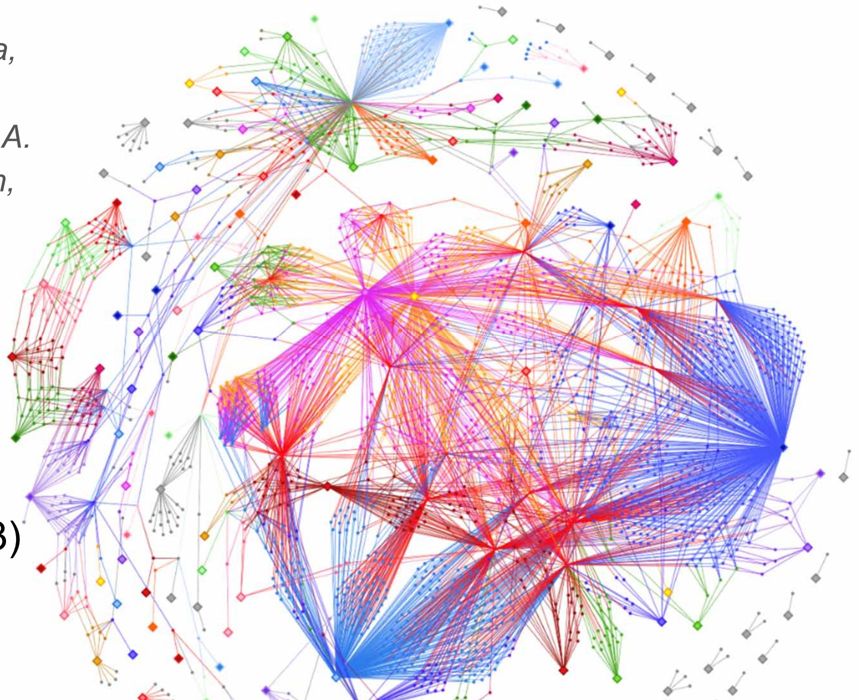
The Karabo Control System

S. Hauf, N. Anakkappalla, J. H. Bin Taufik, V. Bondar, R. Costa, W. Ehsan, S. Esenov, G. Flucke, A. Garcia-Tabares, G. Giovanetti, D. Goeries, D. Hickin, I. Karpics, A. Klimovskaia, A. Parenti, A. Samadli, H. Santos, A. Silenzi, M. A. Smith, F. Sohn, M. Staffehl, C. Youngman*

**steffen.hauf@xfel.eu*

19TH INTERNATIONAL CONFERENCE ON ACCELERATOR AND LARGE EXPERIMENTAL PHYSICS CONTROL SYSTEMS (ICALEPCS 2023)

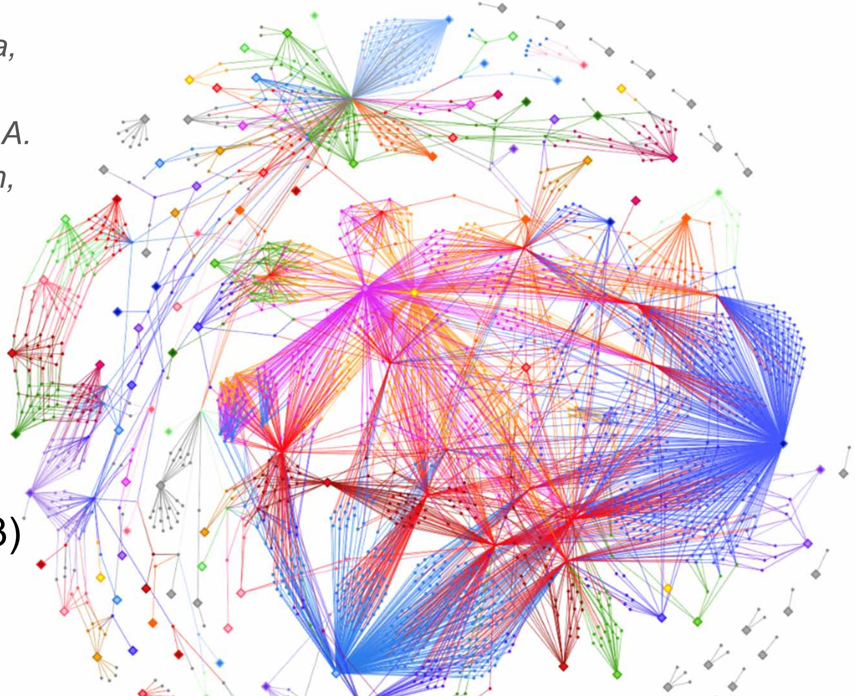


Tswana: **the answer**

The Karabo Control System

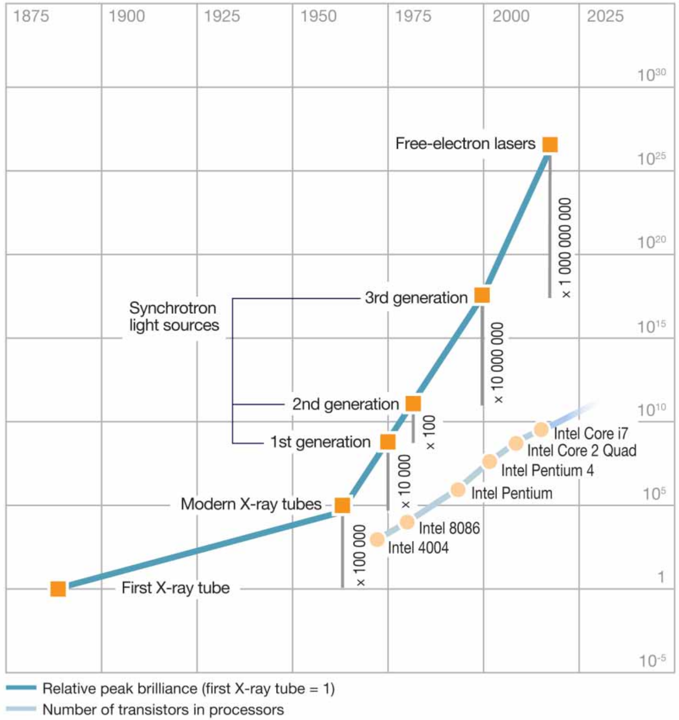
S. Hauf, N. Anakkappalla, J. H. Bin Taufik, V. Bondar, R. Costa, W. Ehsan, S. Esenov, G. Flucke, A. Garcia-Tabares, G. Giovanetti, D. Goeries, D. Hickin, I. Karpics, A. Klimovskaia, A. Parenti, A. Samadli, H. Santos, A. Silenzi, M. A. Smith, F. Sohn, M. Staffehl, C. Youngman*

**steffen.hauf@xfel.eu*



19TH INTERNATIONAL CONFERENCE ON ACCELERATOR AND LARGE EXPERIMENTAL PHYSICS CONTROL SYSTEMS (ICALEPCS 2023)

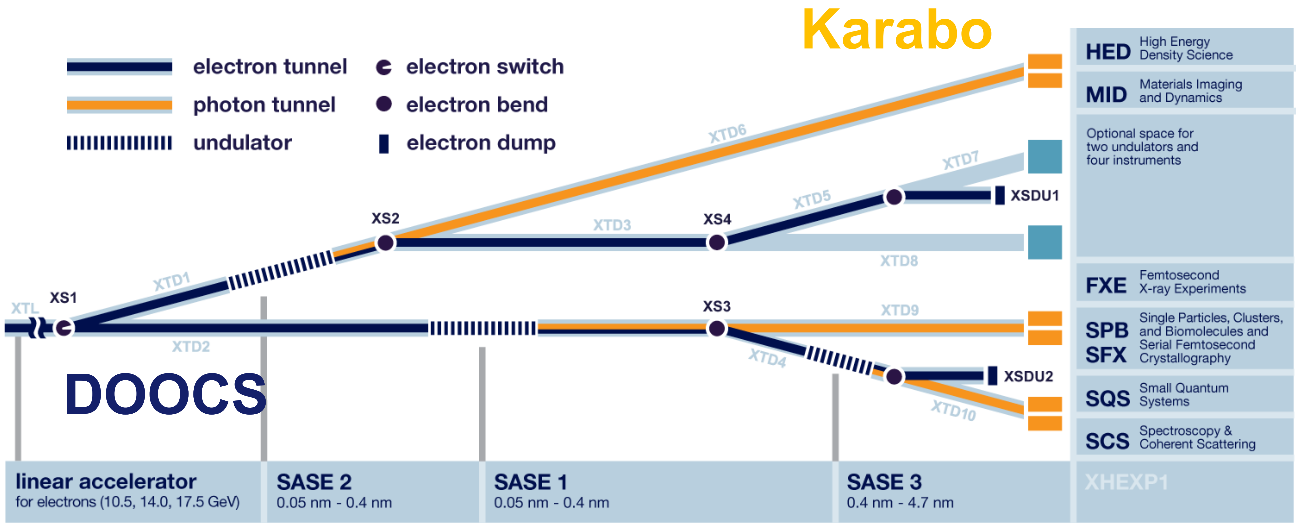
The European XFEL



* The development of light source facilities has been faster than the increase in computer processing capacity (i.e., Moore’s Law)

* We see this in the amount of data generated. For EuXFEL this can be multiple **PetaByte/week.** The Data Acquisition System is implemented in Karabo, as are the starting points of the online preview systems which support near-realtime processing of **>3kHz Mpixel images** .

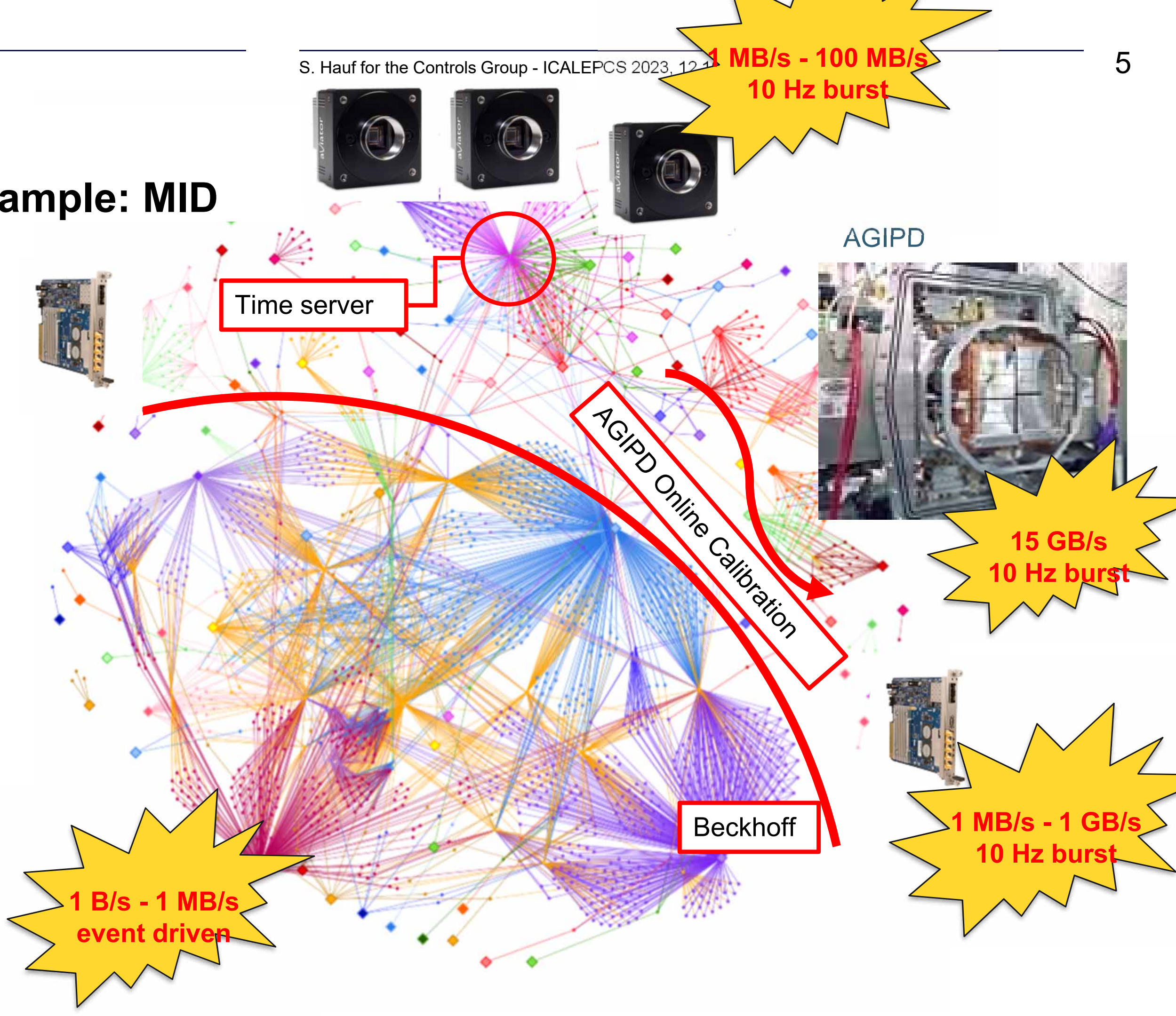
Beamline layout and experiment stations



* Diagnostic data from the accelerator is important for experiment analysis
* Bridges between DOOCS and Karabo are collaboratively maintained with DESY and facilitate transfer between the two systems (see also TUMBMO3)

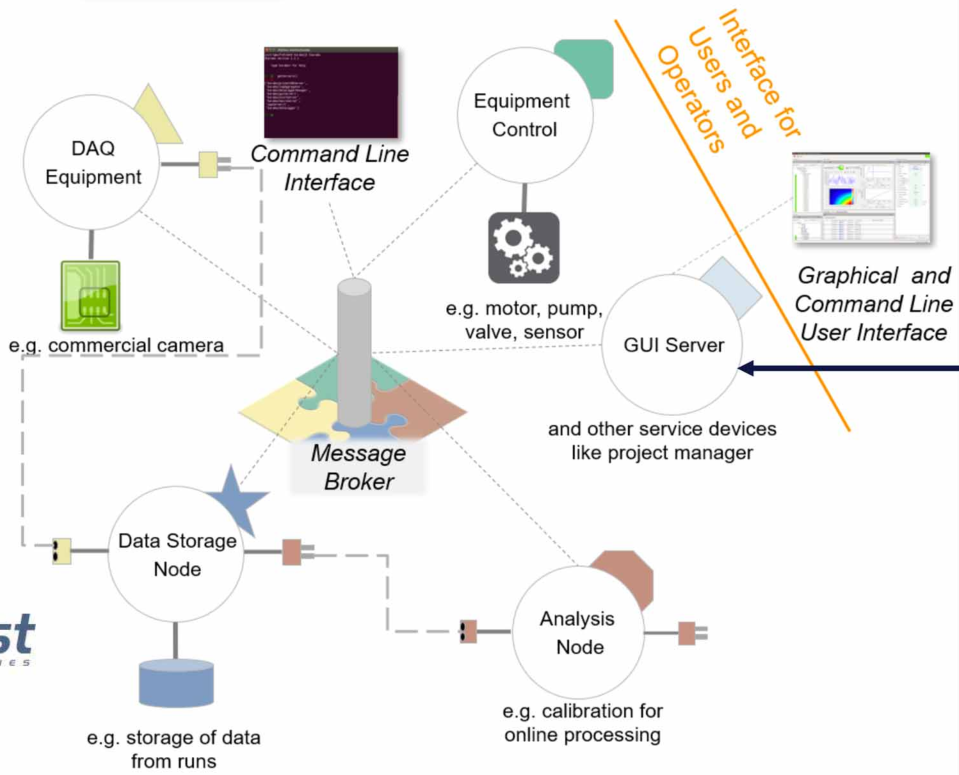
Karabo connects it all – Example: MID

* Key compoenents to look out for in all topics:
  * Timeserver: a central communication point for timing information
  * Bespoke MHz imaging detectors
  * Commercial cameras
  * Fast digitizers
  * Large Beckhoff loops, often interconnected via middlelayer devices and interlock conditions
  * Processing pipelines, e.g. detector calibration



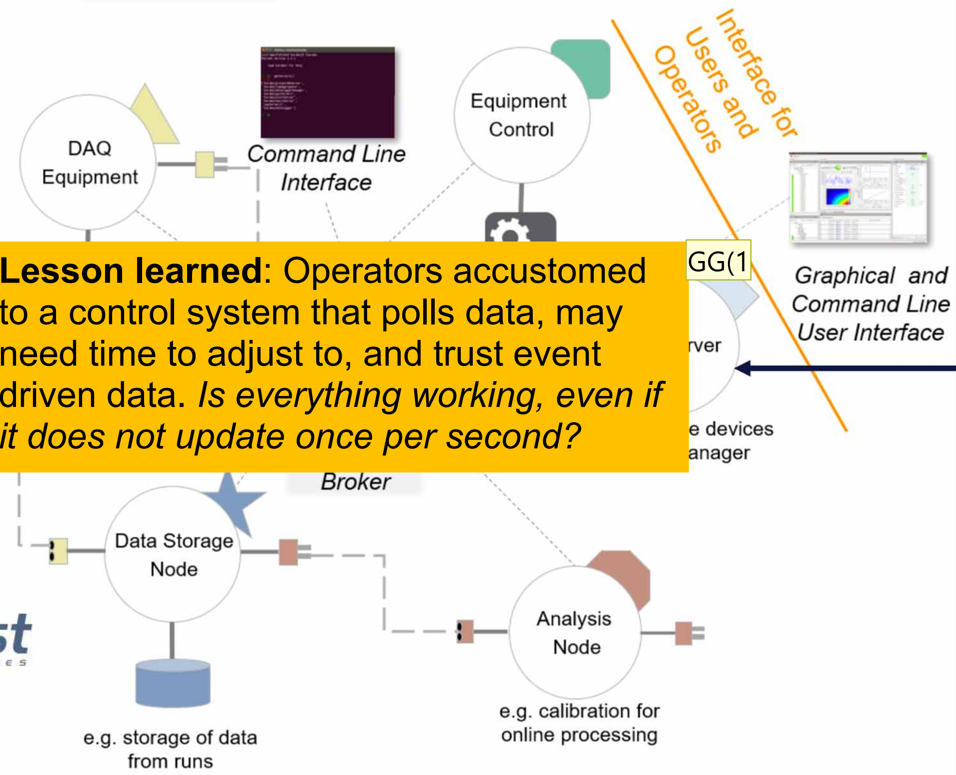
Karabo - Architecture

* Central Message Broker (Control and slow data)
  * Currently: OpenMQ
  * Soon RabbitMQ.
* Event driven:
  * Data propagates through the system when values change – push not poll
* Message driven:
  * Signal – Slot paradigm
  * Asynchronous core, synchronous convenience in middleware



Karabo - Architecture

* Central Message Broker (Control and slow data)
  * Currently: OpenMQ
  * Soon RabbitMQ.
* Event driven:
  * Data propagates through the system when values change – **push not poll**
* Message driven:
  * Signal – Slot paradigm
  * Asynchronous core, synchronous convenience in middleware



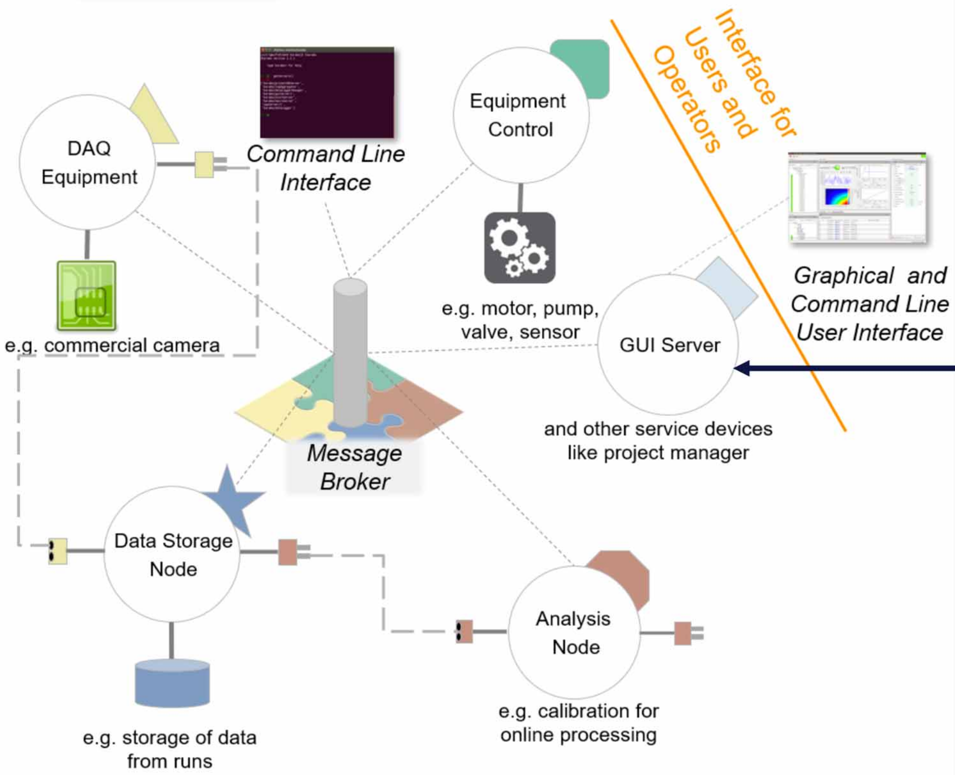
Lesson learned: Operators accustomed to a control system that polls data, may need time to adjust to, and trust event driven data. *Is everything working, even if it does not update once per second?*

Karabo - Architecture

- pipeline (p2p) connections (scientific/large) data
  - Scatter/Gather/Copy/Distribute
  - Block/Drop on congestion
  - TCP
  - Also GUI Server – GUI client
  - Capable of saturating a 10G line

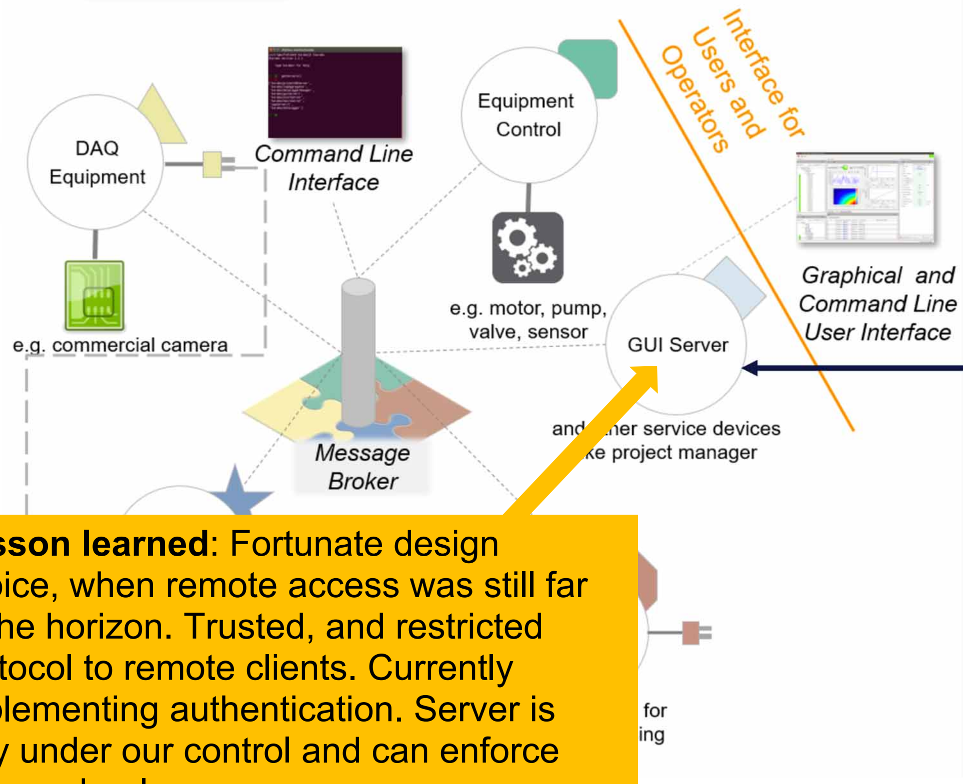
- Dynamic, discoverable topology
  - No central database instance

- GUI Server:
  - Gateway to the Control system



Karabo - Architecture

* pipeline (p2p) connections (scientific/large) data
  * Scatter/Gather/Copy/Distribute
  * Block/Drop on congestion
  * TCP
  * Also GUI Server – GUI client
  * Capable of saturating a 10G line
* Dynamic, discoverable topology
  * No central database instance
* GUI Server:
  * Gateway to the Control system



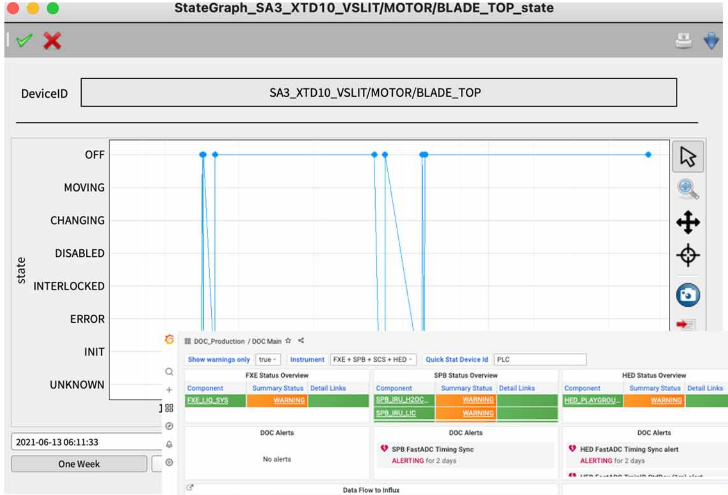
Lesson learned: Fortunate design choice, when remote access was still far at the horizon. Trusted, and restricted protocol to remote clients. Currently implementing authentication. Server is fully under our control and can enforce e.g. read-only access.

Metrics in Influx: > 240 Billion
Increase per month: ~ 10 Billion

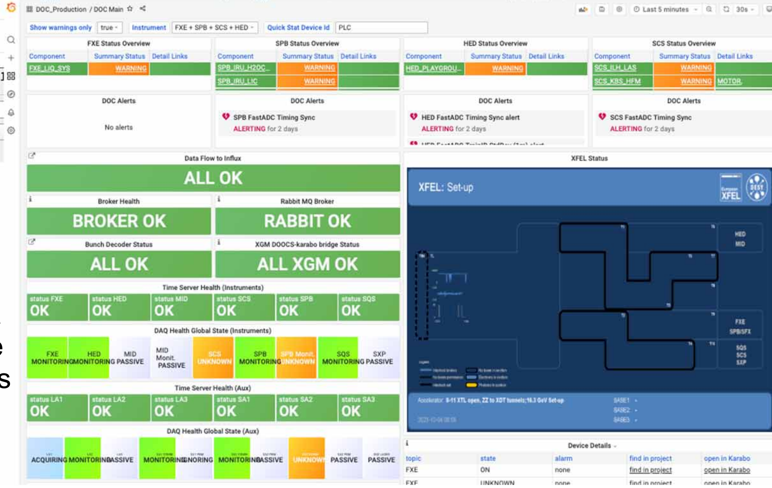
Karabo – Data Logging using the Influx Time Series Database

* Datalogging vs. Data Acquisition
  * Datalogging is continous for slow (broker) data
    - ▶ It is done by default
    - ▶ For all devices
    - ▶ Internal data product for maintenence
  * Data Acquisition is „run“ based
    - ▶ Explicitely started
    - ▶ Includes large and fast data
    - ▶ Subselection of slow data
    - ▶ Data product for facility users
    - ▶ Multiple PB per week per instrument

* Karabo dataloggers
  * Proprietary text-based format
  * **Influx Time-series based**



Trendline in Karabo showing time evolution of device state.



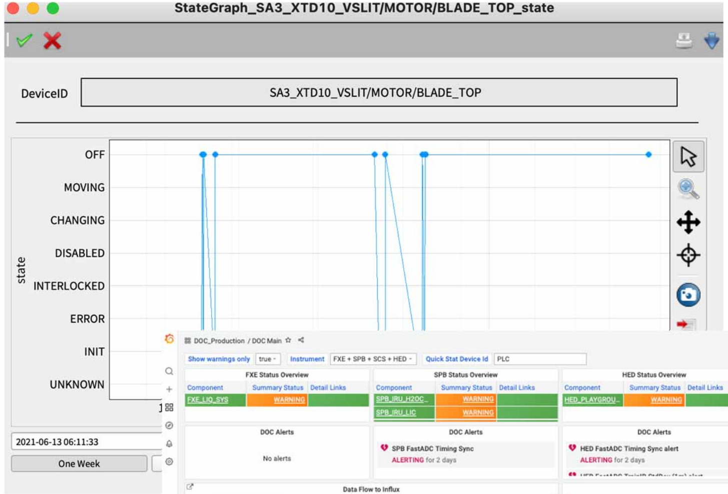
Data Operation Center (DOC) main Grafana panel. Most data here is from the Karabo Influx loggers

Metrics in Influx: > 240 Billion
Increase per month: ~ 10 Billion

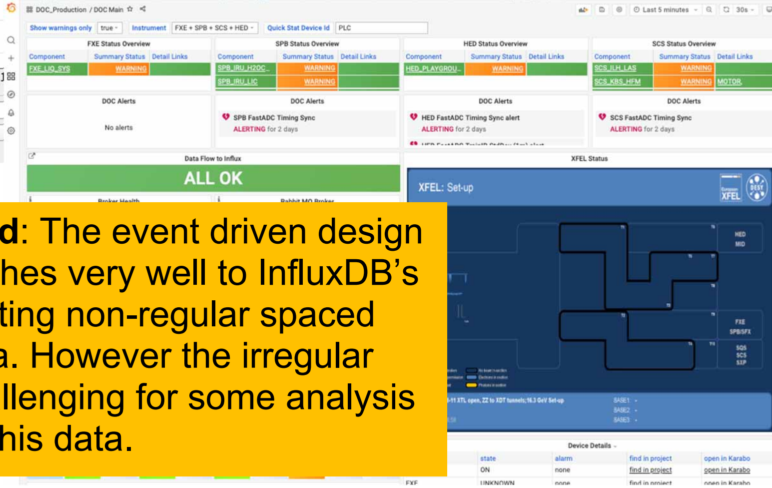
Karabo – Data Logging using the Influx Time Series Database

* Datalogging vs. Data Acquisition
  * Datalogging is continous for slow (broker) data
    - ▶ It is done by default
    - ▶ For all devices
    - ▶ Internal data product for maintenence
  * Data Acquisition is „run“ based
    - ▶ Explicitely started
    - ▶ Includes large and fast data
    - ▶ Subselection of slow data
    - ▶ Data product for facility users
    - ▶ Multiple PB per week per instrument

* Karabo dataloggers
  * Proprietary text-based format
  * **Influx Time-series based**



Trendline in Karabo showing time evolution of device state.



Lesson learned: The event driven design of Karabo matches very well to InfluxDB's design of ingesting non-regular spaced time series data. However the irregular grid can be challenging for some analysis tasks that use this data.

Karabo – C++, Karabo Bound, and Karabo Middlelayer APIs

General	C++ API	Python Bound API	Middlelayer API
• Event driven • Asynchronous • Self-descriptive • Common, hierarchical data container supporting attributes on leafs: Karabo Hash • Binary and XML serialization • Extensible: core + “Devices“	• C++14 and Boost • Smart pointers • Template-heavy • Boost.asio • Eventloop based • Devices are threads on a single server • Aimed at high-performance devices	• Exposes C++ API via Boost.Python • Devices are separate processes • Was aimed at p2p heavy devices which e.g. need numpy • Not always pythonic	• Python asyncio • Decorators annotate Karabo structures • Emphasis on interaction with other devices • Pythonic

Common Example: Concurrent Motion

From Bluesky documentation

```
from ophyd.sim import motor1, motor2

# Move motor1 to 1 and motor2 10 units in the positive direction relative
# to their current positions. Wait for both to arrive.
RE(bps.mvr(motor1, 1, motor2, 10))
```

Karabo Middlelayer

```
[2]: motors = [await connectDevice(motorId) for motorId in motorIds]
...: for device, position in zip(motors, positions):
...:     device.targetPosition = position
...:     futures.append(device.move())
...:
...: await gather(*futures)
...: await waitUntil(lambda: all(dev.state != State.MOVING for dev in motors))
```

Other systems:
- Bliss e.g. motor_group
- Sardana: moveMultiple
- ….
- Karabacon: MotorGroup

Common Example: Concurrent Anything…

Motion

```
[2]: motors = [await connectDevice(motorId) for motorId in motorIds]
...: for device, position in zip(motors, positions):
...:     device.targetPosition = position
...:     futures.append(device.move())
...:
...: await gather(*futures)
...: await waitUntil(lambda: all(dev.state != State.MOVING for dev in motors))
```

Power Supplies

```
...:
...:     for device, voltage in zip(mpodGroups, voltage):
...:         device.voltage = voltage
...:         futures.append(device.on())
...:
...:     await gather(*futures)
...:     await waitUntil(lambda: all(dev.state == State.ON for dev in devices))
```

Instantiation

```
...:     for serverId, classId, deviceId, config in offlineDevices:
...:
...:         futures.append(instantiate(serverId, classId, deviceId, config))
...:
...:     await gather(*futures)
...:
```

Common Example: Concurrent Anything…

Motion

```
[2]: motors = [await connectDevice(motorId) for motorId in motorIds]
...: for device, position in zip(motors, positions):
...:     device.targetPosition = position
...:     futures.append(device.move())
...:
...: await gather(*futures)
...: await waitUntil(lambda: all(dev.state != State.MOVING for dev in motors))
```

Karabo Middlelayer prefers using Python "primitives" where possible over domain specific extensions.

→ Learn once – use often

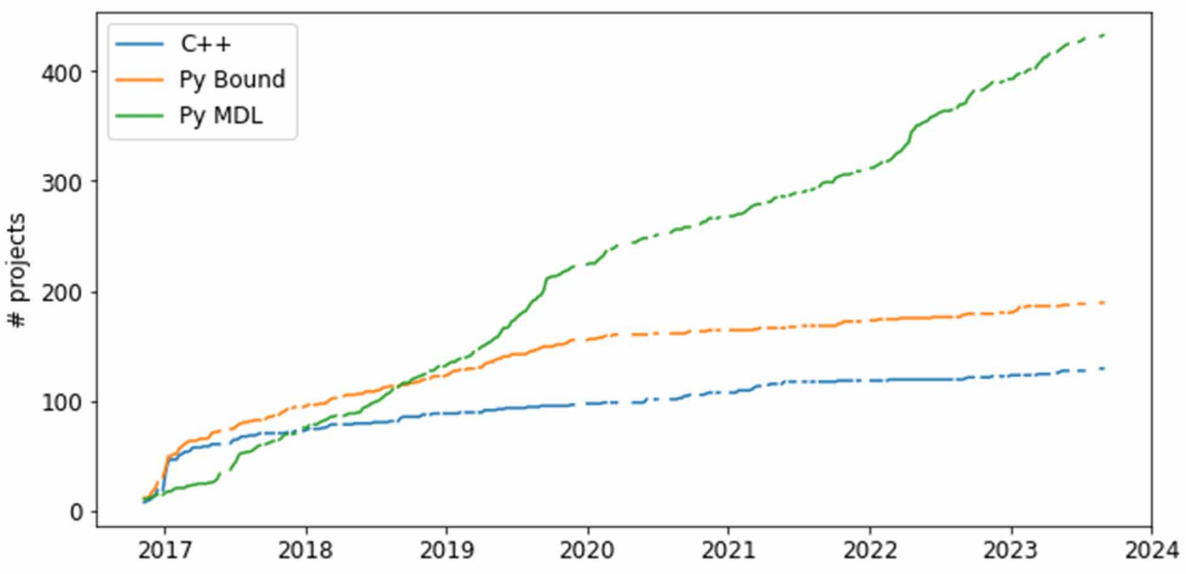
```
...:         futures.append(device.on())
...:
...:     await gather(*futures)
...:     await waitUntil(lambda: all(dev.state == State.ON for dev in devices))
```

Instantiation

```
...:     for serverId, classId, deviceId, config in offlineDevices:
...:
...:         futures.append(instantiate(serverId, classId, deviceId, config))
...:
...:     await gather(*futures)
...:
```

The Karabo Ecosystem – Usage of the three APIs

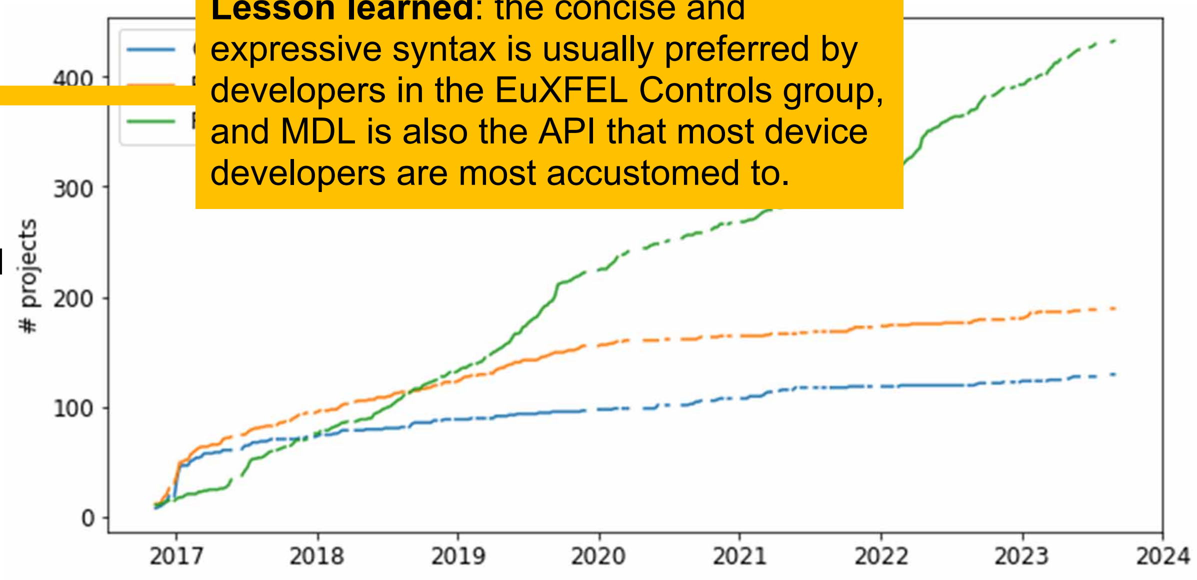
- Middlelayer API frequently has the most expressive syntax coupled with shortest “time-to-market”.
- C++ and Python Bound are actively maintained and new devices are still being implemented, especially in high-performance fields.



The Karabo Ecosystem – Usage of the three APIs

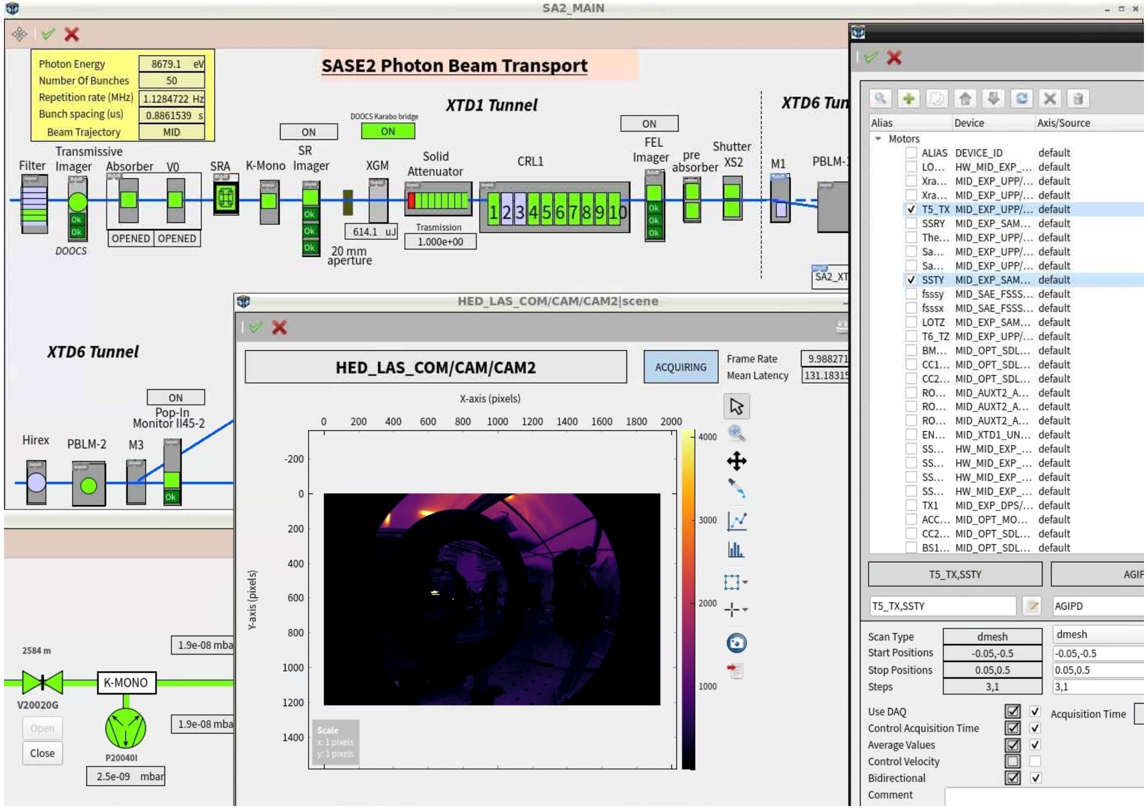
- Middlelayer API frequently has the most expressive syntax coupled with shortest “time-to-market”.
- C++ and Python Bound are actively maintained and new devices are still being implemented, especially in high-performance fields.

Lesson learned: the concise and expressive syntax is usually preferred by developers in the EuXFEL Controls group, and MDL is also the API that most device developers are most accustomed to.



Karabo - The Karabo GUI

- Separate Python Package, well matched to the framework
- PyQt5
- Connects to Karabo via the GUI-server (tcp, p2p)
- Extensible via „gui-extensions“
- **Distinguishing features:**
  - **GUI scene builder**
  - **Projects to logically group devices, scenes and macros**

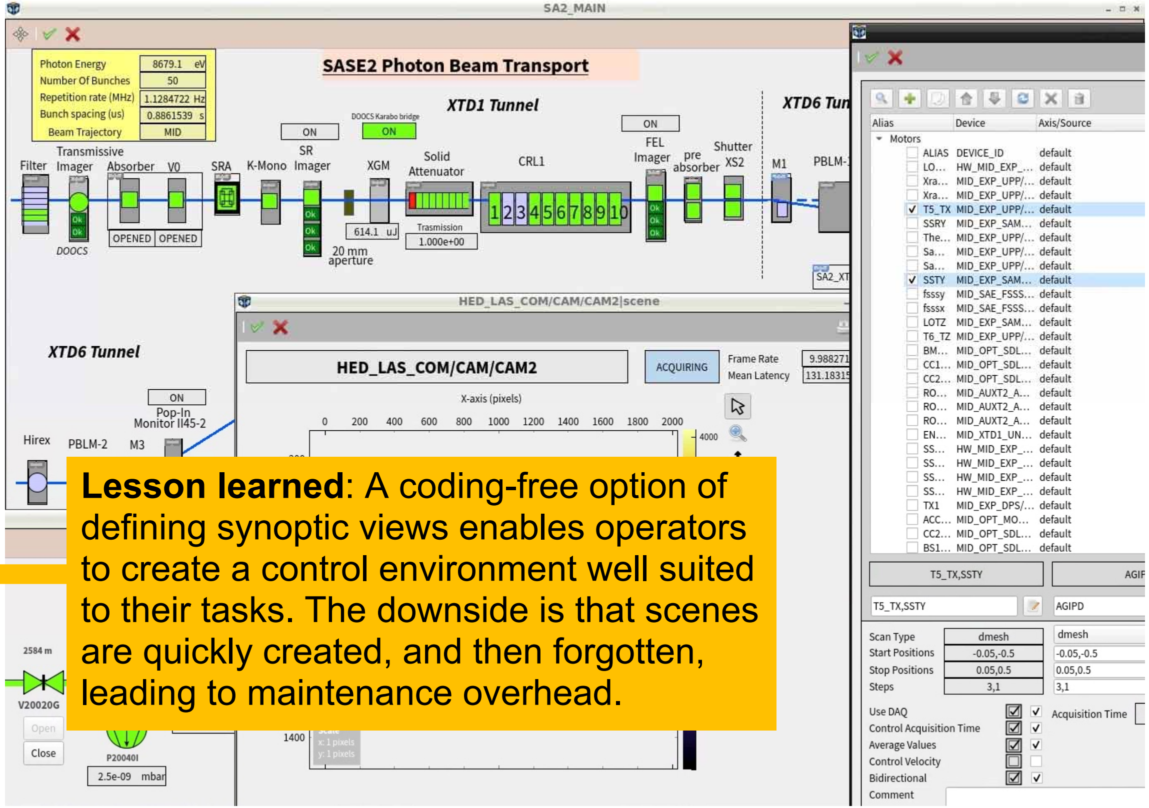


Karabo - The Karabo GUI

* Separate Python Package, well matched to the framework
* PyQt5
* Connects to Karabo via the GUI-server (tcp, p2p)
* Extensible via „gui-extensions“
* **Distinguishing features:**
  * **GUI scene builder**
  * **Projects to logically group devices, scenes and macros**

Karabo - The Karabo GUI

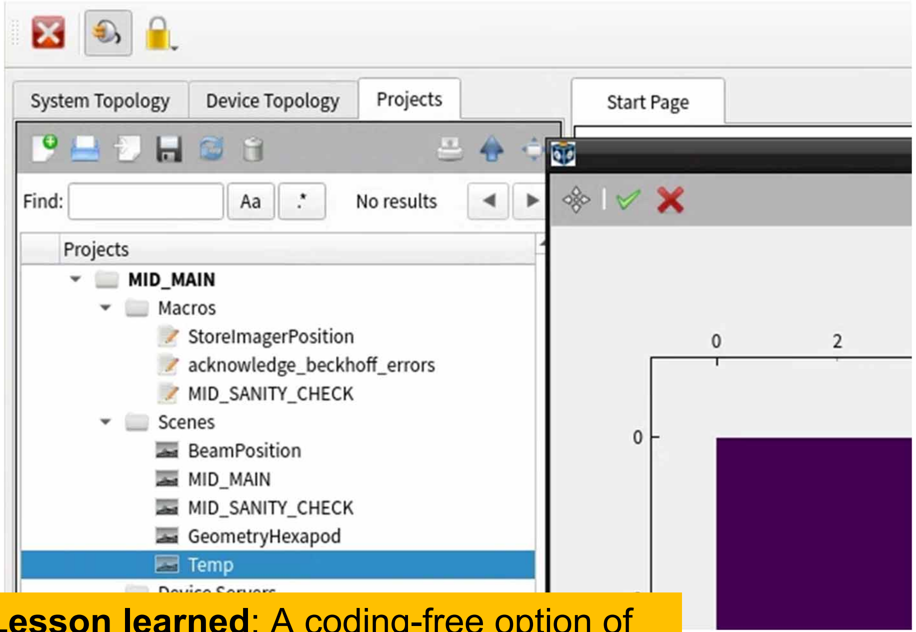
* Separate Python Package, well matched to the framework
* PyQt5
* Connects to Karabo via the GUI-server (tcp, p2p)
* Extensible via „gui-extensions“
* **Distinguishing features:**
  * **GUI scene builder**
  * **Projects to logically group devices, scenes and macros**



Lesson learned: A coding-free option of defining synoptic views enables operators to create a control environment well suited to their tasks. The downside is that scenes are quickly created, and then forgotten, leading to maintenance overhead.

Karabo - The Karabo GUI

* Separate Python Package, well matched to the framework
* PyQt5
* Connects to Karabo via the GUI-server (tcp, p2p)
* Extensible via „gui-extensions“
* **Distinguishing features:**
  * **GUI scene builder**
  * **Projects to logically group devices, scenes and macros**



Lesson learned: A coding-free option of defining synoptic views enables operators to create a control environment well suited to their tasks. The downside is that scenes (and projects) are quickly created, and then forgotten, leading to maintenance overhead.

Karabo Hardware Devices – an incomplete list of what is supported

Core Services

- Data Logging
- Data Acquisition
- Gui Server
- Alarms & Notification
- Recovery Portal
- Motor Configurator
- Scan Tool

Motion

- Beckhoff MC2*
- Smaract
- Hexapod
- NanoCube
- …

Commercial Cameras

- Basler via Aravis
- Genicam
- GreatEyes
- PI MTE3
- Andor Zylar, …
- Shimadzu HPvx

X-ray Detectors

- Jungfrau
- Gotthard
- Epix
- AGIPD
- LPD
- DSSC
- Timepix3
- …

Scopes and …meters

- OceanOptics
- GENTEC Tetronix
- LeCroy
- MCS Beam Stab.
- …

Digitizers

- SP devices ADQ 412
- SP devices ADQ 7
- SP devices ADQ 14
- FastADC

Power Supplies

- Wiener MPOD
- Keithley
- Agilant

Vacuum Components*

- Adixon
- Pfeiffer
- Infinicon
- Agilant

Chillers & Thermo Contr.*

- Huber
- K2
- Julabo
- Keithley
- Lakeshore
- …

Bridging

- SCPI
- DOOCS
- EPICS (in progress)
- Tango (in progress)

* via Beckhoff PLC integration

THPDP023 S. Samadli: Evolution of Control System and PLC Integration at the European XFEL

MO2BCO04 S. Huynh: Applying Standardised Software Architectural Concepts to Design Robust and Adaptable Plc

TUSDSC03 N. Mashayekh: Integrating Tools to Aid the Automation of PLC Development Within the TwinCat Environment

THPDP021 N. Coppola: Equipment life-cycle Management at EuXFEL

Karabo Hardware Devices – an incomplete list of w

Core Services

- Data Logging
- Data Acquisition
- Gui Server
- Alarms & Notification
- Recovery Portal

WE1BCO02 J. Malka: Data Management Infrastructure for European XFEL

Motion

MO2AO03 A. Garcia-Tabares: The Solid Sample Scanning Workflow at the European XFEL

THPDP024 F. Sohn: Automatic Configuration of Motors at the European XFEL

- NanoCube
- …

Commercial Cameras

- PI MTE3
- Andor Zylar, …
- Shimadzu HPvx

X-ray Detectors

- Jungfrau
- Gotthard
- Epix
- AGIPD
- LPD
- DSSC
- Timepix3
- …

Scopes and …meters

- OceanOptics
- GENTEC Tetronix
- LeCroy
- MCS Beam Stab.
- …

Digitizers

- SP devices ADQ 412
- SP devices ADQ 7
- SP devices ADQ 14
- FastADC

MO4AO06 B. Fernandes: Overview and Outlook of Fpga Based Hardware Solutions for Data Synchronization, Acquisition and Processing at the Euxfel

Power Supplies

- Wiener MPOD
- Keithley
- Agilant

Vacuum Components*

- Adixon

THPDP022 B. Rio: Adaptable Control System for the Photon Beamlines at European XFEL: Integrating New Devices and Technologies for Advanced Research

Chillers & Thermo Contr.*

- Huber
- Lakeshore

TUPDP033 M. Smith: Applying Model Predictive Control to Regulate Thermal Stability for a Hard X-ray Monochromator Using the Karabo Software Control Framework

Bridging

- SCPI
- DOOCS
- EPICS (in progress)
- Tango (in progress)

* via Beckhoff PLC integration

Where is this going, and how can I get it?

Move to RabbitMQ and AMQP

RDMA – 80% IB link speed

Elog Integration

MDL on PyPi

WebUI

https://github.com/European-XFEL/Karabo

Authenticating & Authorizing Web API

{REST:API}*

GUI side Authentication and Authorization

ELK Stack for System Logs

2024

2025

← Back

2023/07/28

European XFEL control software Karabo released as open source

European XFEL has released the control software framework Karabo and selected Karabo devices into the public domain as free and open-source software, enabling external developers to use and adapt the code as they need. The extendable system can be used to control installations that range from single machines to highly complex research facilities, such as the European XFEL.

Download [9.5 MB. 5256 x 2932]

Since its first inception in 2011, the European XFEL control system has been developed into a modern, distributed software framework that enables control and monitoring of the photon systems and instrumentation at the facility, as well as data acquisition from X-ray detectors capable of megahertz frame rates. It is highly interoperable with DOOCS, a similar system developed at DESY that is used to control the European XFEL acc ator

European XFEL

WebUI

https://github.com/European-XFEL/Karabo

Web API

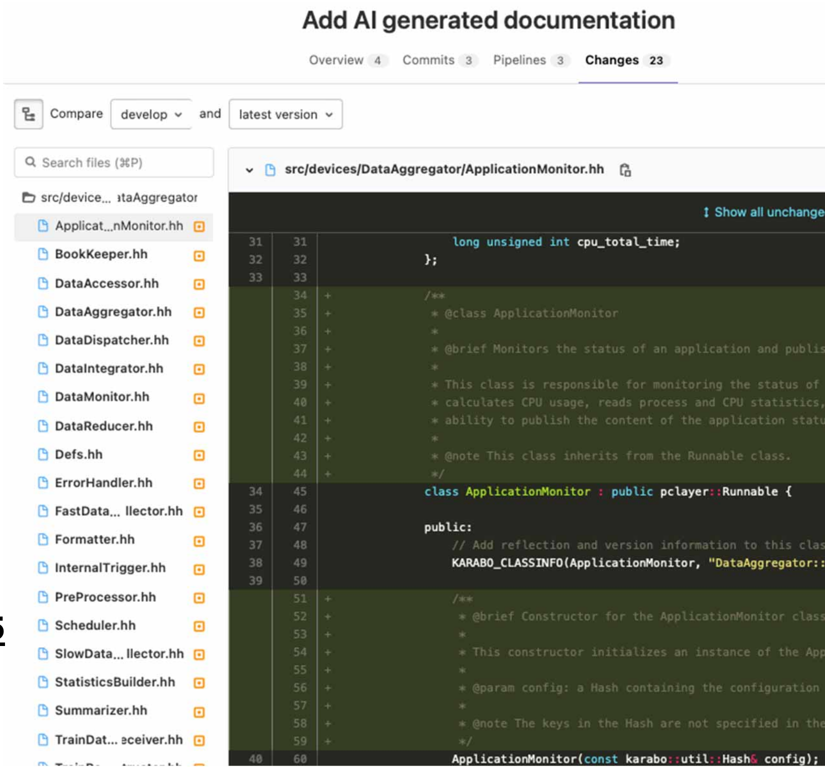
elasticsearch

ELK Stack for System Logs

Karabo and Large Language Model: towards AI assistants

* **LLMs as code documentation assistants**
  * System prompt to ask AI to add or update documentation to code using a diff format that doesn't require counting lines
  * Works well to batch-document code lacking most documentation
  * Karabo agnostic, available at https://github.com/European-XFEL soon

* **LLMs as coding assistants**
  * GPT4 was not trained on Karabo code at the time of tests
  * System prompts describing the MDL API and the scene model suffice
  * Iterative approach, feeding exceptions back into the model
  * **https://syncandshare.xfel.eu/index.php/s/kt6NbSjJfMg7Pf5**



Conclusion

* Karabo is a flexible SCADA framework with a modern event-driven and asynchronous core.
  * 3 APIs with individual strengths
  * State of the art condition logging using the Influx Time Series database
  * Highly scalable, and no central database authority required
* The Karabo GUI is closely matched to the framework
  * Can flexibly visualize all data types the framework provides
  * Synoptic views (scenes) without coding
  * Extensible
* LLMs can document and code in Karabo reasonable well and offer opportunities in improving development workflows.
* Karabo (and some devices) have recently been open sourced:

https://github.com/European-XFEL/Karabo

Survey about Knowledge Sharing among Software Developers and Scientists

Study objectives:

* Analyze the knowledge sharing practices in scientific software development
* Derive tool needs for developers and scientists

Survey:

* 10-20 min
* For **everyone** developing or using **control or data analysis systems** at scientific facilities
* **Win one of five 50€ vouchers** for an online shop of your choice

Access the survey [1]:

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

[1] https://umfragen.uni-hamburg.de/index.php/975348?lang=en