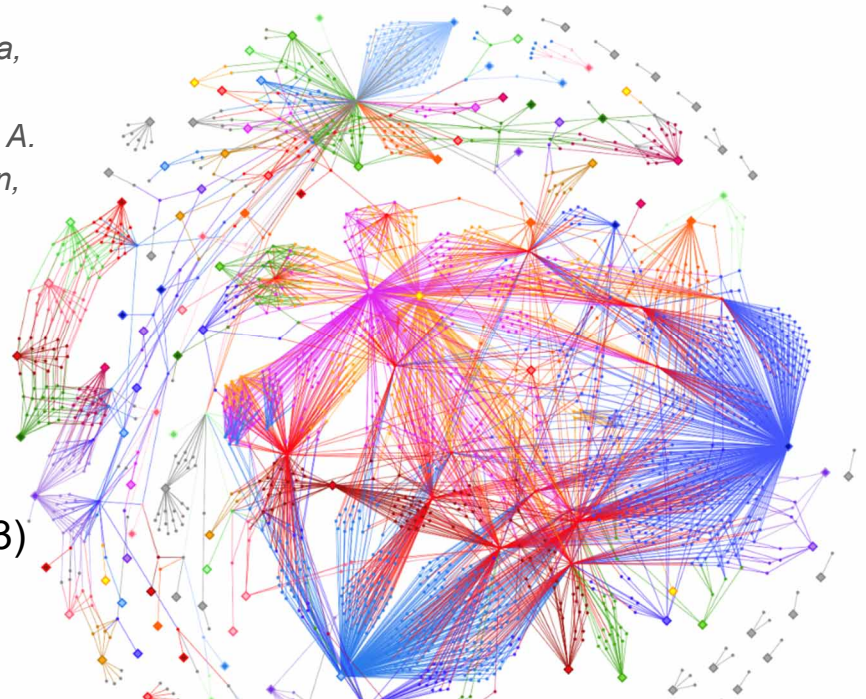


The Karabo Control System

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

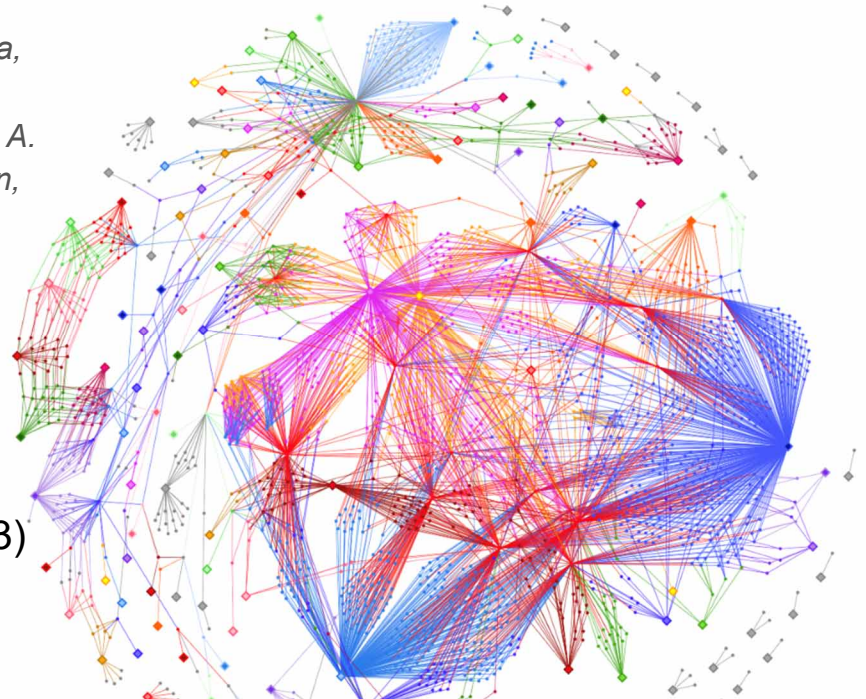


The **Karabo** Control System

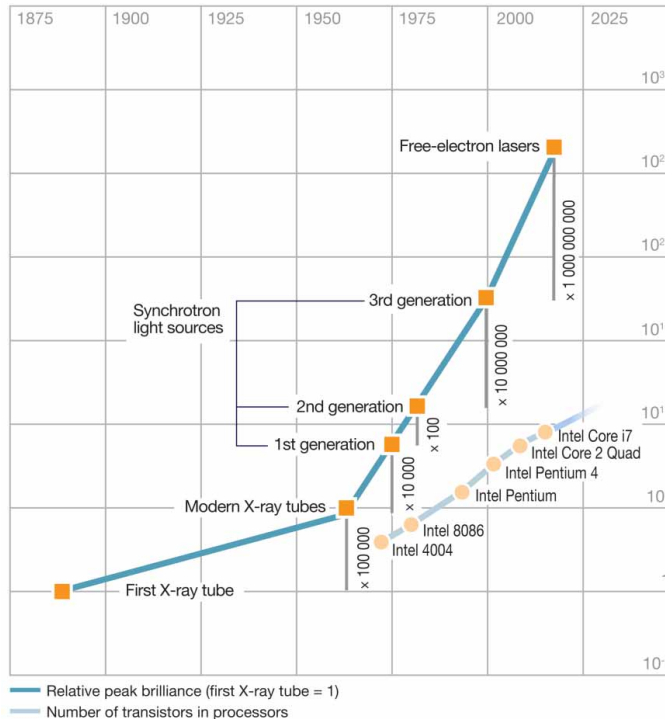
S. Hauf, N. Anakkappalla, J. H. Bin Taufik, V. Bondar, R. Costa,
W. Ehsan, S. Esenov, G. Flucke, A. Garcia-Tabares,
G. Giovanetti, D. Goeries, D. Hickin, I. Karpics, A. Klimovskaia, A.
Parenti, A. Samadli, H. Santos, A. Silenzi, M. A. Smith, F. Sohn,
M. Staffehl, [C. Youngman](#)*

**steffen.hauf@xfel.eu*

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

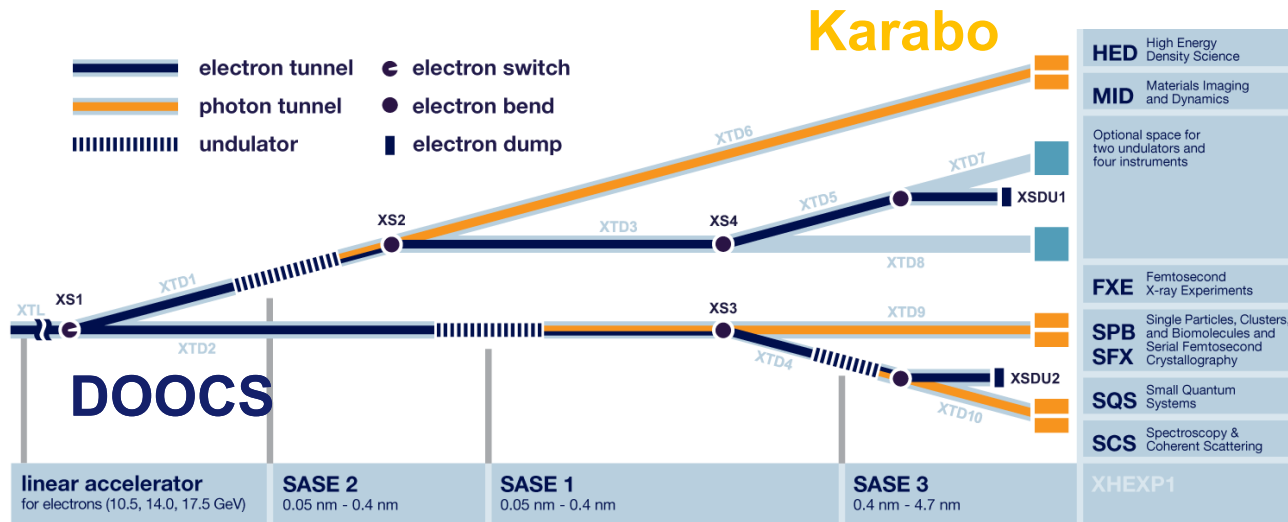


The European XFEL



- * 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**.

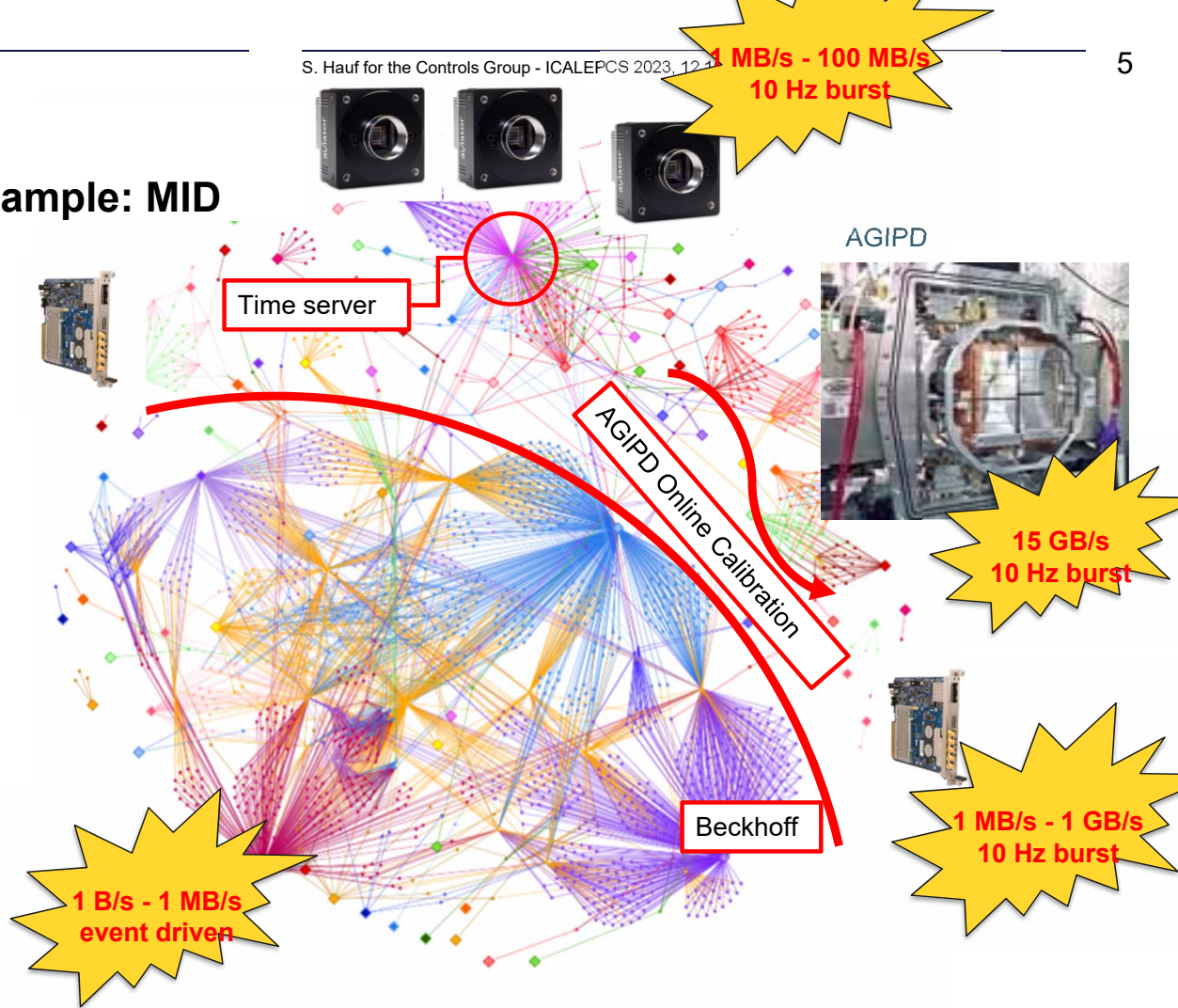
Beamline layout and experiment stations



- * 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)

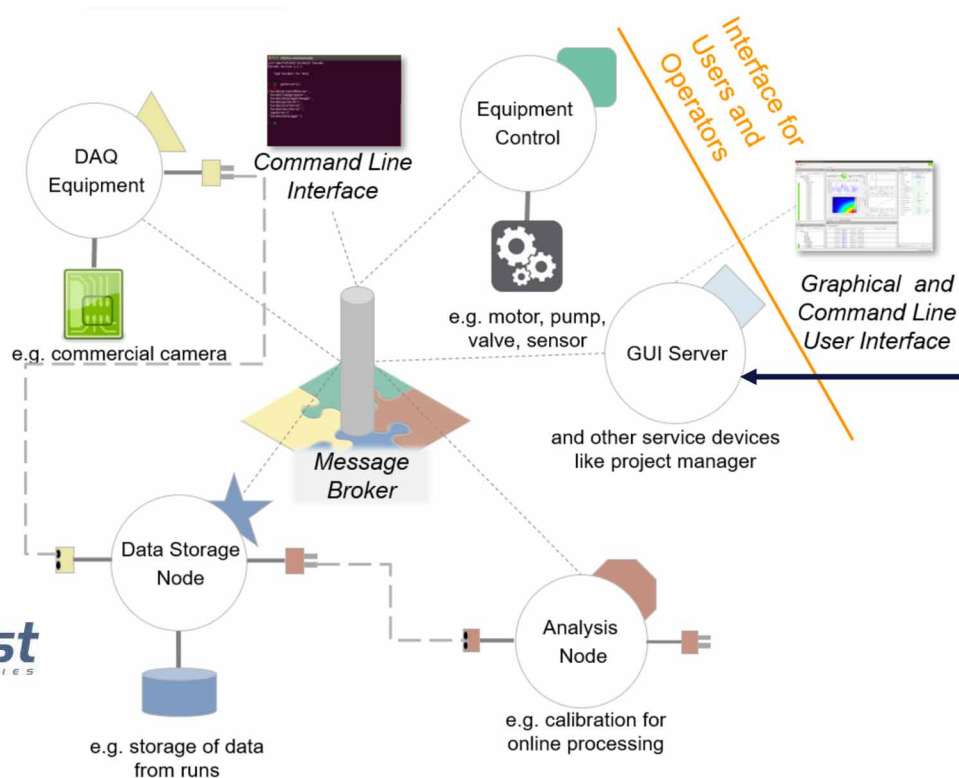
Karabo connects it all – Example: MID

- * Key components 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



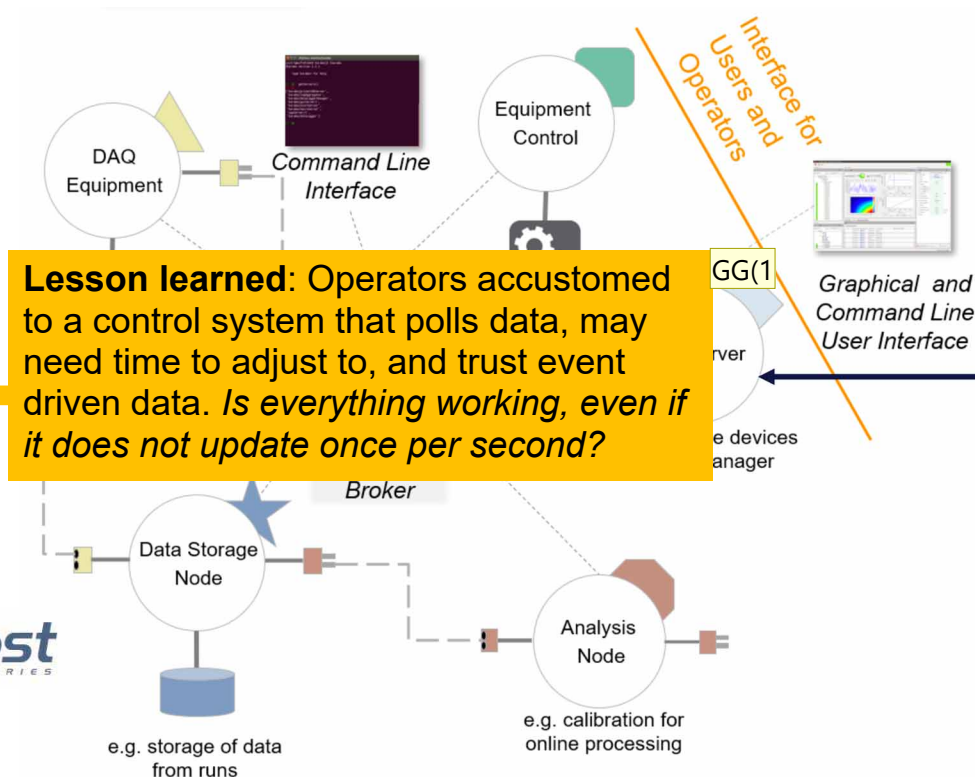
Karabo - Architecture

- * 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



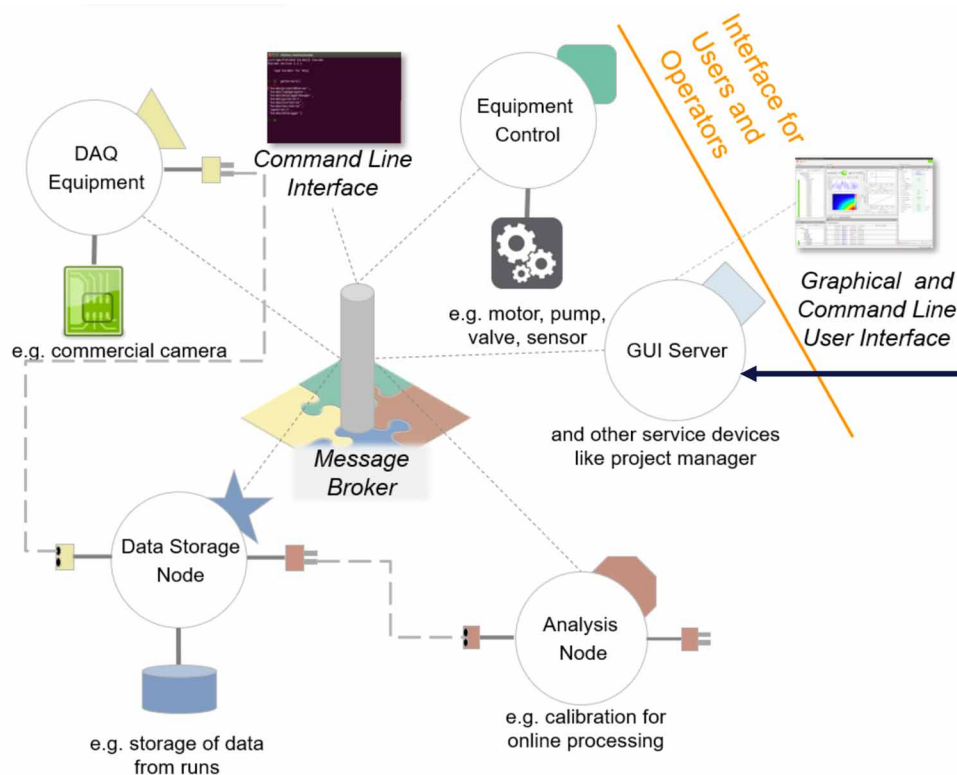
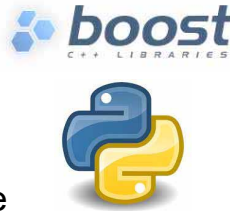
Karabo - Architecture

- * 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



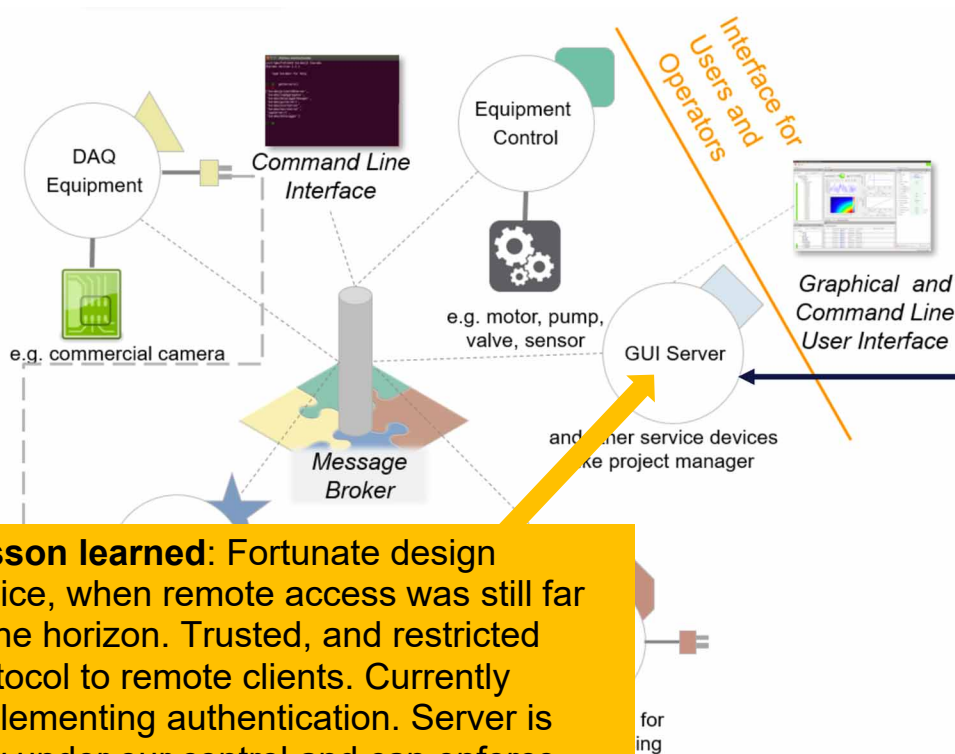
Karabo - Architecture

- * 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

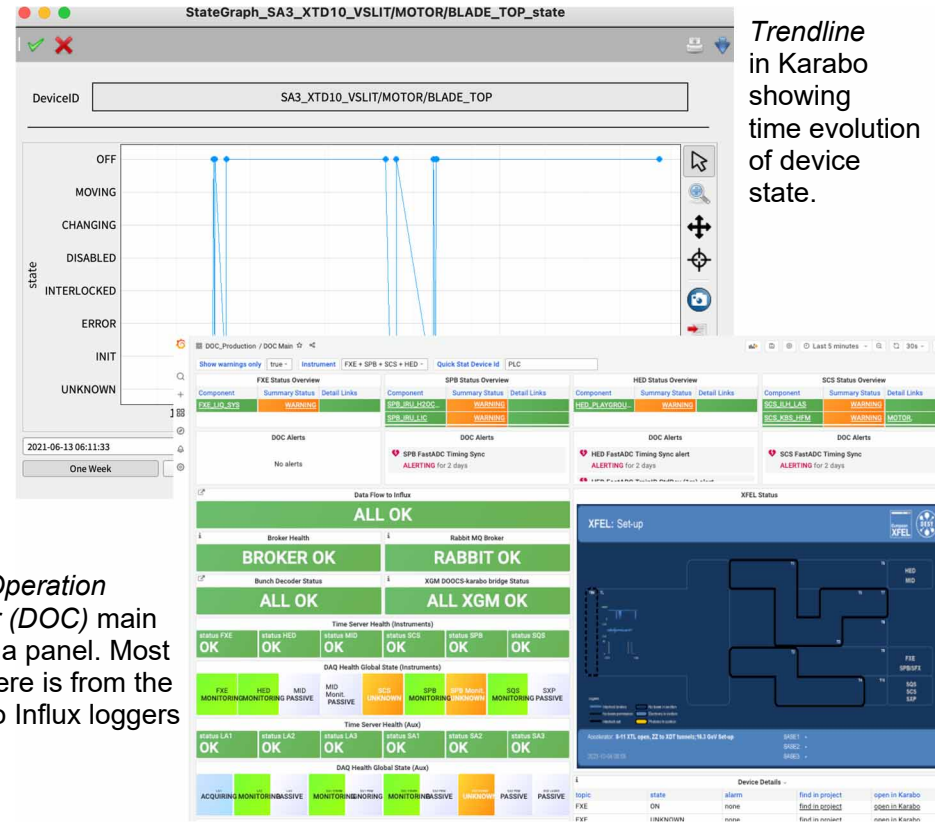


Lesson learned: Fortunate design choice, when remote access was still far at the horizon. Trusted, and restricted protocol to remote clients. Currently implementing authentication. Server is fully under our control and can enforce e.g. read-only access.

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 continuous for slow (broker) data
 - ▶ It is done by default
 - ▶ For all devices
 - ▶ Internal data product for maintenance
 - * Data Acquisition is „run“ based
 - ▶ Explicitly 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.

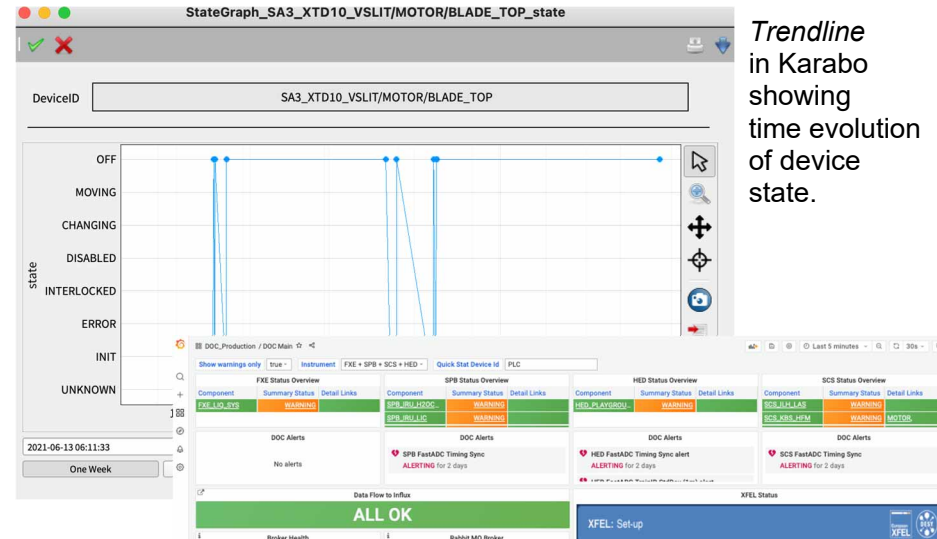
*Data Operation
Center (DOC) main
Grafana panel. Most
data here is from the
Karabo Influx loggers*

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 continuous for slow (broker) data
 - ▶ It is done by default
 - ▶ For all devices
 - ▶ Internal data product for maintenance
 - * Data Acquisition is „run“ based
 - ▶ Explicitly 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 
<ul style="list-style-type: none">• Event driven• Asynchronous• Self-descriptive• Common, hierarchical data container supporting attributes on leafs: Karabo Hash<ul style="list-style-type: none">• Binary and XML serialization• Extensible: core + “Devices“	<ul style="list-style-type: none">• C++14 and Boost• Smart pointers• Template-heavy• Boost.asio• Eventloop based• Devices are threads on a single server• Aimed at high-performance devices	<ul style="list-style-type: none">• Exposes C++ API via Boost.Python• Devices are separate processes• Was aimed at p2p heavy devices which e.g. need numpy• Not always pythonic	<ul style="list-style-type: none">• 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

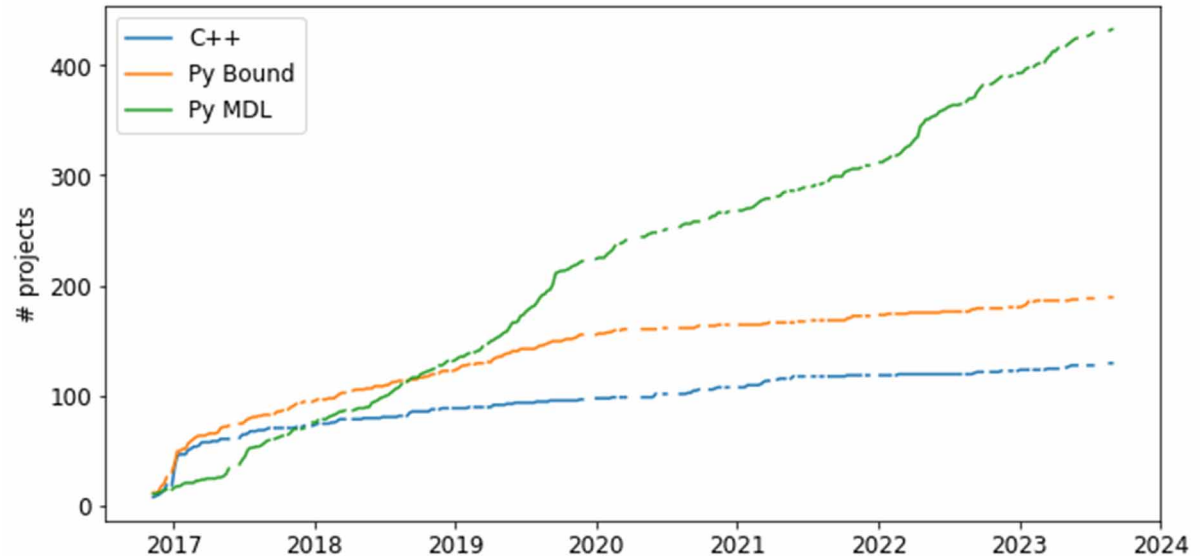
```
...:     futures.append(device.on())
...:
...:     await gather(*futures)
...:     await waitUntil(lambda: all(dev.state == State.ON for dev in devices))
```

Instantiation

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

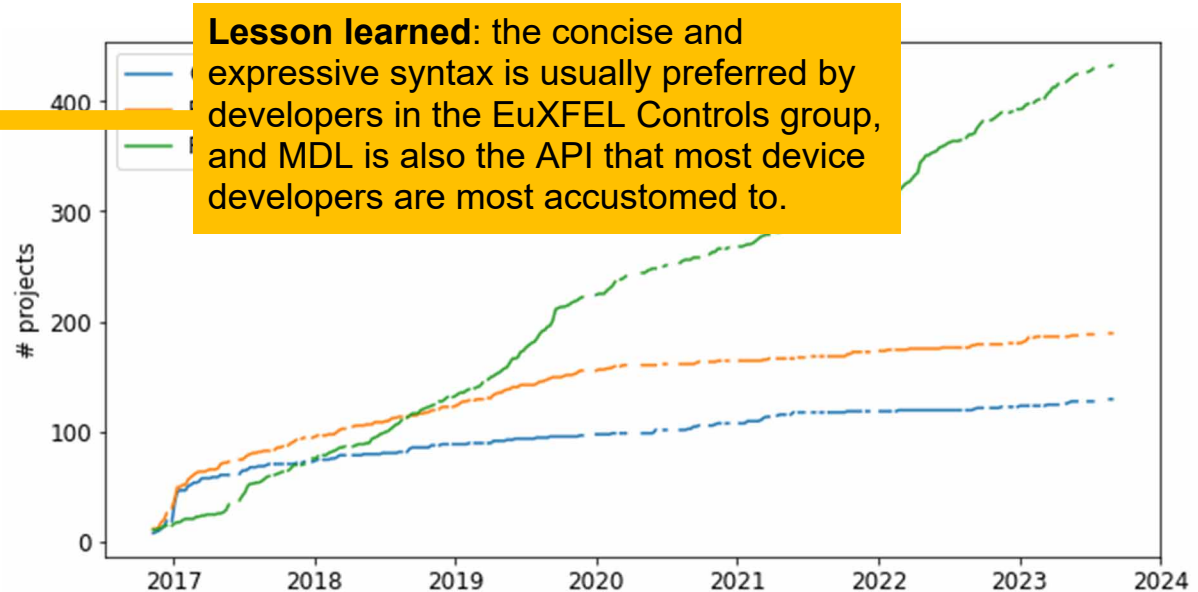
The Karabo Ecosystem – Usage of the three APIs

- * 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.



Karabo - The Karabo GUI

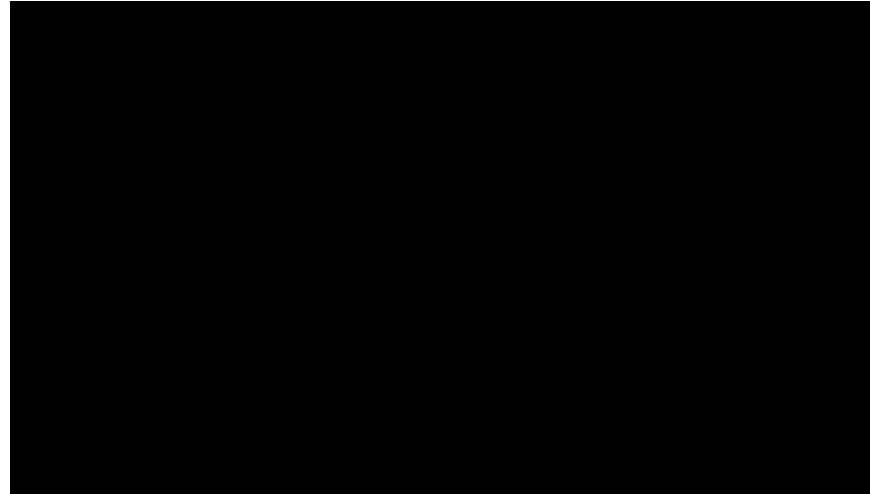
- * Separate Python Package, well matched to the framework
- * PyQt5
- * Connects to Karabo via the GUI-server (tcp, p2p)
- * Extensible via „gui-extensions“
- * Distinguishing features:
 - * GUI scene builder
 - * Projects to logically group devices, scenes and macros

The screenshot displays the Karabo GUI interface, which is divided into several panels:

- SA2_MAIN Panel:** Shows the SASE2 Photon Beam Transport system. It includes a control panel with parameters: Photon Energy (8679.1 eV), Number Of Bunches (50), Repetition rate (1.1284722 Hz), Bunch spacing (0.8861539 us), and Beam Trajectory (MID). Below this, a schematic diagram of the beam transport line is shown, featuring components like DOOCS, SR Imager, XGM, Solid Attenuator, CRL1, FEL Imager, pre absorber, Shutter XS2, and M1. A status bar at the bottom indicates 'Transmission 1.000e+00' and '20 mm aperture'.
- HED_LAS_COM/CAM/CAM2 Panel:** Displays a live camera scene. The scene shows a circular structure, likely a synchrotron ring, with a color scale on the right ranging from 0 to 4000. The axes are labeled 'X-axis (pixels)' and 'Y-axis (pixels)'. The status bar indicates 'ACQUIRING', 'Frame Rate 9.988271', and 'Mean Latency 131.18315'.
- Right Panel:** A device list table with columns for Alias, Device, and Axis/Source. It lists various motors and their configurations. Below the list, there are control buttons for 'TS_TX_SSTY' and 'AGIPD', and a table for 'Scan Type' and 'Control Acquisition Time'.

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**

SASE2 Photon Beam Transport

Photon Energy: 8679.1 eV
 Number Of Bunches: 50
 Repetition rate (MHz): 1.1284722
 Bunch spacing (us): 0.8861539
 Beam Trajectory: MID

XTD1 Tunnel

XTD6 Tunnel

HED_LAS_COM/CAM/CAM2

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.

Alias	Device	Axis/Source
ALIAS	DEVICE_ID	default
LO...	HW_MID_EXP...	default
Xra...	MID_EXP_UPP/...	default
Xra...	MID_EXP_UPP/...	default
TS_TX	MID_EXP_UPP/...	default
SSRY	MID_EXP_SAM...	default
The...	MID_EXP_UPP/...	default
Sa...	MID_EXP_UPP/...	default
Sa...	MID_EXP_UPP/...	default
SSY	MID_EXP_SAM...	default
fssy	MID_SAE_FSS...	default
fssx	MID_SAE_FSS...	default
LOTZ	MID_EXP_SAM...	default
T6_T2	MID_EXP_UPP/...	default
BM...	MID_OPT_SDL...	default
CC1...	MID_OPT_SDL...	default
CC2...	MID_OPT_SDL...	default
RO...	MID_AUXT2_A...	default
RO...	MID_AUXT2_A...	default
RO...	MID_AUXT2_A...	default
EN...	MID_XTD1_UN...	default
SS...	HW_MID_EXP...	default
SS...	HW_MID_EXP...	default
SS...	HW_MID_EXP...	default
SS...	HW_MID_EXP...	default
TX1	MID_EXP_DPS/...	default
ACC...	MID_OPT_MO...	default
CC2...	MID_OPT_SDL...	default
BS1...	MID_OPT_SDL...	default

TS_TX_SSTY AGIP

TS_TX_SSTY AGIPD

Scan Type: dmesh dmesh

Start Positions: -0.05,-0.5 -0.05,-0.5

Stop Positions: 0.05,0.5 0.05,0.5

Steps: 3,1 3,1

Use DAQ: Acquisition Time

Control Acquisition Time:

Average Values:

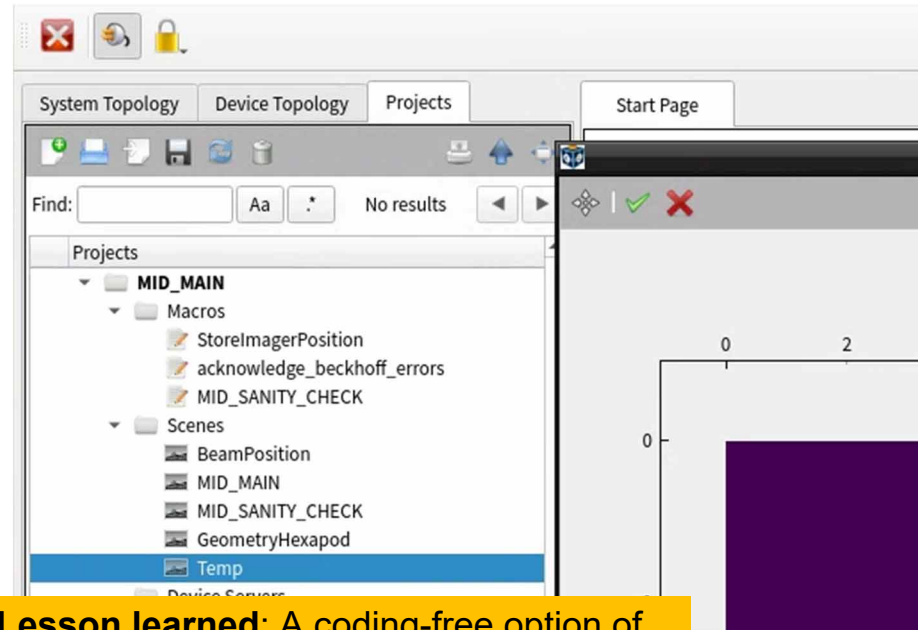
Control Velocity:

Bidirectional:

Comment:

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 <ul style="list-style-type: none"> • Data Logging • Data Acquisition • Gui Server • Alarms & Notification • Recovery Portal • Motor Configurator • Scan Tool 	Motion <ul style="list-style-type: none"> • Beckhoff MC2* • Smaract • Hexapod • NanoCube • ... 	Commercial Cameras <ul style="list-style-type: none"> • Basler via Aravis • Genicam • GreatEyes • PI MTE3 • Andor Zylar, ... • Shimadzu HPvx 	X-ray Detectors <ul style="list-style-type: none"> • Jungfrau • Timepix3 • Gotthard • ... • Epix • AGIPD • LPD • DSSC 	Scopes and ...meters <ul style="list-style-type: none"> • OceanOptics • GENTEC • Tetronix • LeCroy • MCS Beam Stab. • ...
Digitizers <ul style="list-style-type: none"> • SP devices ADQ 412 • SP devices ADQ 7 • SP devices ADQ 14 • FastADC 	Power Supplies <ul style="list-style-type: none"> • Wiener MPOD • Keithley • Agilent 	Vacuum Components* <ul style="list-style-type: none"> • Adixon • Pfeiffer • Infinicon • Agilent 	Chillers & Thermo Contr.* <ul style="list-style-type: none"> • Huber • K2 • Julabo • Keithley • Lakeshore • ... 	Bridging <ul style="list-style-type: none"> • SCPI • DOOCS • EPICS (in progress) • Tango (in progress)

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	Motion	Commercial Cameras	X-ray Detectors	Scopes and ...meters
<ul style="list-style-type: none"> Data Logging Data Acquisition Gui Server Alarms & Notification Recovery Portal 	<p>MO2AO03 A. Garcia-Tabares: The Solid Sample Scanning Workflow at the European XFEL</p> <p>THPDP024 F. Sohn: Automatic Configuration of Motors at the European XFEL</p> <ul style="list-style-type: none"> NanoCube ... 	<ul style="list-style-type: none"> PI MTE3 Andor Zylar, ... Shimadzu HPvx 	<ul style="list-style-type: none"> Jungfrau Gotthard Epix AGIPD LPD DSSC 	<ul style="list-style-type: none"> OceanOptics GENTEC Tetronix LeCroy MCS Beam Stab. ...
<p>WE1BCO02 J. Malka: Data Management Infrastructure for European XFEL</p>				
Digitizers	Power Supplies	Vacuum Components*	Chillers & Thermo Contr.*	Bridging
<ul style="list-style-type: none"> SP devices ADQ 412 SP devices ADQ 7 SP devices ADQ 14 FastADC 	<ul style="list-style-type: none"> Wiener MPOD Keithley Agilent 	<ul style="list-style-type: none"> Adixon <p>THPDP022 B. Rio: Adaptable Control System for the Photon Beamlines at European XFEL: Integrating New Devices and Technologies for Advanced Research</p>	<ul style="list-style-type: none"> Huber 	<ul style="list-style-type: none"> SCPI DOOCS EPICS (in progress) Tango (in progress)
<p>MO4AO06 B. Fernandes: Overview and Outlook of Fpga Based Hardware Solutions for Data Synchronization, Acquisition and Processing at the Euxfel</p>		<p>TUPDP033 M. Smith: Applying Model Predictive Control to Regulate Thermal Stability for a Hard X-ray Monochromator Using the Karabo Software Control Framework</p>		

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

Move to RabbitMQ and AMQP



RDMA – 80% IB link speed 

ZULIP

Elog Integration



MDL on PyPi

WebUI

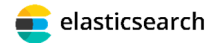


<https://github.com/European-XFEL/Karabo>



Authenticating & Authorizing Web API

{REST:API}*



ELK Stack for System Logs

GUI side Authentication and Authorization



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.



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 accelerator.

Download 19.5 MB. 5256 x 29321



WebUI



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

Web API



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>

Add AI generated documentation

Overview 4 Commits 3 Pipelines 3 Changes 23

The screenshot shows a code diff viewer for the file `src/devices/DataAggregator/ApplicationMonitor.hh`. The diff compares the `develop` branch with the `latest version`. The code is shown in a dark theme with green highlights for added lines. The diff shows the following changes:

```
31 | 31 |                                     long unsigned int cpu_total_time;
32 | 32 |                                     };
33 | 33 |
34 | + 34 |                                     /**
35 | + 35 |                                     * @class ApplicationMonitor
36 | + 36 |                                     *
37 | + 37 |                                     * @brief Monitors the status of an application and publishes
38 | + 38 |                                     *
39 | + 39 |                                     * This class is responsible for monitoring the status of
40 | + 40 |                                     * calculates CPU usage, reads process and CPU statistics,
41 | + 41 |                                     * ability to publish the content of the application status
42 | + 42 |                                     *
43 | + 43 |                                     * @note This class inherits from the Runnable class.
44 | + 44 |                                     */
34 | 45 |                                     class ApplicationMonitor : public pPlayer, Runnable {
35 | 46 |
36 | 47 |                                     public:
37 | 48 |                                     // Add reflection and version information to this class
38 | 49 |                                     KARABO_CLASSINFO(ApplicationMonitor, "DataAggregator::
39 | 50 |
51 | + 51 |                                     /**
52 | + 52 |                                     * @brief Constructor for the ApplicationMonitor class
53 | + 53 |                                     *
54 | + 54 |                                     * This constructor initializes an instance of the App
55 | + 55 |                                     *
56 | + 56 |                                     * @param config: a Hash containing the configuration
57 | + 57 |                                     *
58 | + 58 |                                     * @note The keys in the Hash are not specified in the
59 | + 59 |                                     */
48 | 68 |                                     ApplicationMonitor(const karabo::util::Hash& config);
```


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:



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>