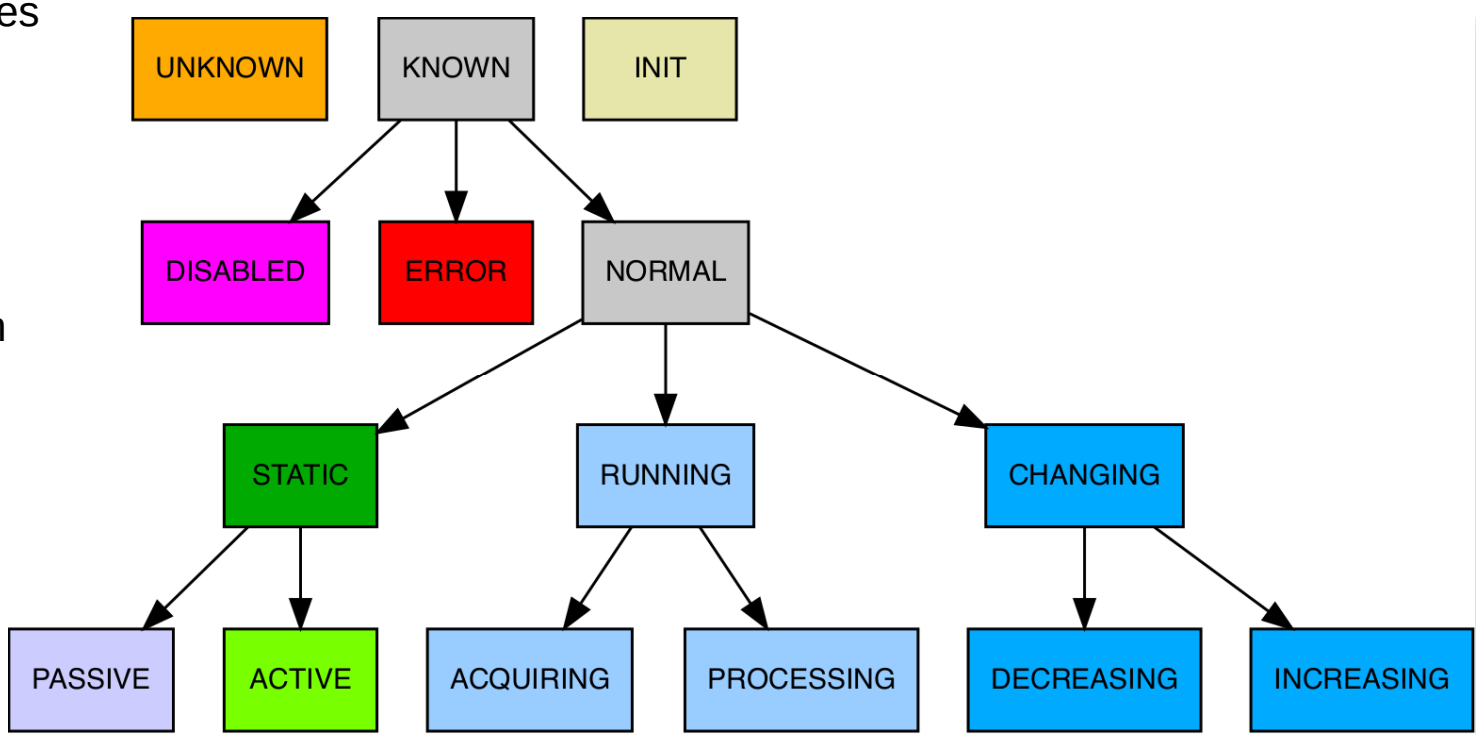


Data Types Supported by Karabo Communication

- 8-, 16-, 32-, 64-bit signed and unsigned integer values
  - 8-bit raw data, booleans
- 32- and 64-bit floating point numbers
  - complex numbers thereof
- strings
- vectors (lists) of all the above
- `NDArray` (for pipelines): container of multi-dimensional “big data”, optimised serialisation
- `ImageData`: container for
  - `NDArray`
  - meta-data like info about bits-per-pixel, roi offsets, binning, encoding type,...
- Karabo Hash as nested key / value container
  - key: string – nested access through concatenation separated by dot: `aHash.get("key1.k2.k3")`
  - value: can be any of the above

Unified Device States

- Predefined list of device states
- Inheritance system
  - E.g. ERROR is more concrete than KNOWN
- Unified colour representation in the GUI
  - Colour inherited if not explicit

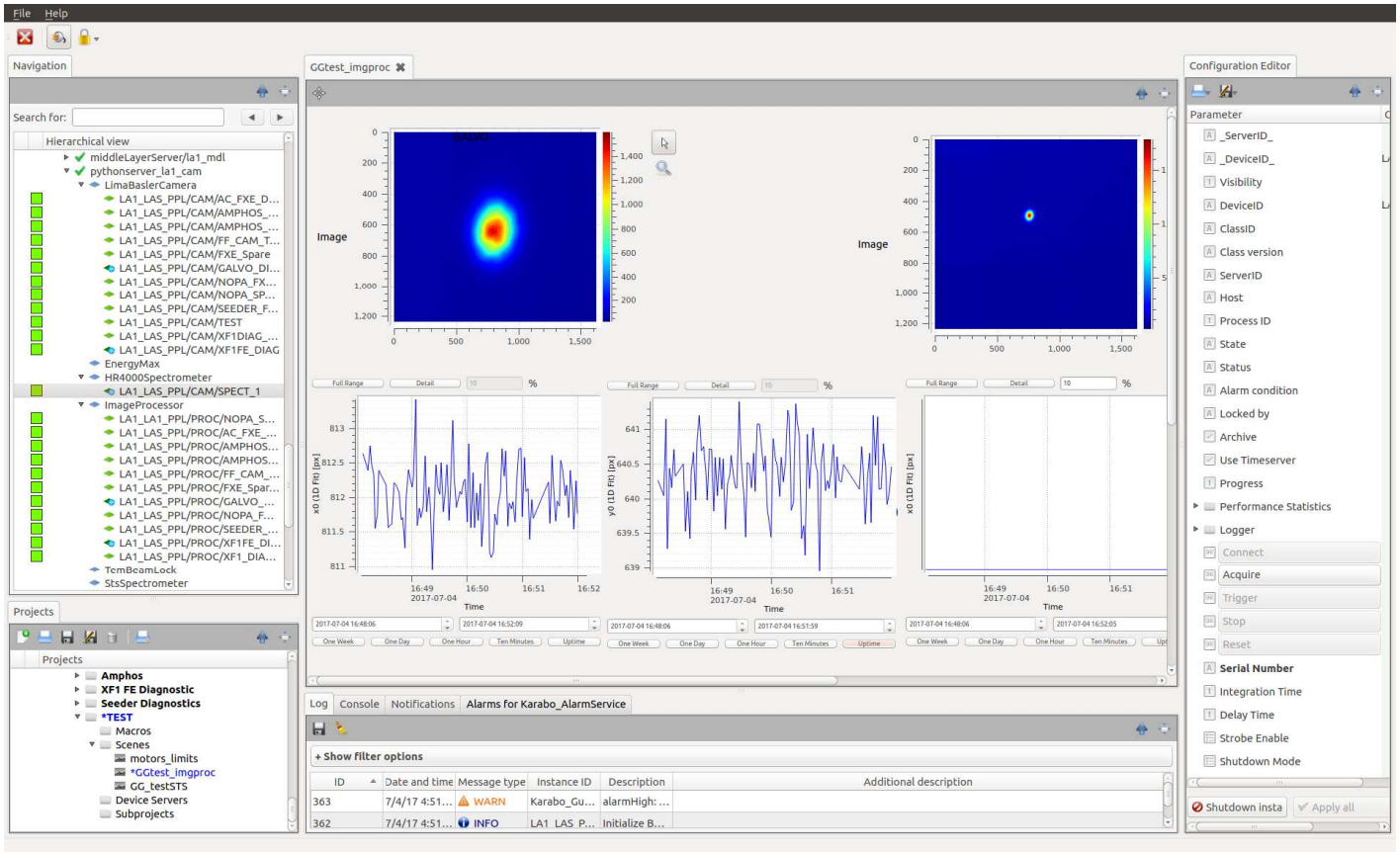


About 60 more states inheriting from those of the last row.

Data Logging and History Reading for Control

- Provides history retrieval for
  - GUI trendlines and configurations from past
  - CLI interface
- Not a data acquisition, no data policy, no pipeline nor big 2D detector data:
  - No data for users…
- Karabo logger service devices subscribe to parameter changes
- Store data in human readable text files per device (and index files):
  20190115T013900.706424Z|1547516340.706424|0|**perfStats.maxProcessingLatency**|UINT32|**35**||VALID
  - Data older than about a month compressed and put on archive
- Karabo reader service reads back using index files
- Plans for refurbishment:
  - Proper data base to replace storage in text files
  - Karabo-less interface

Graphical User Interface: Integrated cockpit



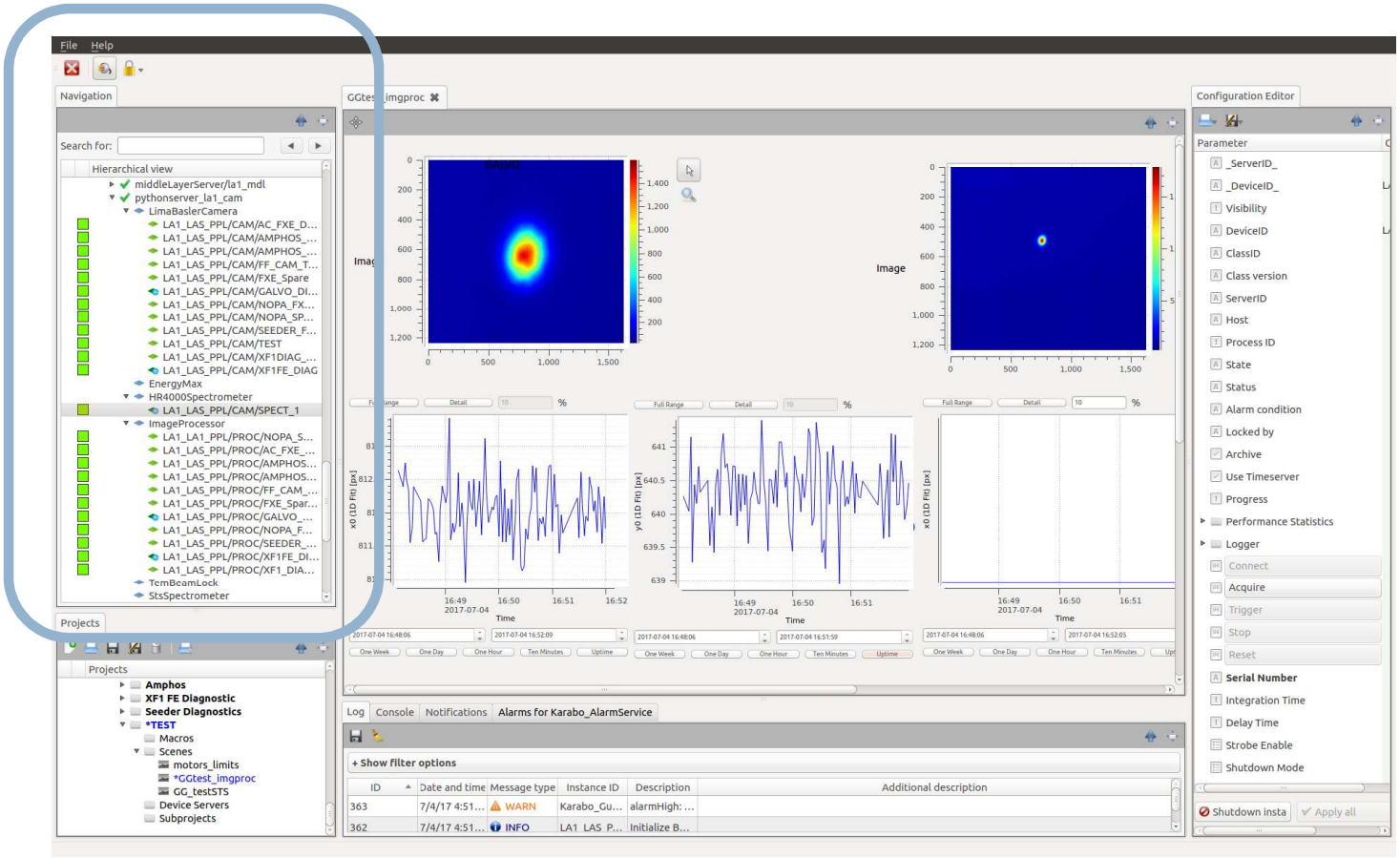
Unified tool:

- For experts:
  - Full system view.
  - Detailed access to configurations.
  - Integrated command line.
- For everybody:
  - Customizable “scenes” (no coding required).
  - Rich set of widgets.
  - Coherent handling, e.g. limits of settings.
  - Visualisation of
    - ► error state,
    - ► properties in alarm condition.
  - …

Graphical User Interface

Navigation panel:

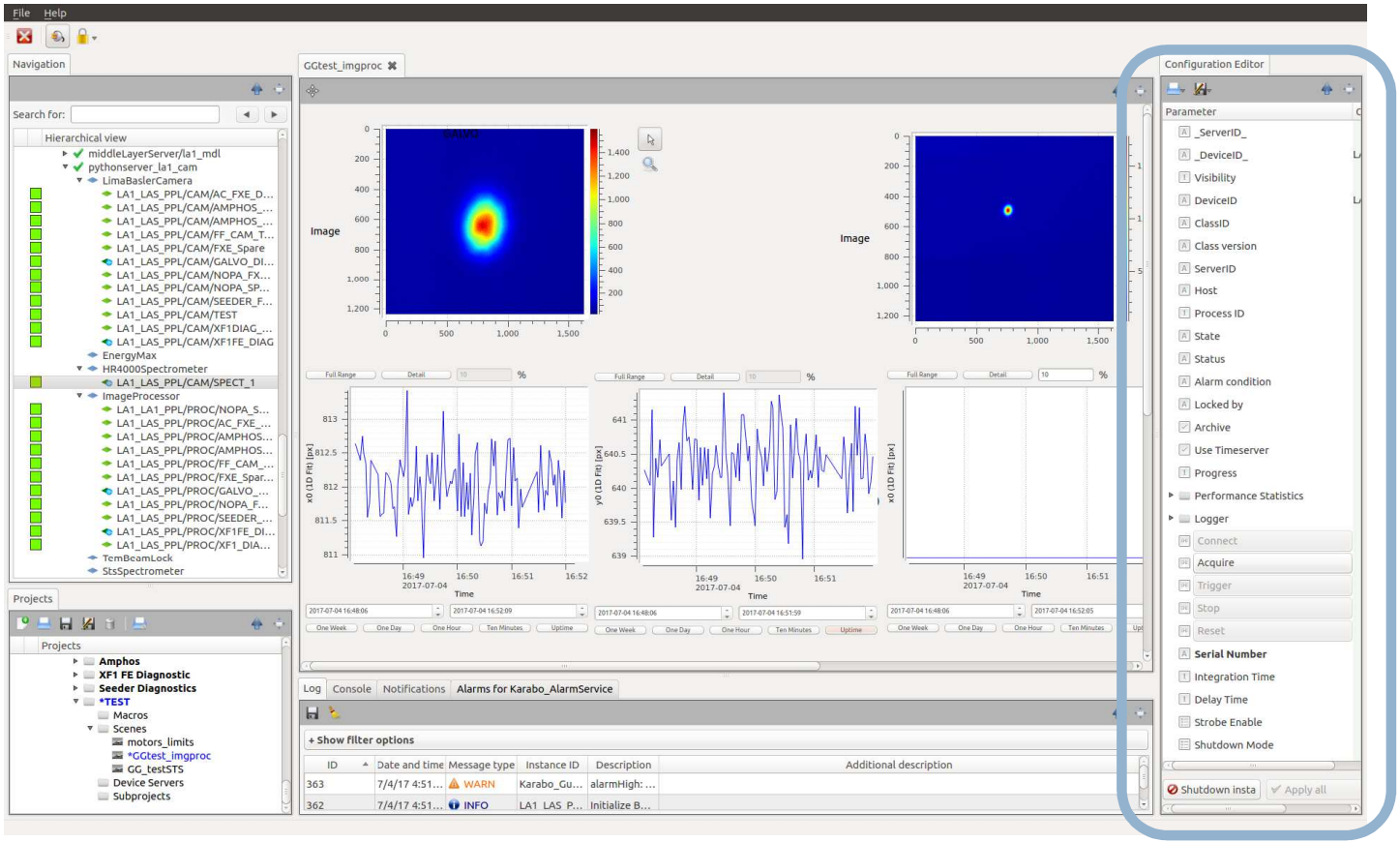
- Complete Karabo system topology
- Searchable
- Colour coded device state indicator (error vs. healthy)
- Alarm levels icons if reached:



Graphical User Interface

Configuration panel

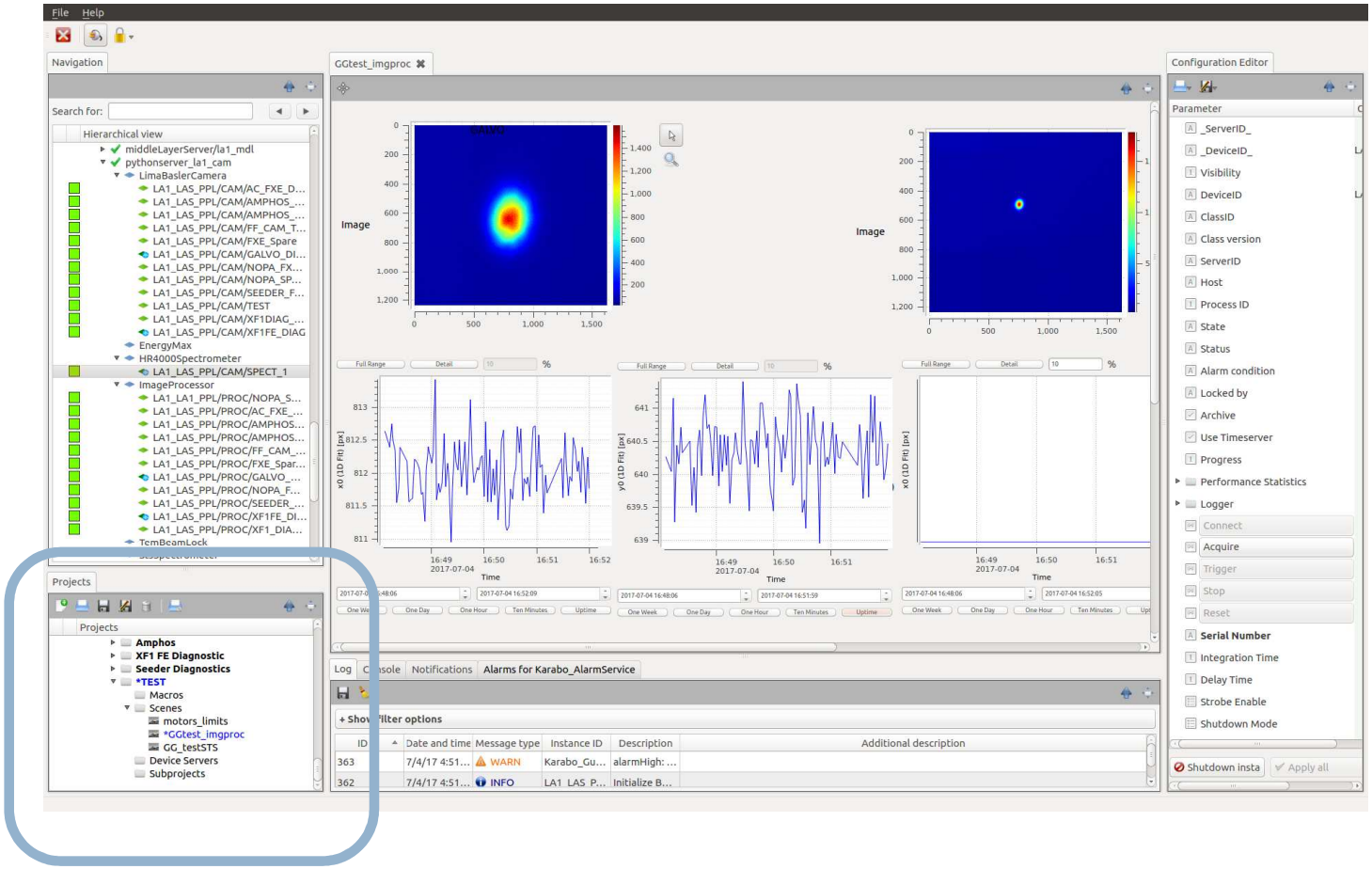
- Edit online and offline device configuration
- Send commands to devices
- State field colouring
- Coloured background for properties in alarm



Graphical User Interface

Project panel

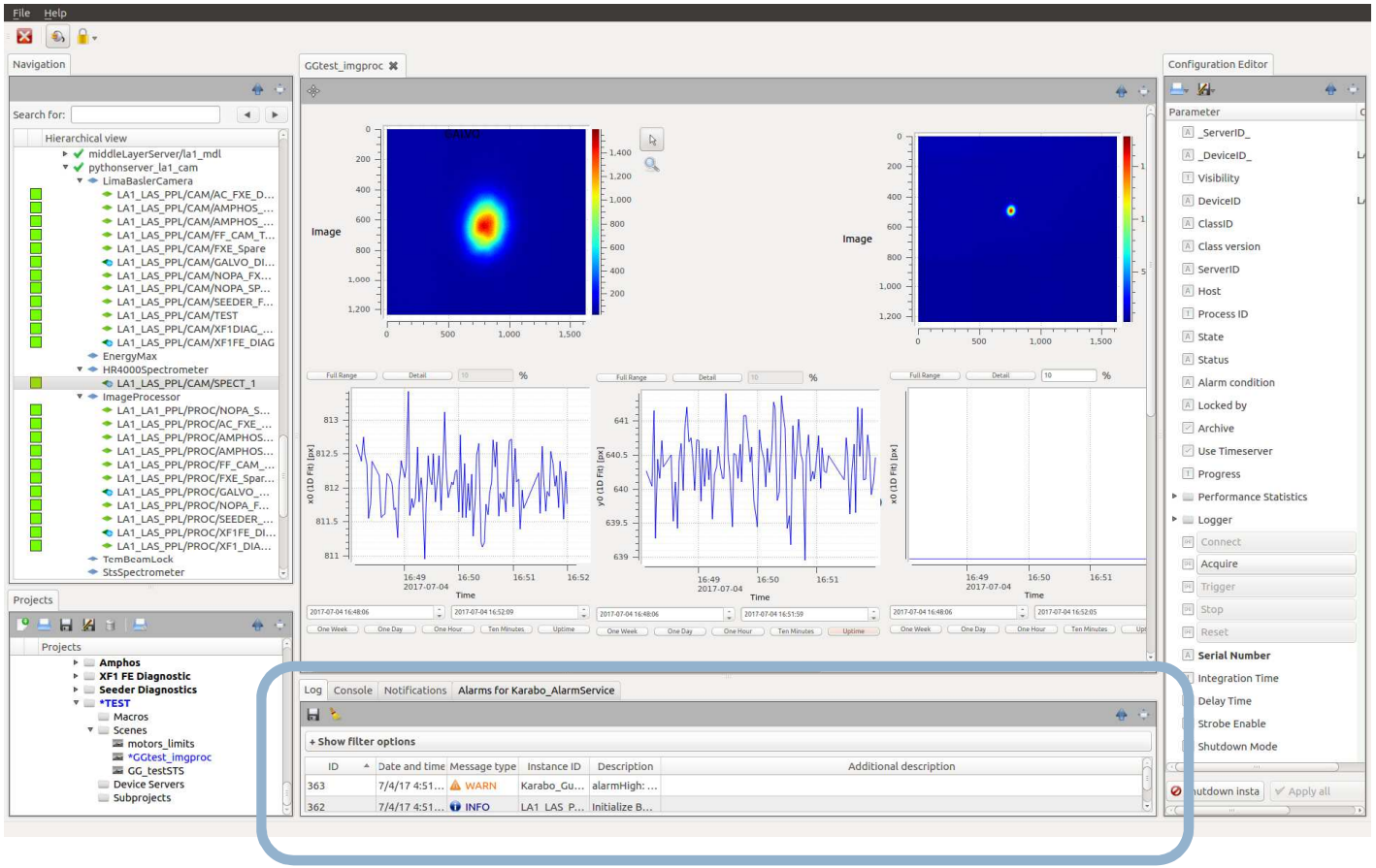
- Project data:
  - device configurations,
  - scenes (→),
  - macros,
  - sub-projects.
- Projects stored in central data base as XML files.
  - local storage option



Graphical User Interface

Panels for

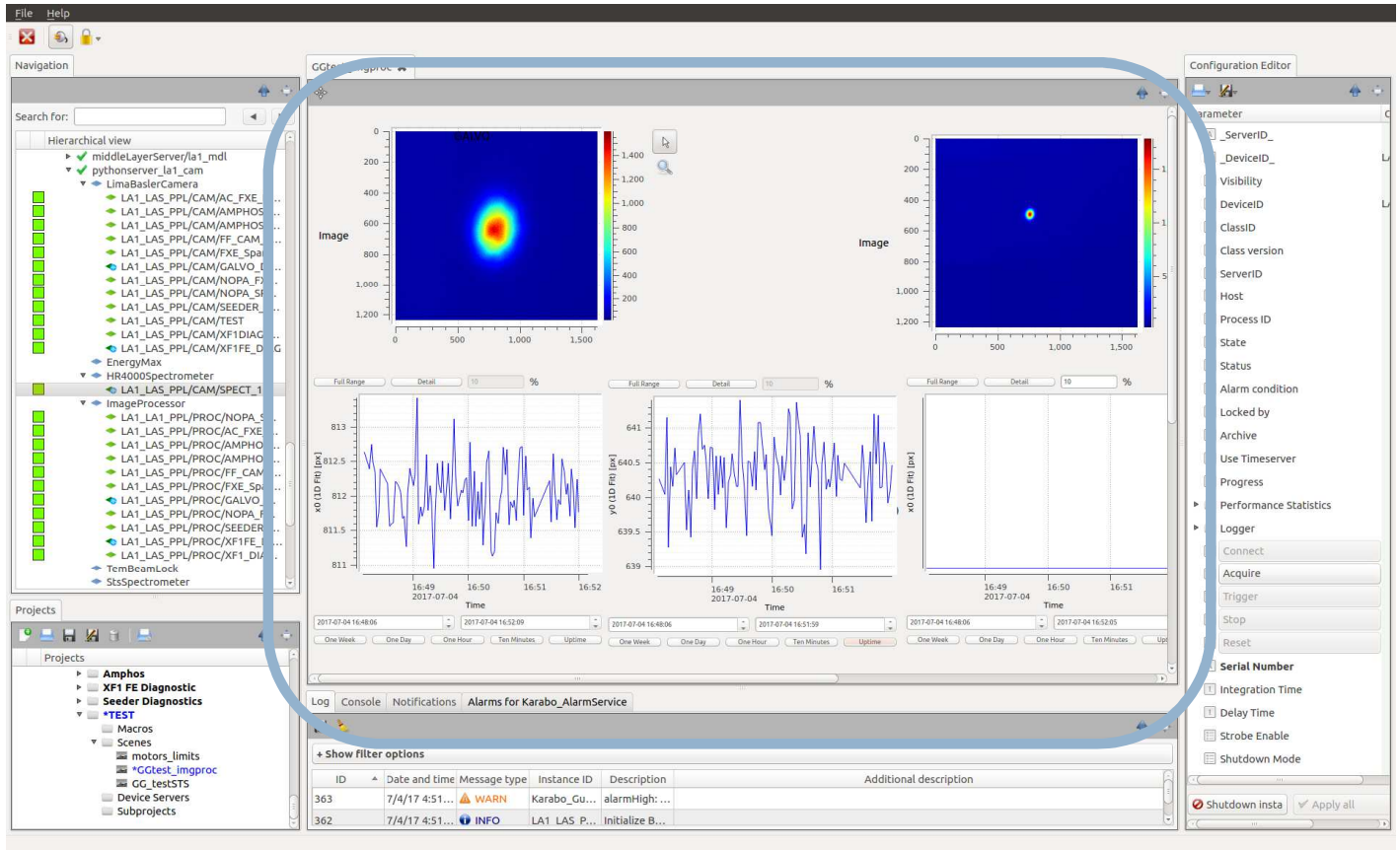
- Message logging
- Alarm handling
- Integrated command line



Scenes

Graphical User Interface

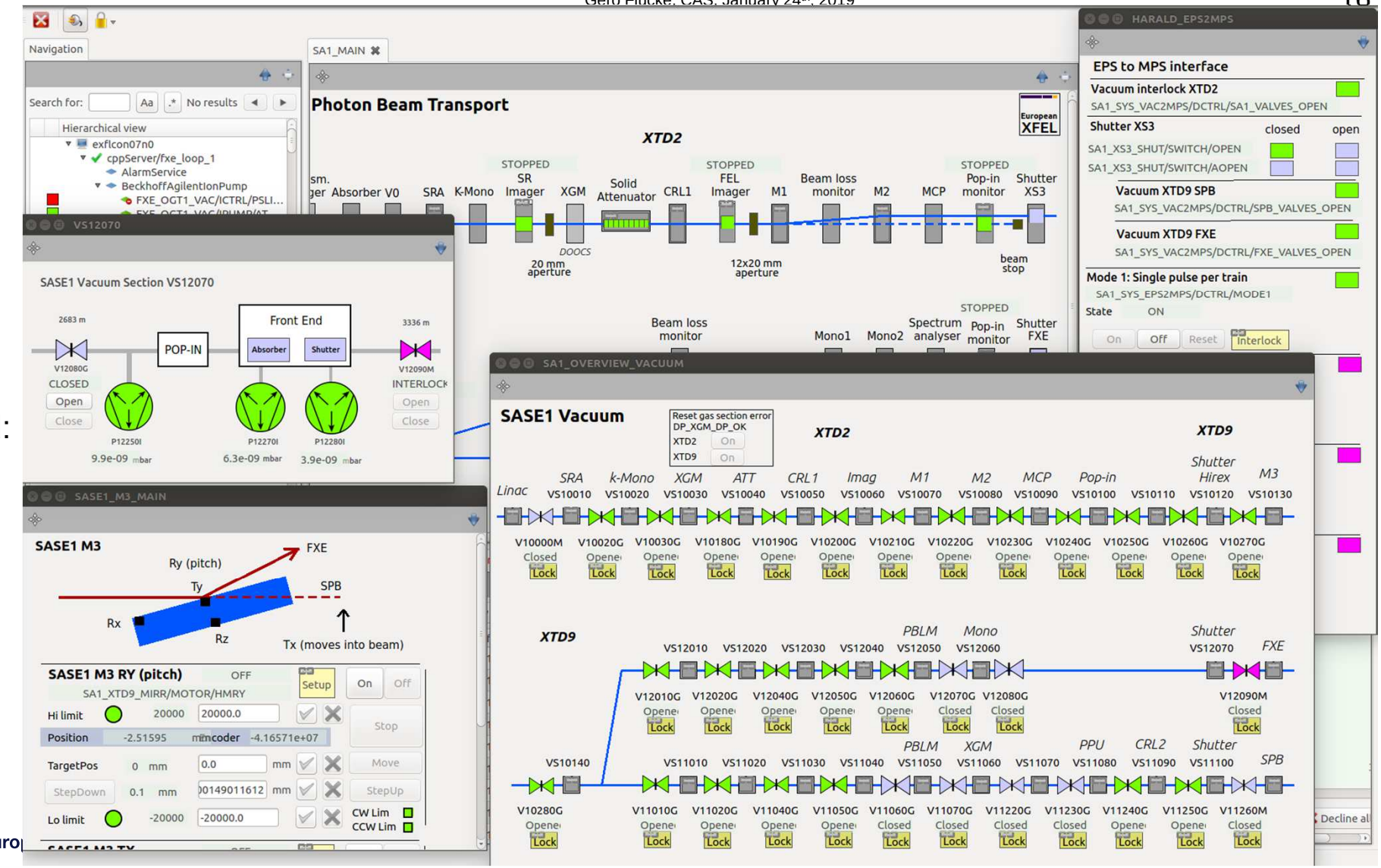
- Easily editable
  - using drag and drop,
  - include svg files,
  - link to other scenes.
- Rich widget set
  - various image views,
  - trend and spark lines,
  - commands,
  - device states,
  - standardized icons,
  - (some details in backup)
- Can un-dock from main window (as all panels)
- Now can run standalone (“cinema” mode)



Scene Example

Beamline Control: SASE1

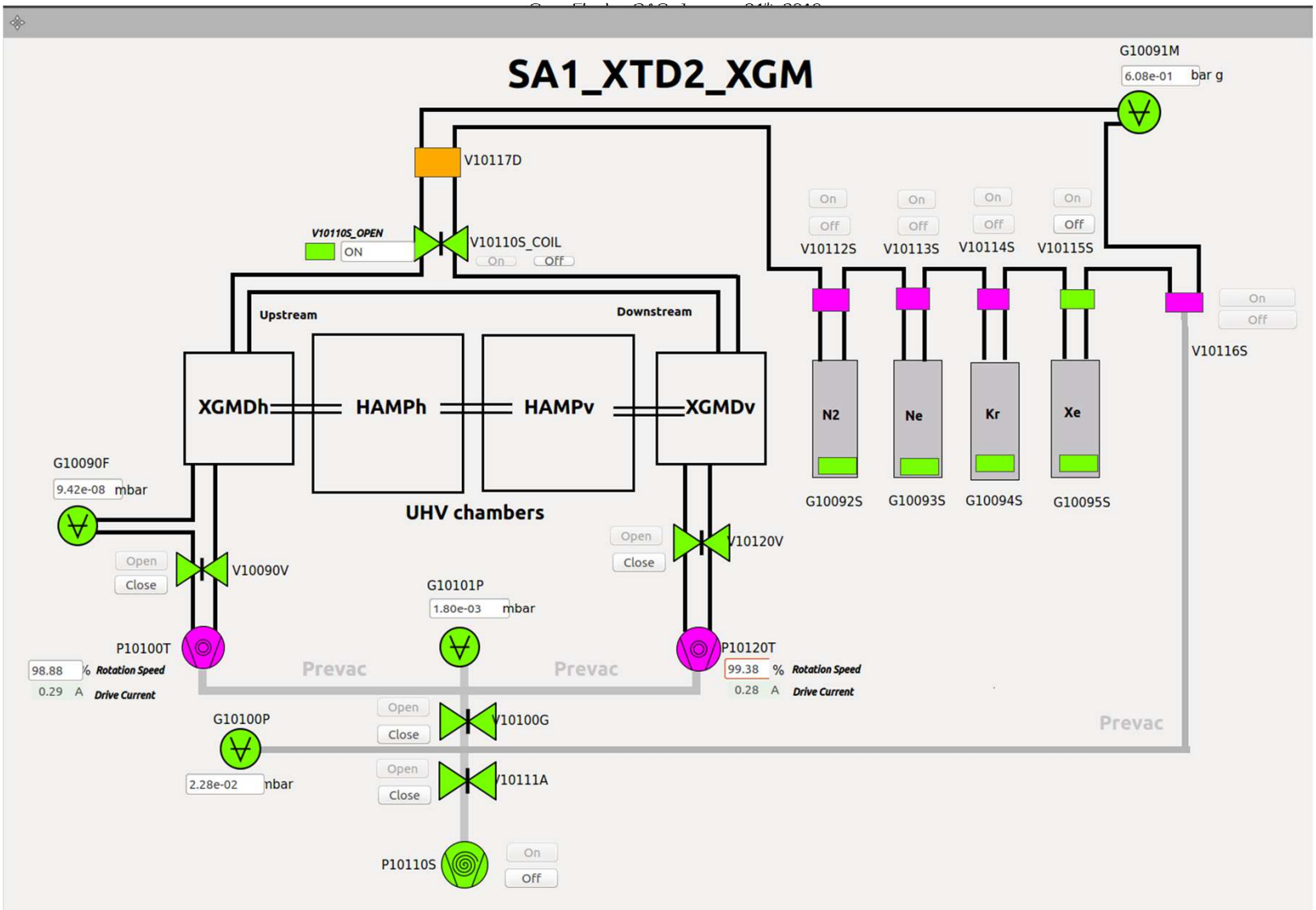
- Icons, e.g.
  - ion pumps,
  - valves
- State coloured:
  - open,
  - passive/ closed,
  - interlock
- Scene links



Scene Example

Diagnostic:
XGM monitoring

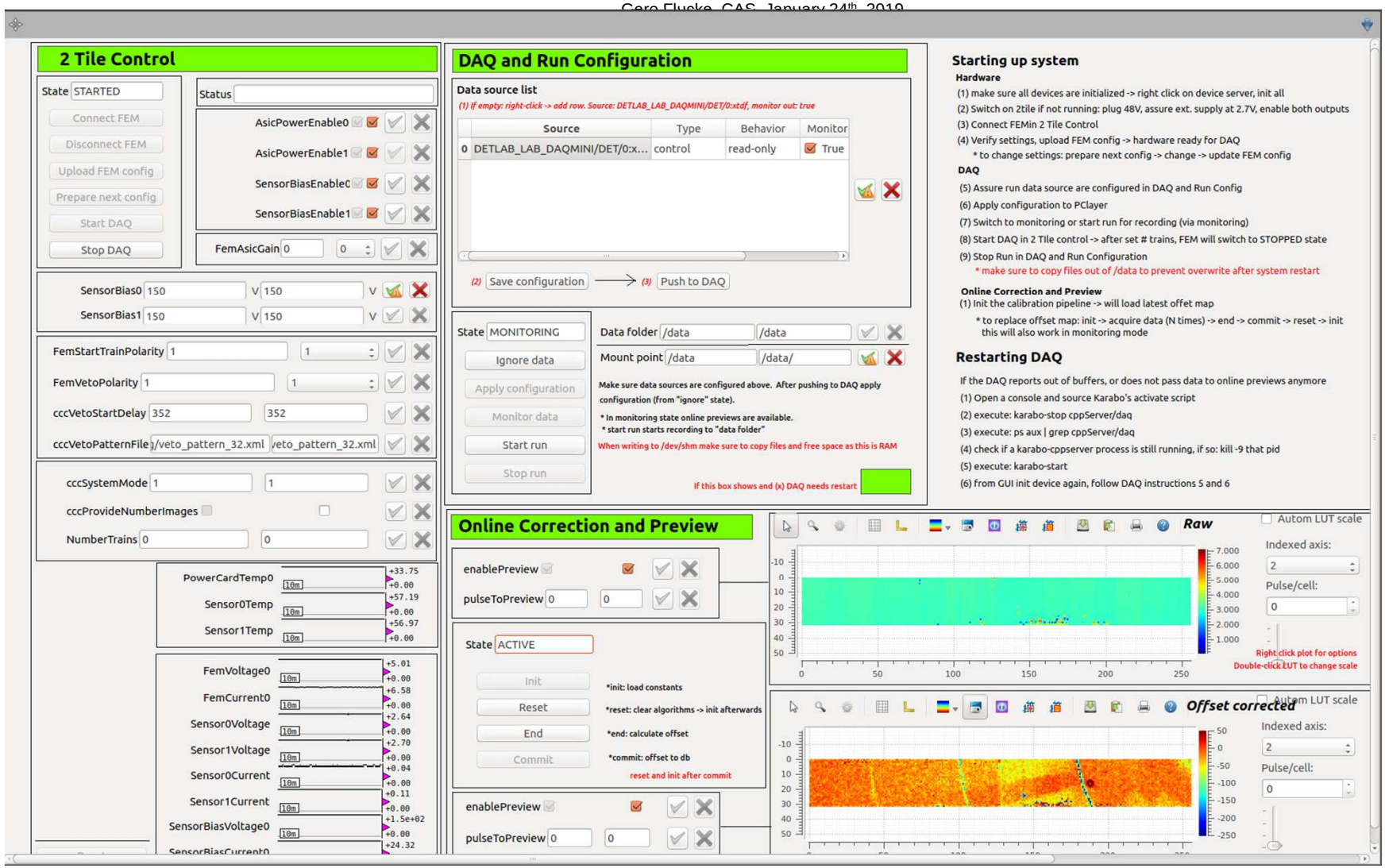
- More icons, e.g.
  - gauges,
  - turbo pumps,
  - scroll pumps
- More state colours:
  - unknown (i.e. no h/w info),
  - disabled / interlocked



Scene Example

Detector control: LPD Tile Control

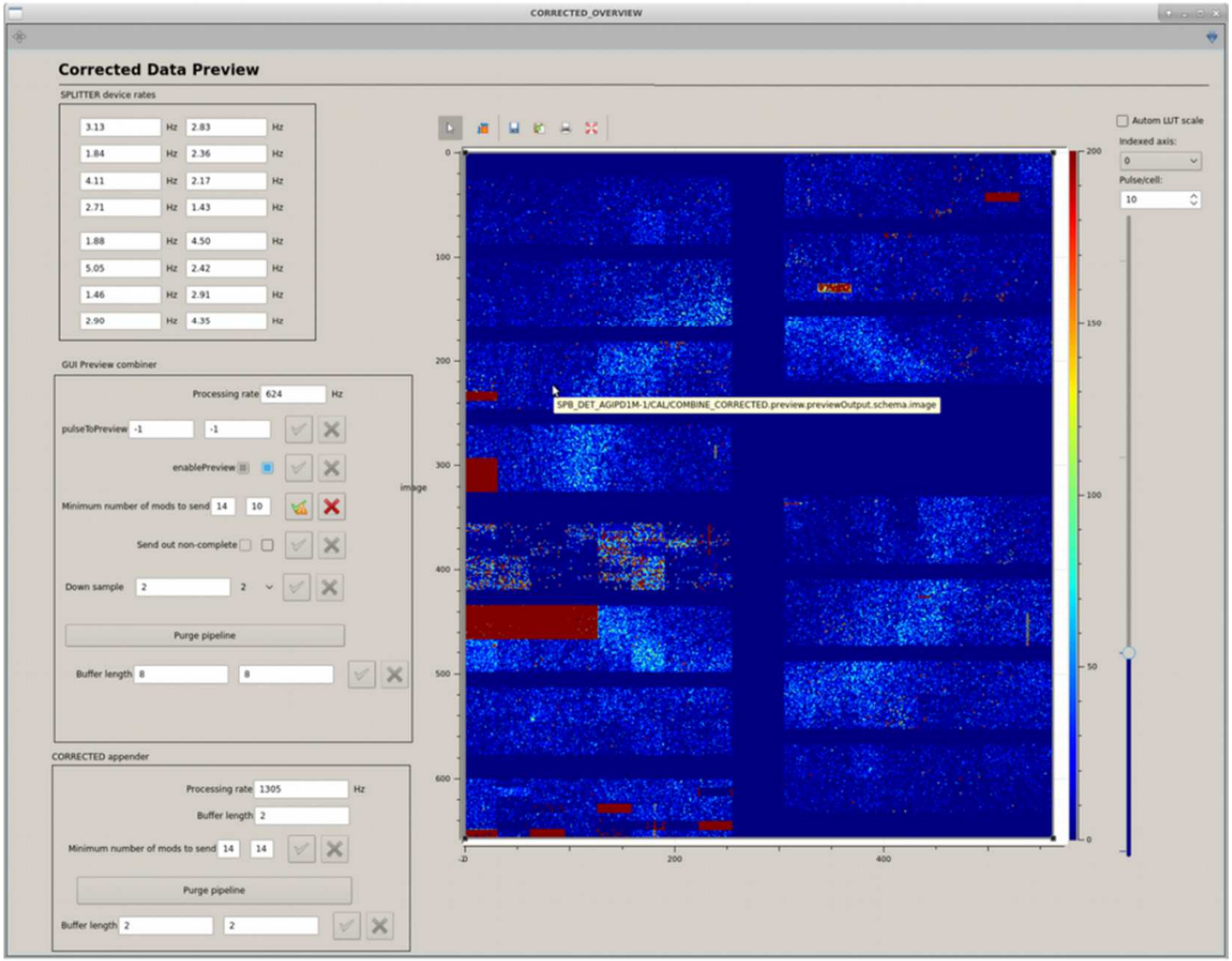
- View result of online calibration



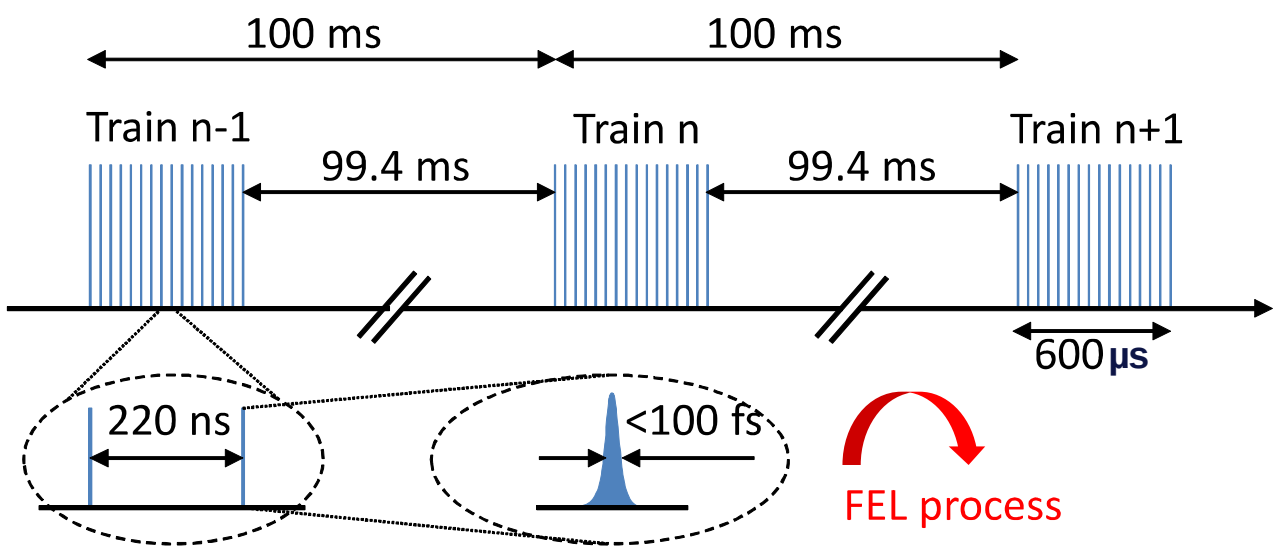
Scene Example

AGIPD Data Preview

- View of online calibrated data
- All 16 readout streams combined



European XFEL: Photons come in trains with short pulses



Pattern:

- **Trains** with 10 Hz
  - Relevant time unit for data storage
- Up to 2700 **pulses** per train
  - pulse duration < 100 fs
  - 220 ns spacing (4.5 MHz)
  - => train length 600 µs
  - Goal: 600 pulses at end of 2019
- On average:
  up to 27 kHz of pulses
  (up to 6 kHz in 2019)

Karabo Data Acquisition (DAQ) Integration

- Support for different types of data sources:
  - Control data with train resolution: e.g. sensors, motors (Karabo)
    → **slow data**
  - 2D or pulse resolved data: e.g. cameras, digitizers (Karabo pipeline)
    → **fast and/or medium sized data**
  - Big 2D detectors as LPD, AGIPD (XFEL train data format)
    → **big & fast data**

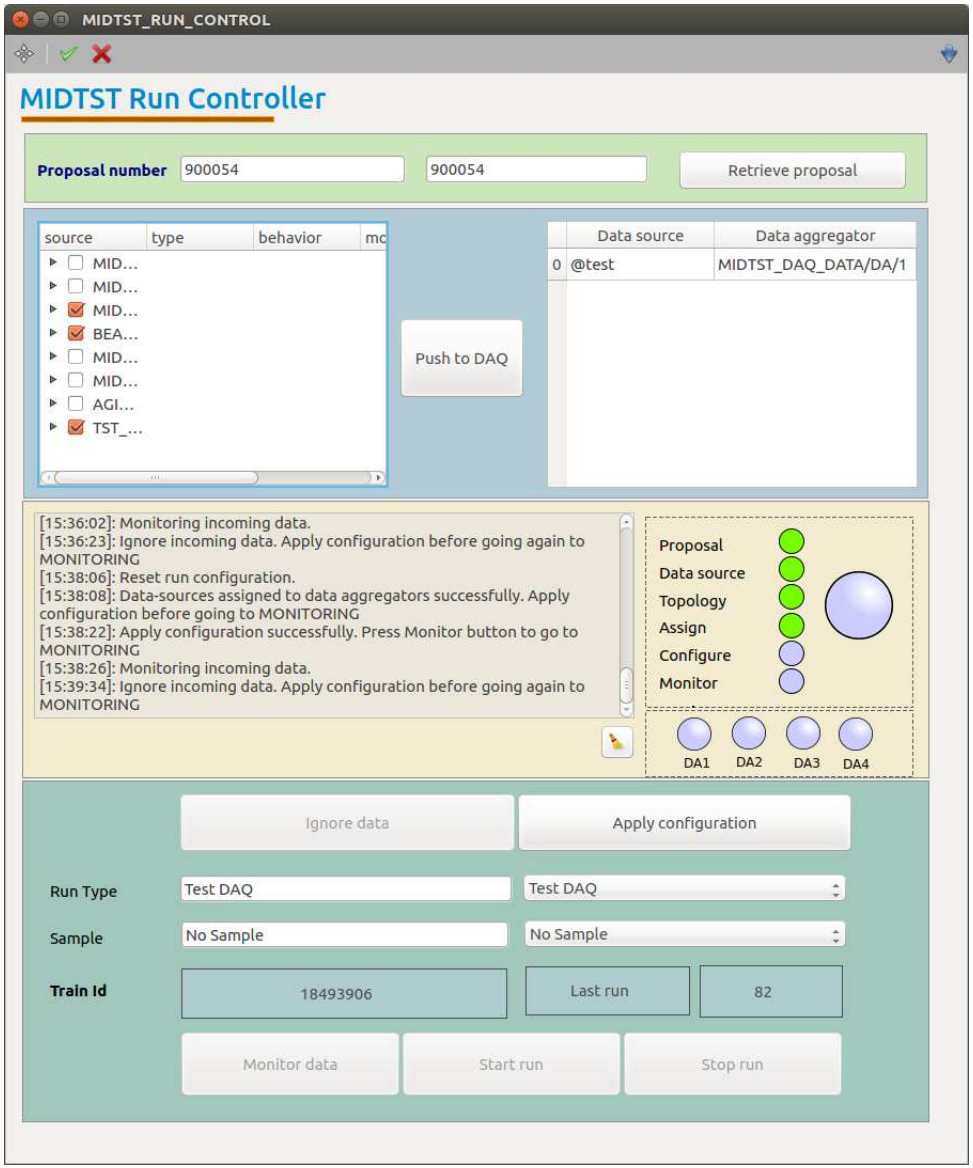
- Data acquisition organised via several Karabo devices

- Data stored in HDF5 files
  - available after run has finished,
  - Karabo and data access tools provide data of all pulses of a train in one go.

- Provide data stream for online display and online analysis,
  - e.g. feeds calibration of big 2D detectors,
  - can be replicated offline to tune online tools.

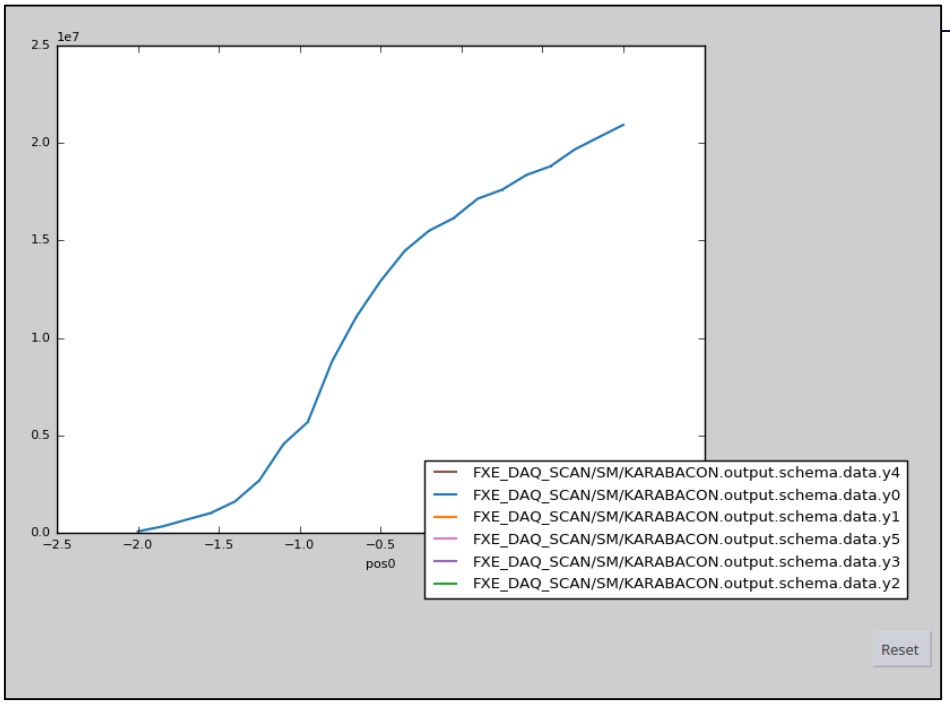
Karabo Data Acquisition (DAQ) Configuration

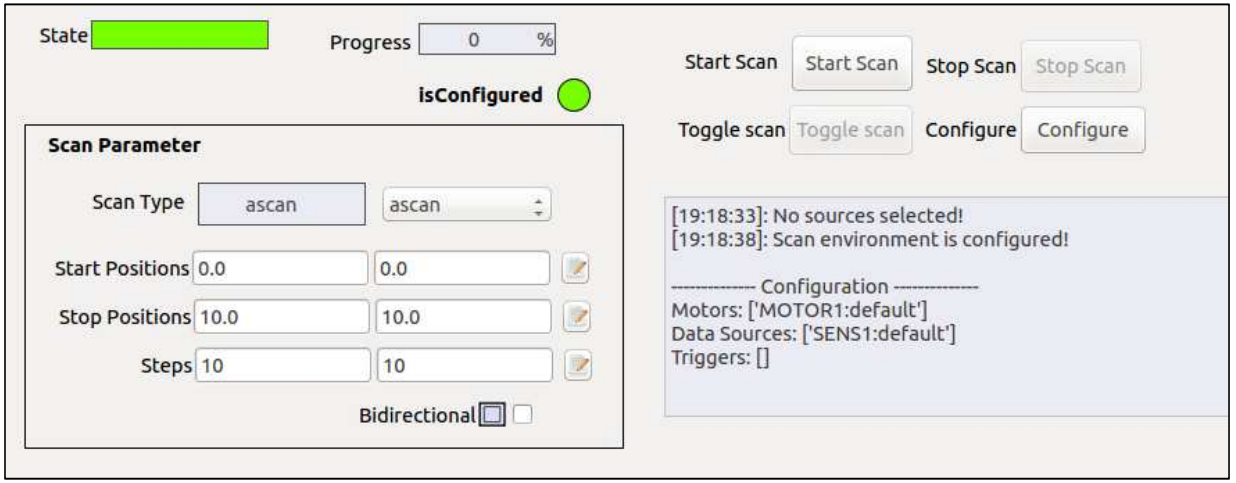
- Configure run-by-run which data to store:
  - select predefined sets of data sources (~devices)
- Choose to
  - ignore data,
  - monitor data, i.e. just feed online analysis pipeline,
  - start/stop run



Scan Tool

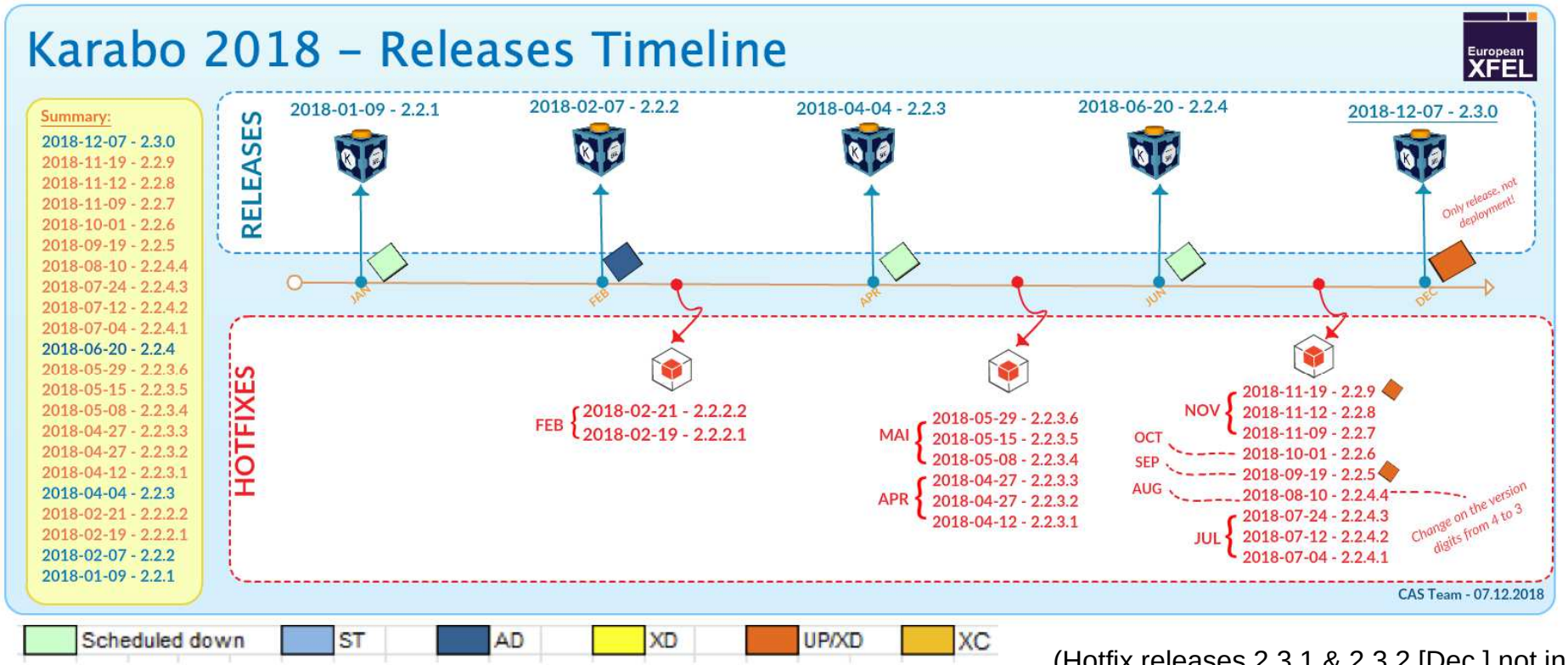
- DAQ-integrated step scans:
  - one scan - one DAQ run.
- GUI and (spec like) command line.
- Configure devices:
  - abstract motor(s): steps to take,
  - passive and active (“triggers”) data sources.
- Scan types
  - absolute – 1-4 motors: ascan, a2scan,…
  - relative – 1-4 motors: dscan, d2scan,…
  - 2D grid – absolute and relative: mesh, dmesh
- Plotting being extended to non-scalar data





Ongoing Karabo Development in 2018

- Five main, 19 hotfix releases
- Release schedule and deployment driven by EuXFEL schedule to minimise risks



(Hotfix releases 2.3.1 & 2.3.2 [Dec.] not in figure)

Karabo Release High Lights in 2018

- 2.2.2 February 2018: refurbished GUI client for speed
- 2.2.3 April 2018: Substantial stabilisation of servers with hundreds of devices
- 2.2.4 June 2018: Significant speed-up of serialisation in data pipelines
- 2.3.0 December 2018:
  - Configuration retrieval from past in GUI
  - Double-click access to parameter trendlines
  - Double-click access to device specific GUI scenes
  - Defined interfaces with CLI listing: motor, camera, trigger, processor, multi axis motor
  - Python 3.4.3 ➔ 3.6.6
  - GUI “cinema” mode to run scenes only
- All along:
  - Bug fixes, stability, performance, minor features, harmonizing APIs, improve interface to accelerator control system DOOCS,…

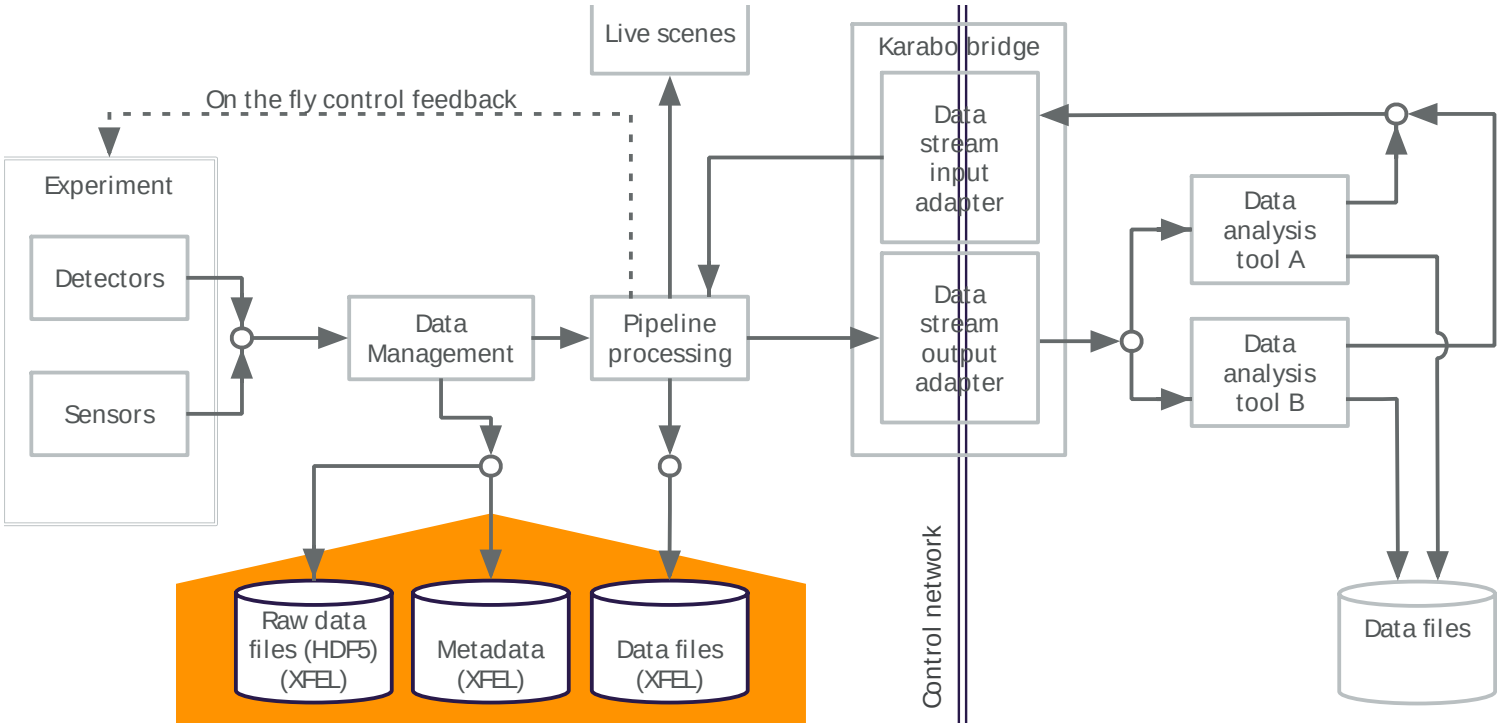
Karabo:
Supervisory Control and Data Acquisition (SCADA) at work!

- More than 10 large Karabo “ecosystems” for instruments and photon beamlines
  - Nov. 2018 count:
    - ► more than 400 installations (plus those for GUI clients),
    - ► 8266 devices (excl. Karabo services),
    - ► 1.087 million of control points
  - Communication in between ecosystems being improved

- 2,026 TB of raw data stored in 2018 (558 in 2017)

- Performance experiences:
  - Single device can stably consume 2 kHz of messages
    - ► production systems have usually an overall rate 50-500 Hz
  - Latencies at GUI server << 100 ms
  - Online calibration pipelines run smoothly with 2 s latency and 1.79 GB/s throughput

Online Data Analysis



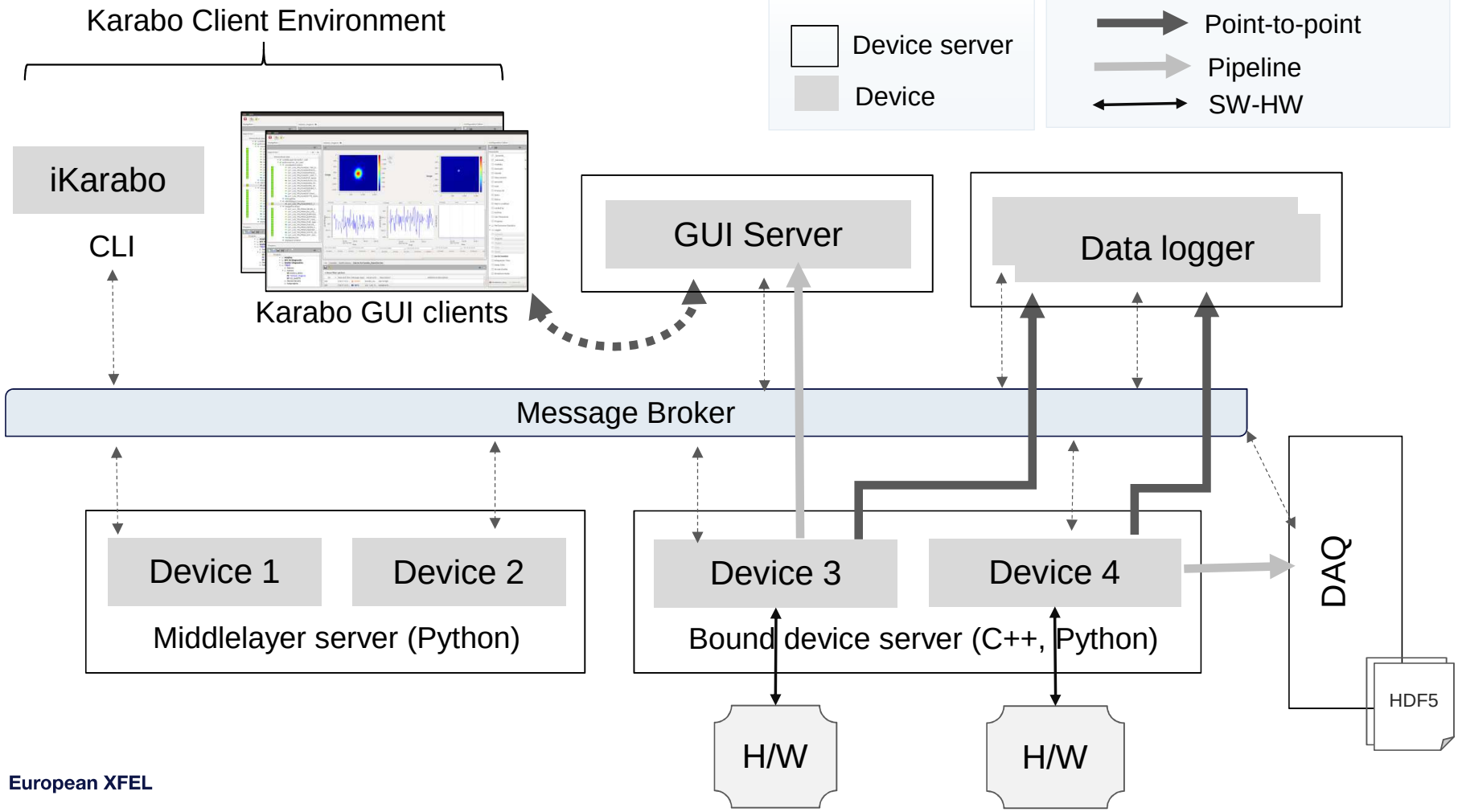
- Karabo goes beyond
  - data on tape
  - online preview
- Online pipeline can be extended by analysis devices
- Karabo bridge device opens space of Karabo agnostic user tools via ZeroMQ
- Details in later talks

Collection of open and internal documentation:
https://in.xfel.eu/readthedocs/docs/docs-collection/en/latest/

Summary

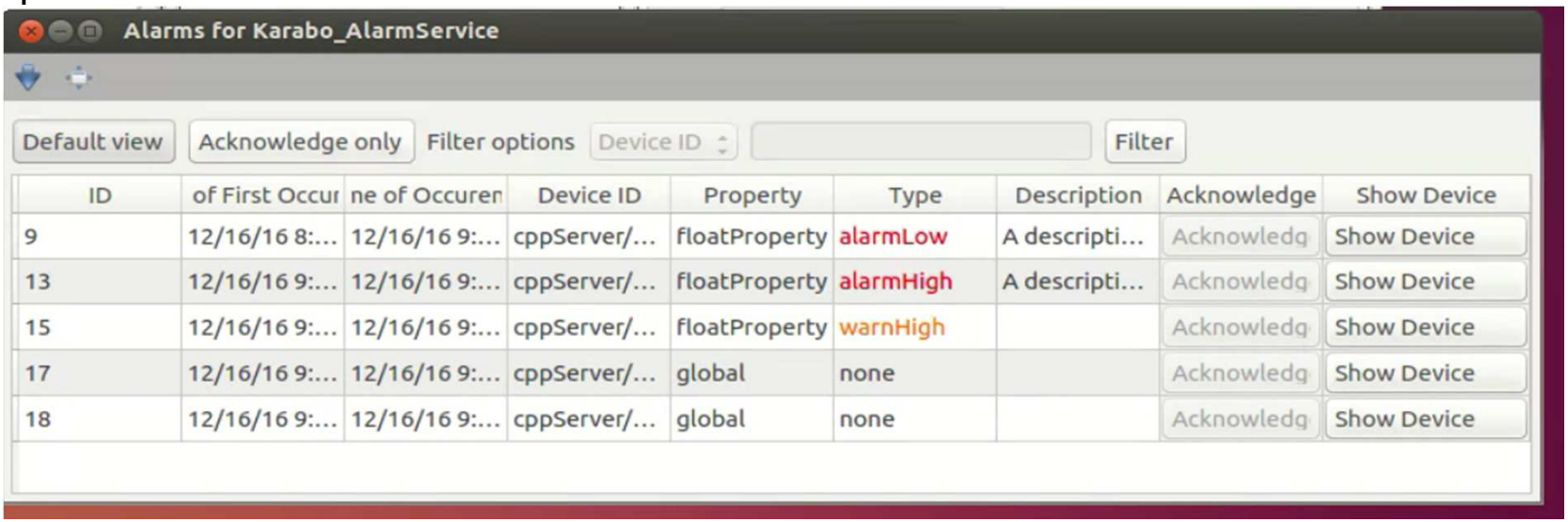
- Karabo, an integrated framework for
  - Instrument Control
  - Data Acquisition
  - Data Management
  - Data Analysis
- Broker based communication
- Workflow support through data pipelines, e.g. for online monitoring
- Four APIs (C++, Python) for extensions
- GUI easily customizable via scenes (by operator or device programmer)
- Experiment automation using macros and scan tools
- Online analysis with external tools

Architecture



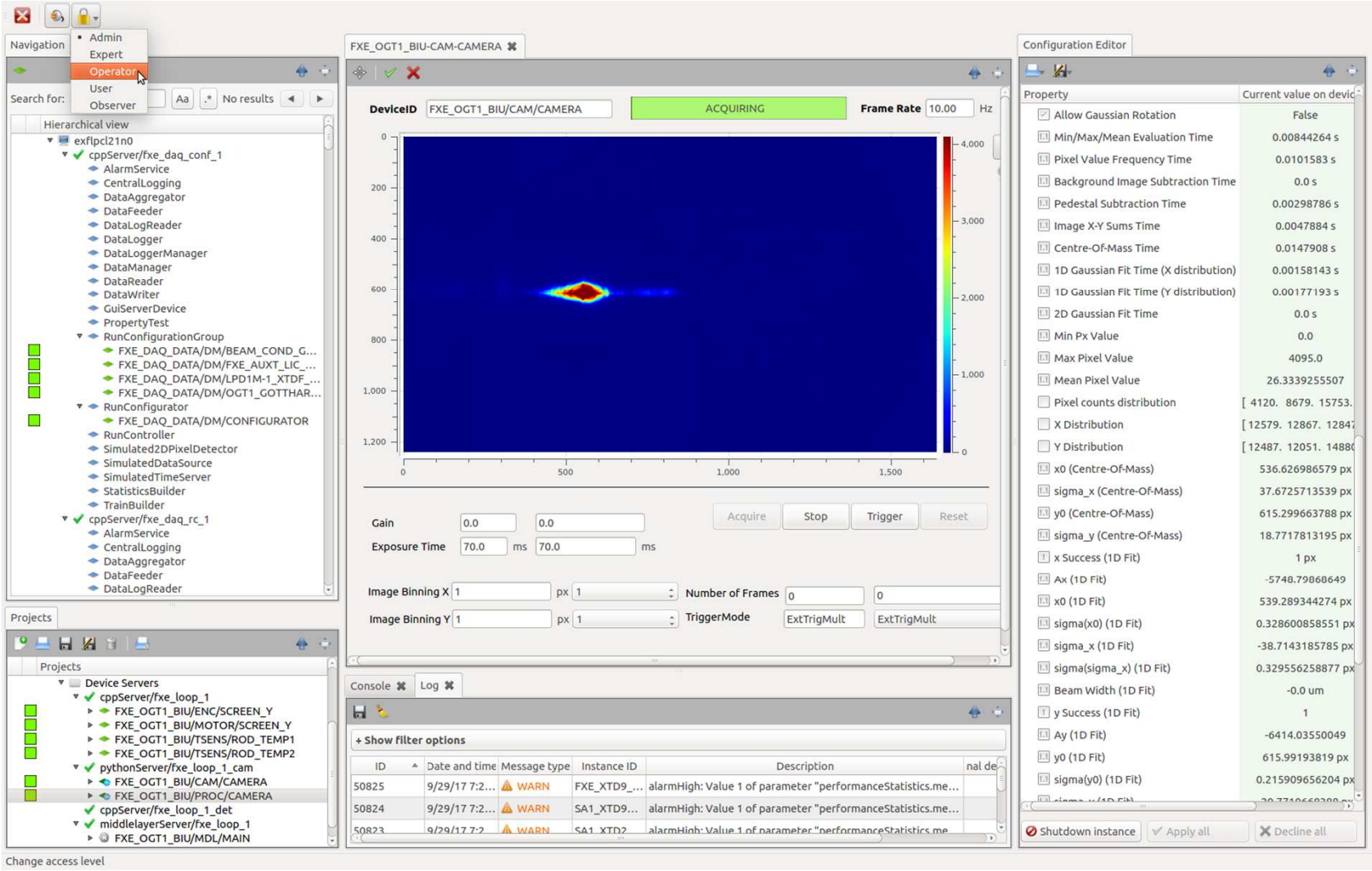
Alarms

- Reports any trespassing of property-specific thresholds
- Can be investigated from the GUI (and CLI) and acknowledged as soon as the alarm condition has passed



Scene Example

Experiment Control:
FXE's Beam Imaging Unit



Introduction to Value Displays

- Karabo comes with a set of standard visualization options for:
  - Scalars
  - Vectors
  - Images
  - Image stacks

- For scalars: value field, expression evaluation, trendline, sparklines, x-y plots, bit fields, checkboxes

- For vectors: value fields, x-y plots, categorial plot

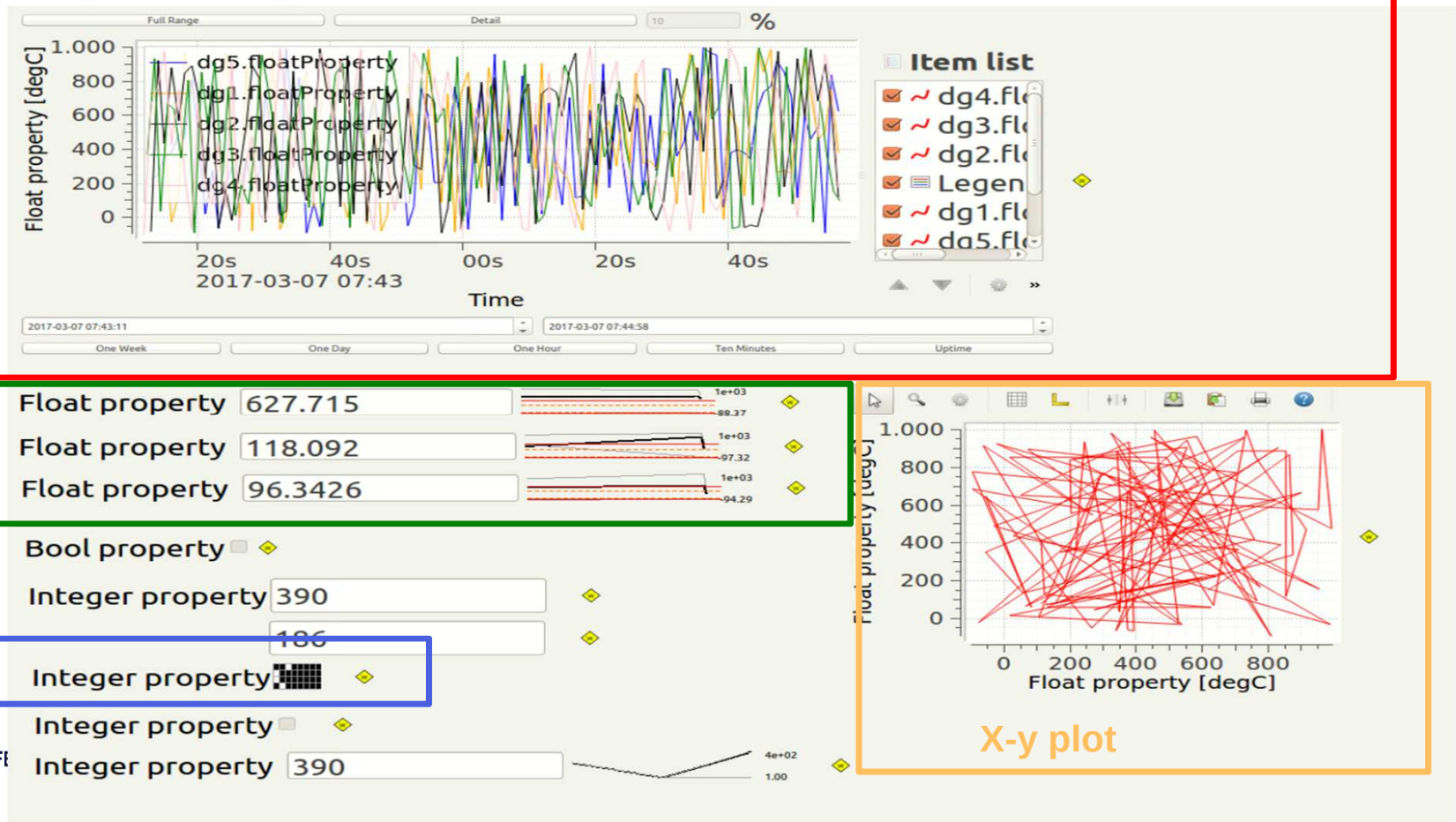
- For images: scientific, webcam, minimal, stacked

Introduction to Value Displays - Scalars

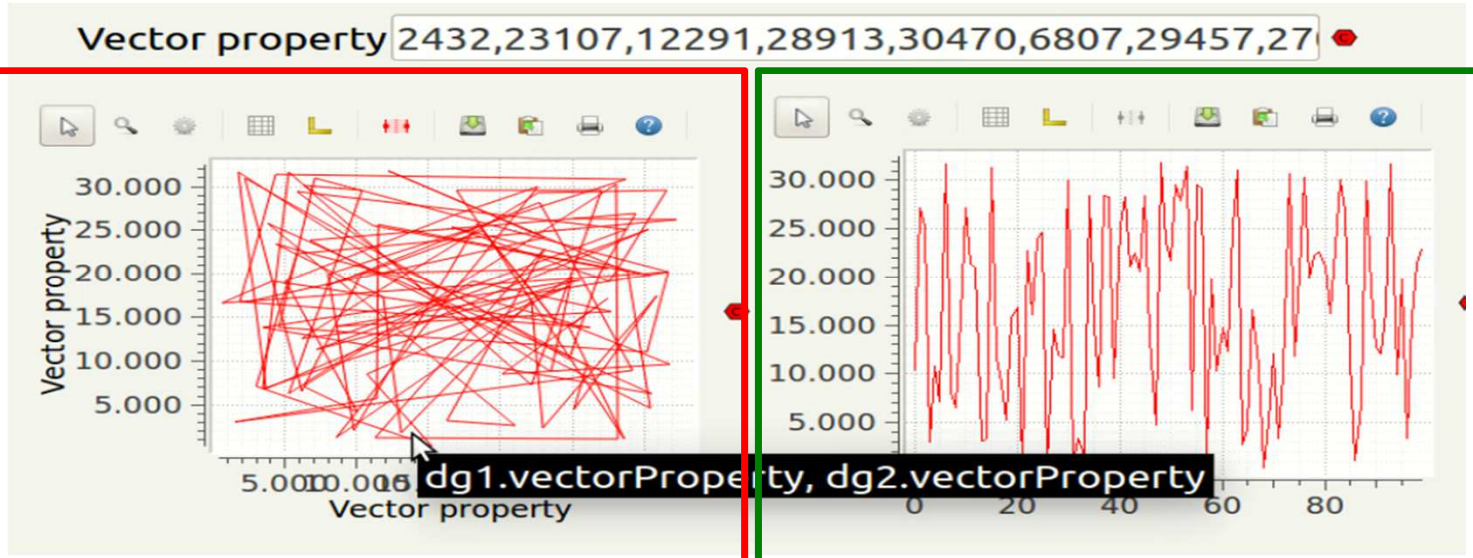
Trendline

Sparkline

BitField



Introduction to Value Displays - Vectors

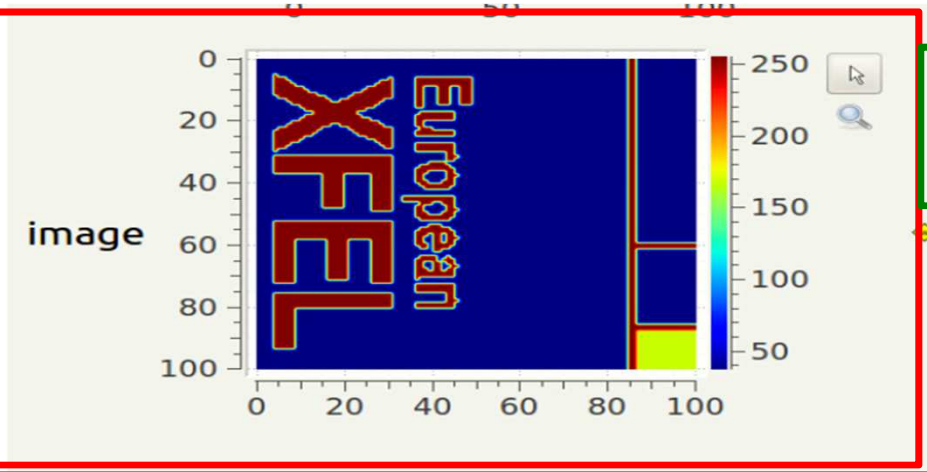


X-y plot

Categorial plot

Introduction to Value Displays - Images

Webcam



Scientific, stacked