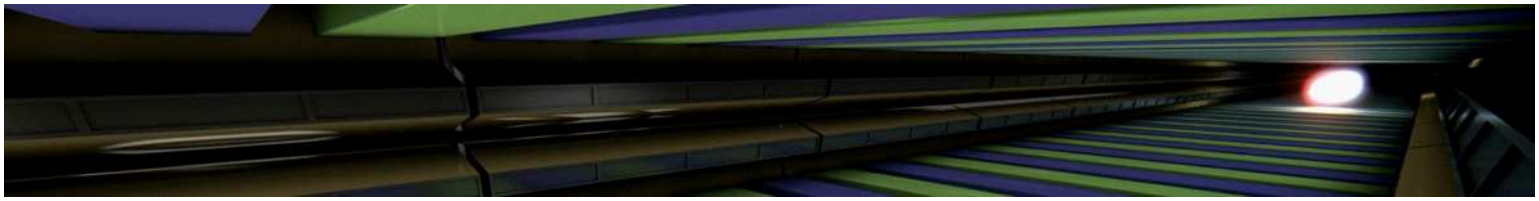


Karabo

Integrating Control, Data Acquisition, Data Management, and Data Analysis Tasks into a Single Software Framework

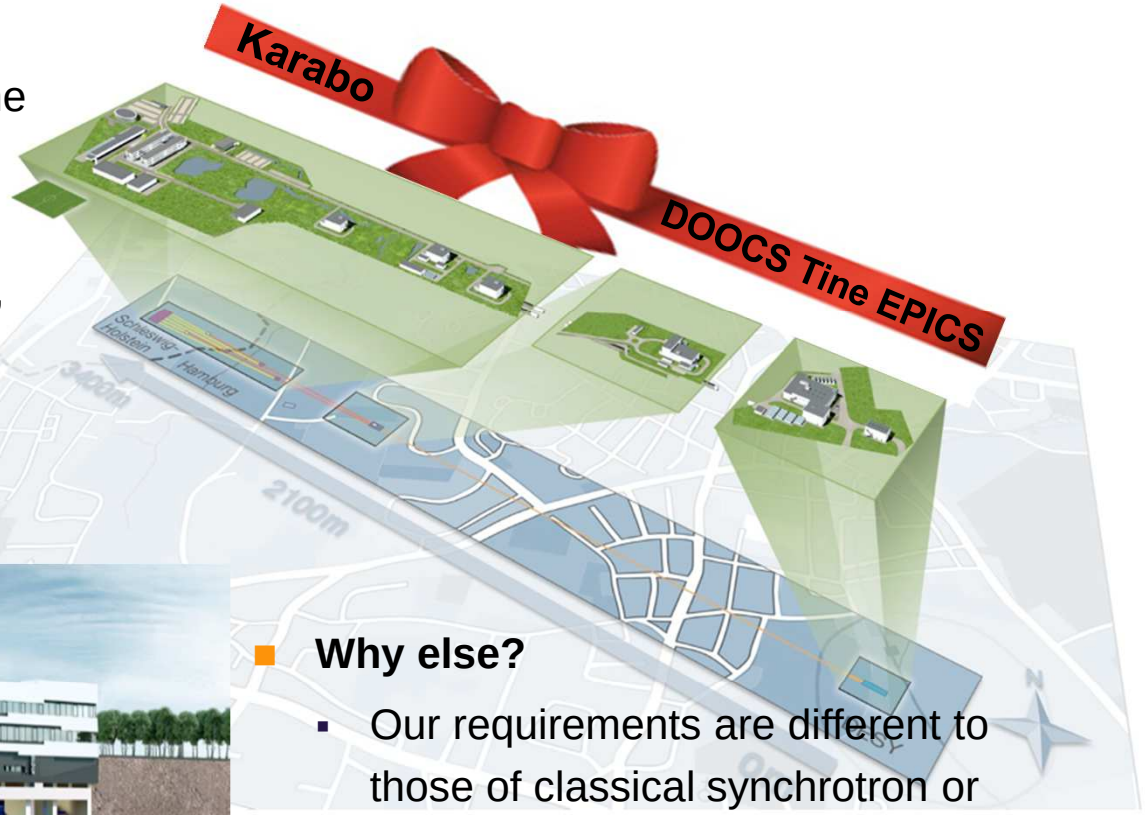
Burkhard Heisen
Control and Analysis Software Group – European XFEL GmbH

European XFEL User's Meeting
Hamburg, January 28, 2015

European XFEL

Karabo will be used at the photon beamlines

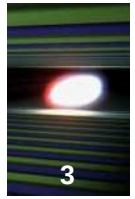
- **Why?**
  - One software framework, one user interface, one programming interface
  - Less learning, less teaching, less maintenance, less changes, less conversions, more freedom



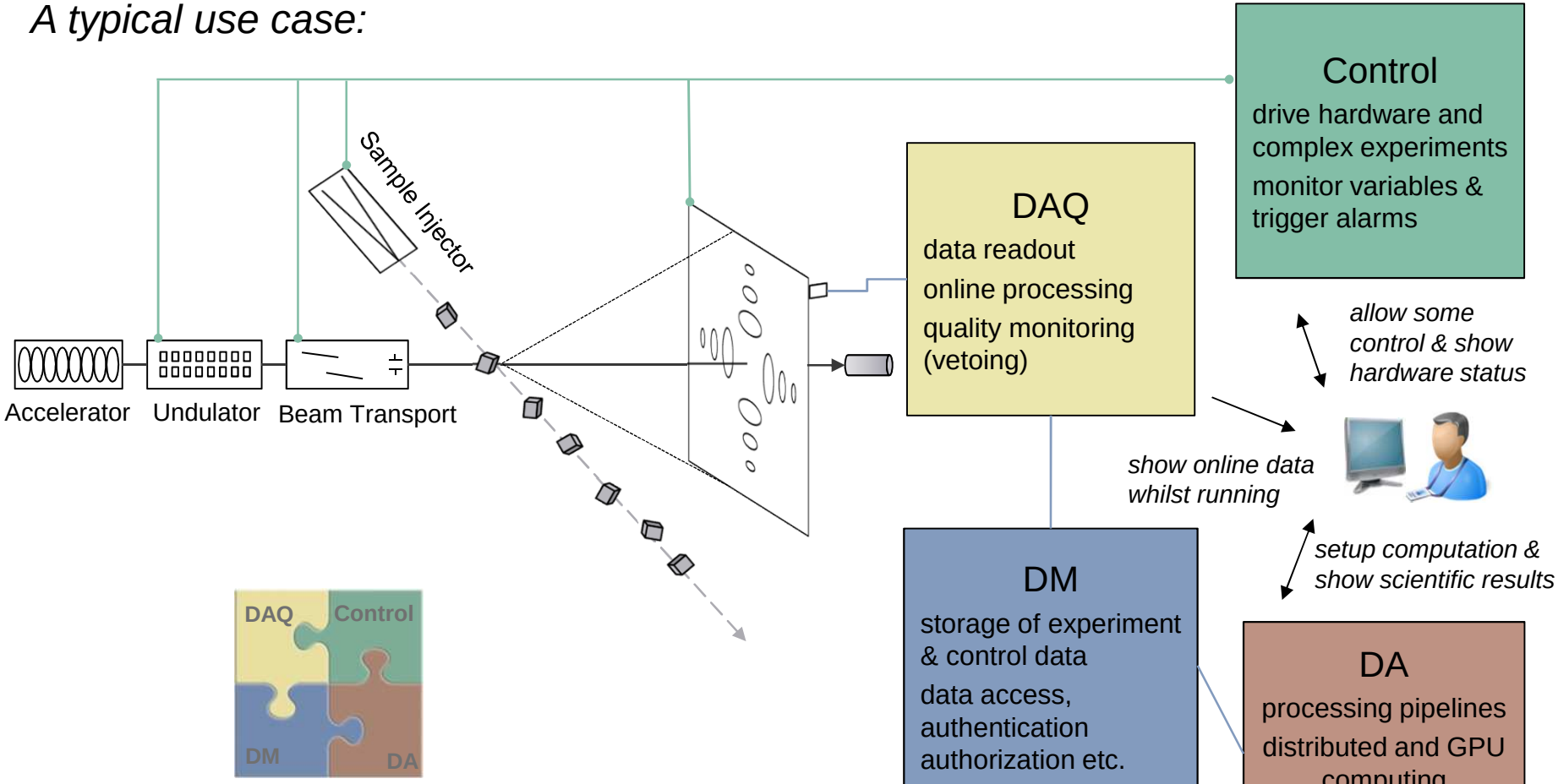
- **Why else?**
  - Our requirements are different to those of classical synchrotron or high energy physics experiments
  - We invest a lot of money for better science, a small fraction is for better software

European XFEL

The vision – From a user perspective

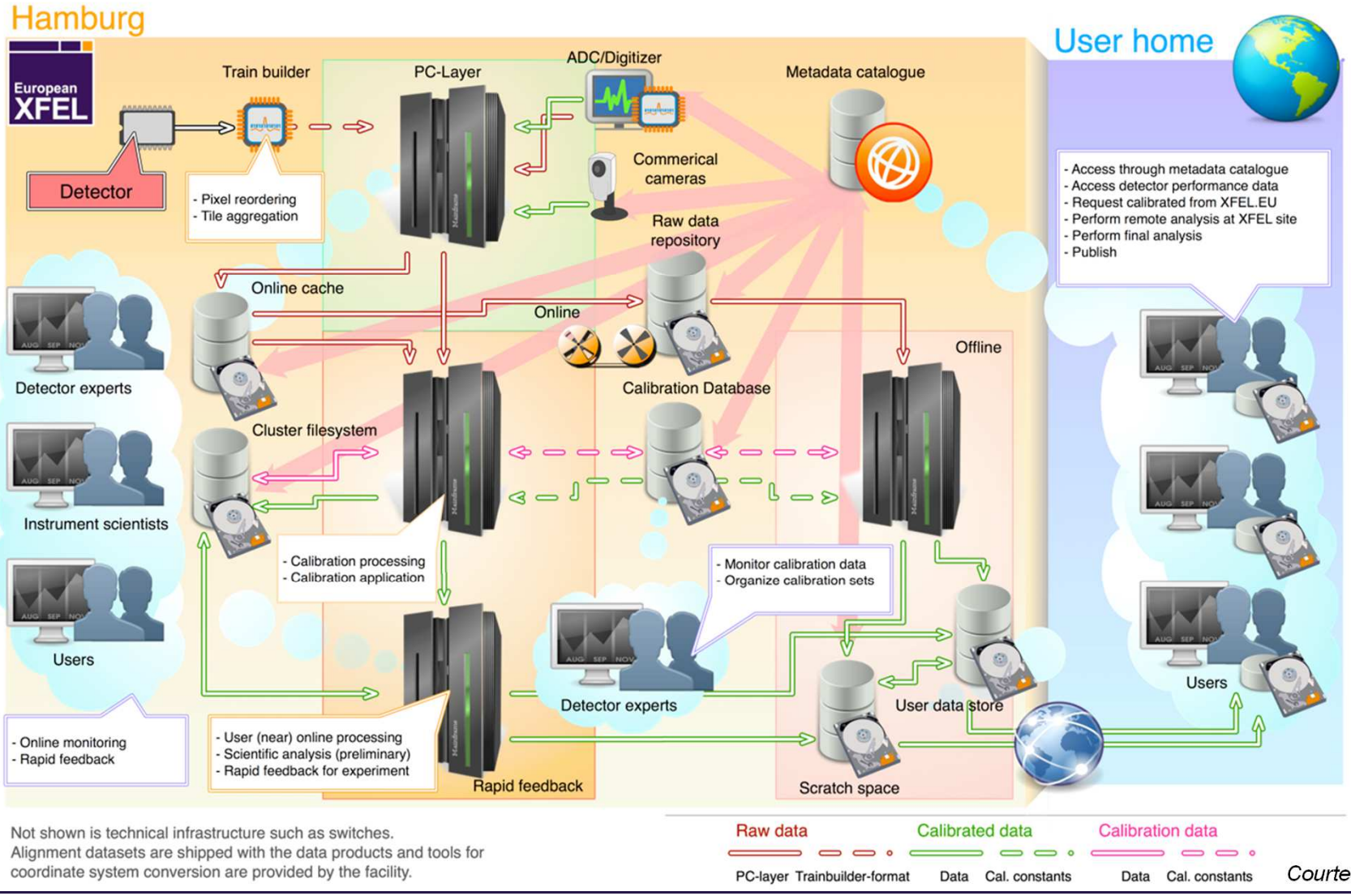


A typical use case:



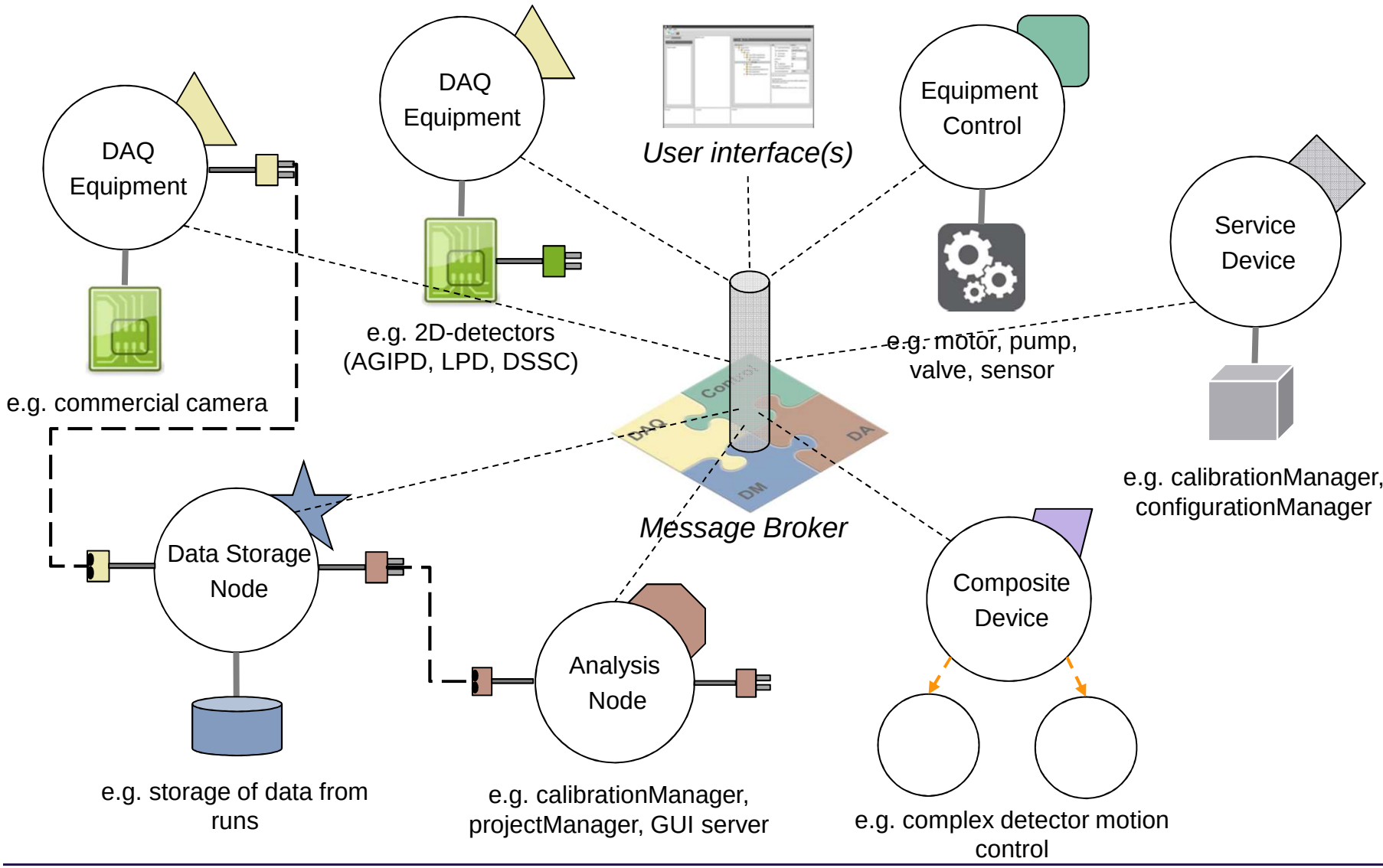
Tight integration of applications

The vision – From a data perspective



Courtesy of S. Hauf

Karabo is a “device” based system



European XFEL

Device – As controllable object

- Device classes can be loaded at runtime (**plugin technology**)
- Can be written in **C++** or **Python**
- Devices completely **describe themselves**. Allows automatic GUI creation
- **Runtime-extension** of properties, commands and attributes is possible
- **Hierarchical groupings** of parameters are possible
- Any property setting or command execution can be restricted to a set of **allowed states**
- The GUI reflects state limitations by **enabling/disabling buttons** and **properties** dynamically

```
class Conveyor(Device):

    @staticmethod
    def expectedParameters(expected):
        (
        SLOT_ELEMENT(expected).key("start")
         .displayedName("Start")
         .description("Instructs device to go to started state")
         .allowedStates("Stopped")
         .commit()
         ,
        SLOT_ELEMENT(expected).key("stop")
         .displayedName("Stop")
         .description("Instructs device to go to stopped state")
         .allowedStates("Started")
         .commit()
         ,
        DOUBLE_ELEMENT(expected).key("targetSpeed")
         .displayedName("Target Conveyor Speed")
         .description("Configures the speed of the conveyor belt")
         .unit(Unit.METER_PER_SECOND)
         .assignmentOptional().defaultValue(0.8)
         .reconfigurable()
         .commit()
         ,
        DOUBLE_ELEMENT(expected).key("currentSpeed")
         .displayedName("Current Conveyor Speed")
         .description("Shows the current speed of the conveyor")
         .readOnly()
         .commit()
         ,
```

Command

Attribute

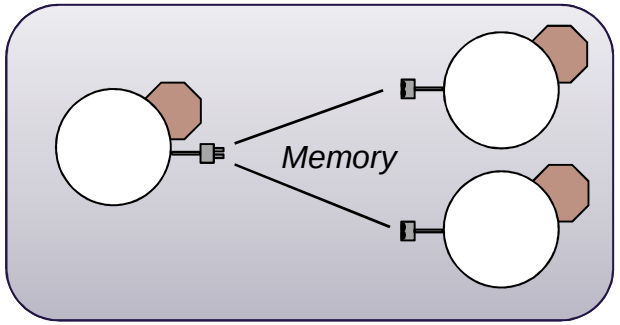
Property

auto generated

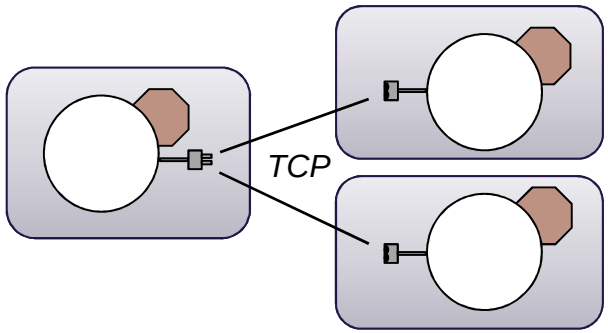
European XFEL

Device – As pipeline module

- **Devices can act as modules of a streaming data pipeline**
  - Configurable generic input/output channels on devices
  - Developers just need to code the **onData** and (optionally) the **onEndOfStream** method
  - **IO** system is **decoupled** from processing system (process whilst transferring data)
  - Input channels configure whether they **share** the sent data amongst all input channels or whether they receive a **copy** of each data token
  - Broker-based communication **transparently** establishes a point-to-point connection
  - Any pipeline device has full **access to** the live **control-system** (feedbacks easily possible)



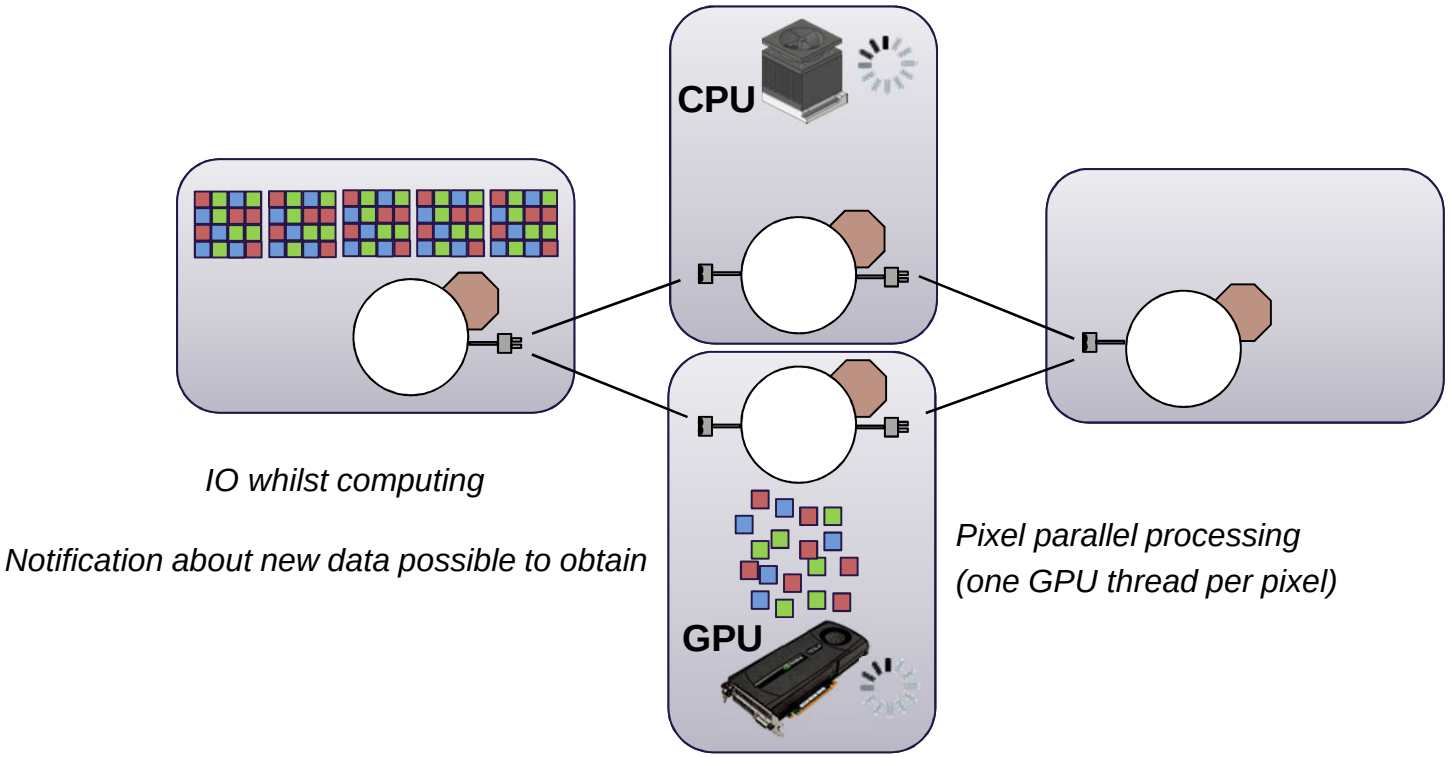
Multi-threaded processing



Distributed processing

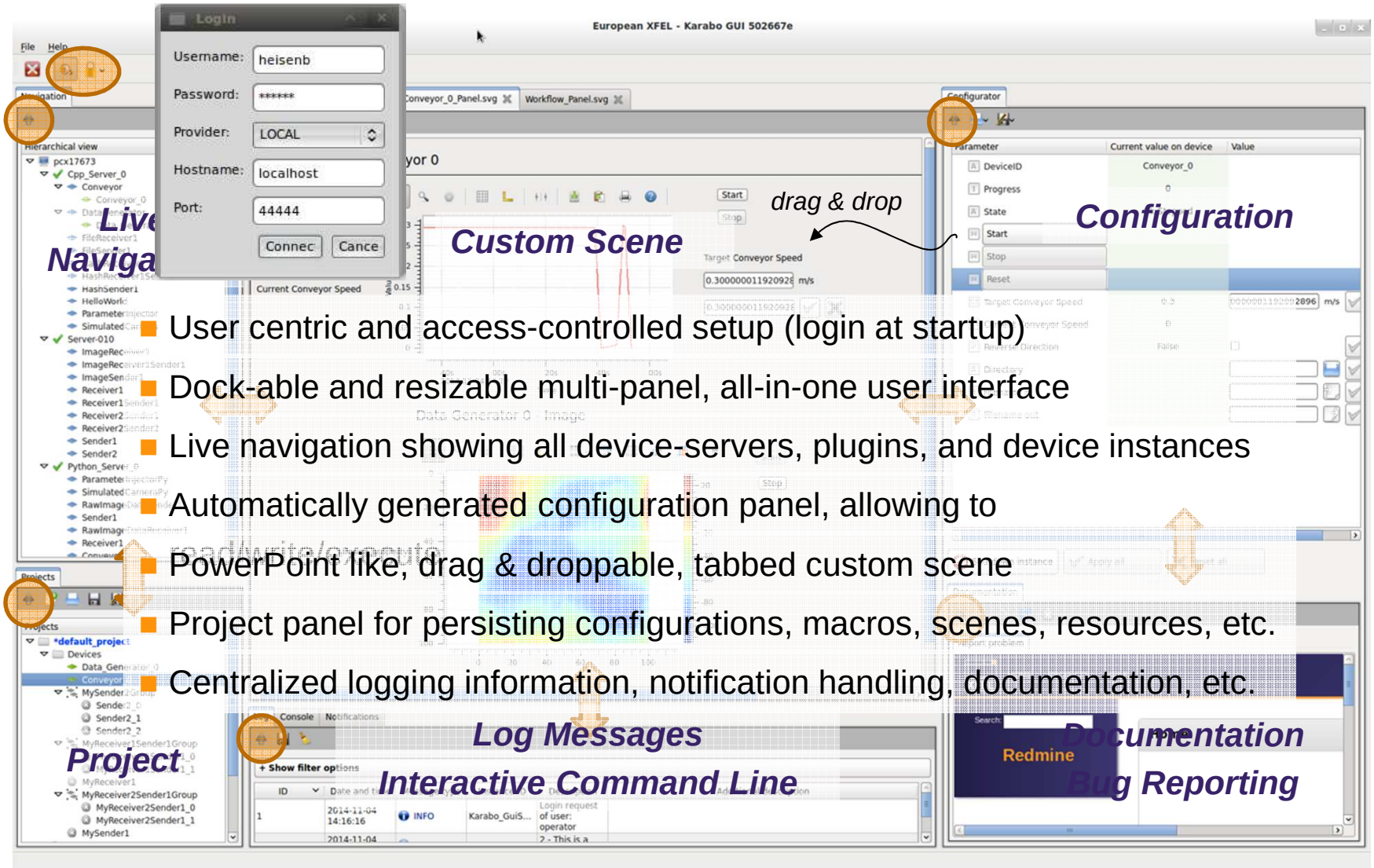
European XFEL

Pipelines support load balancing



- Once resources are available, input channels **request new data** from connected output channels
- Configurable output channel behavior in case no input currently available: **error, queue, wait, drop**

Graphical interface - Overview

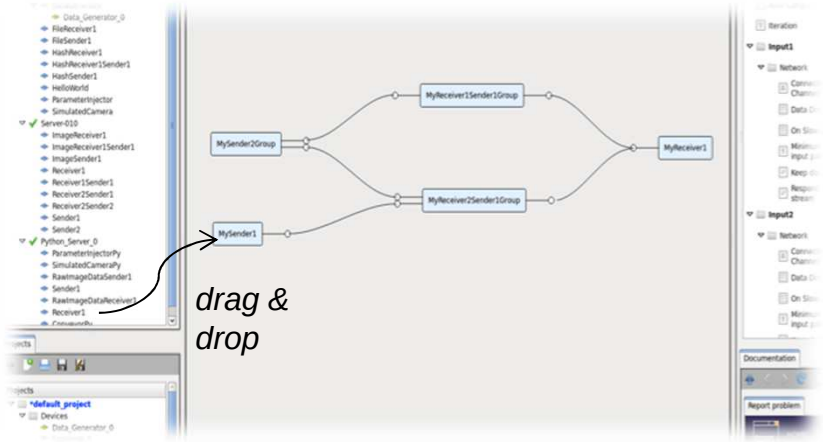


- User centric and access-controlled setup (login at startup)
- Dock-able and resizable multi-panel, all-in-one user interface
- Live navigation showing all device-servers, plugins, and device instances
- Automatically generated configuration panel, allowing to
- PowerPoint like, drag & droppable, tabbed custom scene
- Project panel for persisting configurations, macros, scenes, resources, etc.
- Centralized logging information, notification handling, documentation, etc.

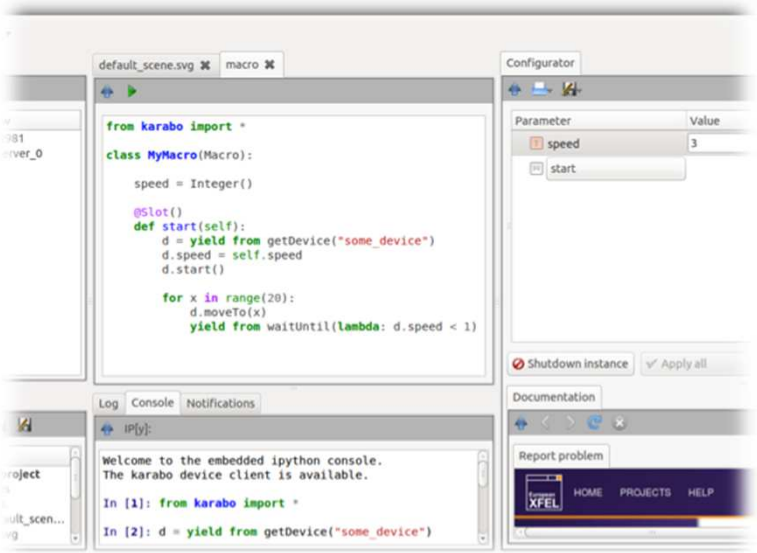
European XFEL

Graphical interface – Pipelines and Macros

- Pipeline configuration
  - Whole devices can be **dragged** (from left side) as pipeline nodes
  - Dragging individual parameters from right is still possible (e.g. control parameters)
  - Devices can be **grouped** and edited as group (connections and configurations)
- Macro editing
  - Editing and execution from within GUI or using regular terminal
  - Macro parameters and functions **integrate** automatically **into configuration** panel
  - Macro API can be **interactively executed** in embedded IPython interpreter
  - Macro writing is something we hope our **users can** easily **do** themselves



Courtesy of K. Weger



Courtesy of M. Teichmann

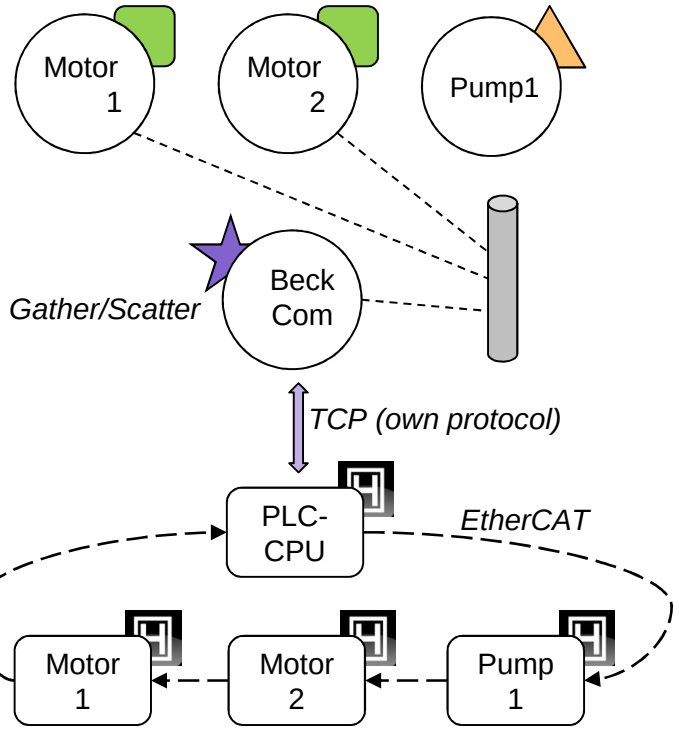
European XFEL

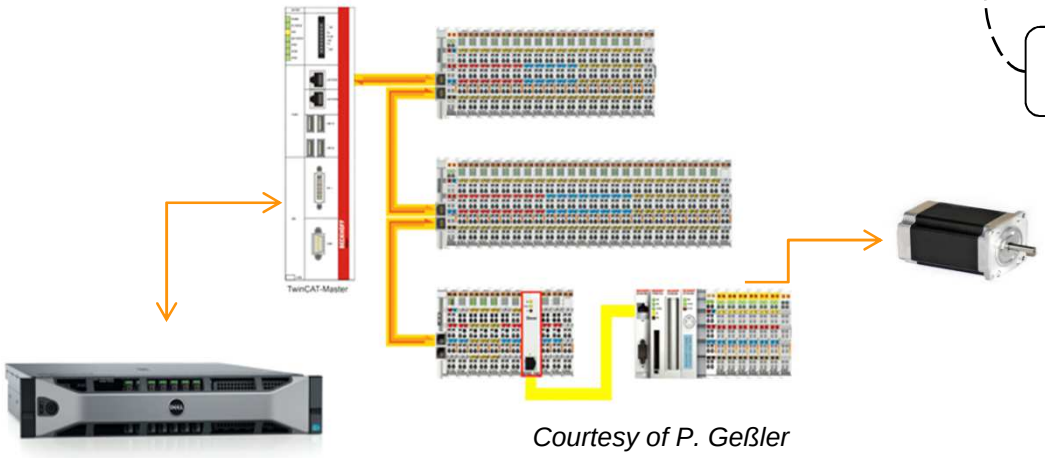
Cameras, power supplies, heaters…

- Supported standards
  - **GenICam** (generic interface for cameras) - industry standard
  - **Lima** (library for image acquisition) – ESRF project
  - **SPCI** (Standard Commands for Programmable Instruments)
- Cameras
  - Basler (all GigE), Photonic Science (sCMOS) ✔
  - IMPERX (Bobcat B4020), Andor (Neo) ✔
  - AVT (all Prosilica), Axis (all) ✔
- Other devices
  - FUG, Heinziger (power supplies), TEM (beam lock) ✔
  - Amphos (laser power stage) ✔
  - EnergyMax (laser sensors) ✔
  - Lakeshore (heater), STS (spectrometer) ✔

Motors, pumps, valves… and real-time control

- Karabo itself does not directly provide real-time processes/communications -> uses Beckhoff layer
- Beckhoff PLC-firmware blocks are mapped to Karabo devices and can be **auto-generated** from PLC self-description
- Integrated devices (now extending to different vendors):
  - Turbo-, ion-, and scroll-pumps ✔
  - Valves, gauges, temperature sensors ✔
  - Stepper-, piezo- and servo-motors ✔





Courtesy of P. Geßler

BECKHOFF

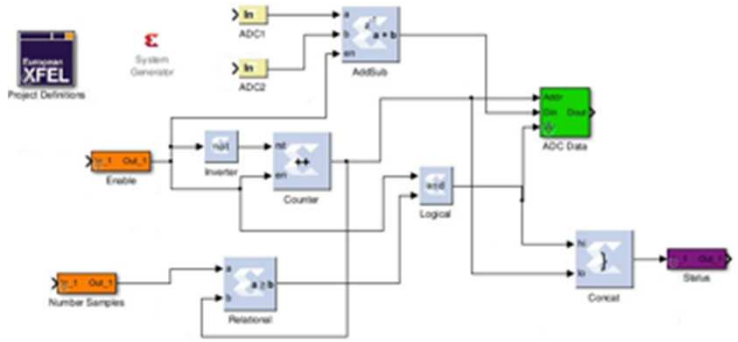
Digitizers, ADCs, Timing interfaces…

- Digitizers (used in XGMs, PESs) ✓
- Fast ADCs (analog to digital converters) ✓
- Timing interfaces (in progress) ✓
- Simulink/Matlab based FPGA programming framework with **Karabo integration** ✓

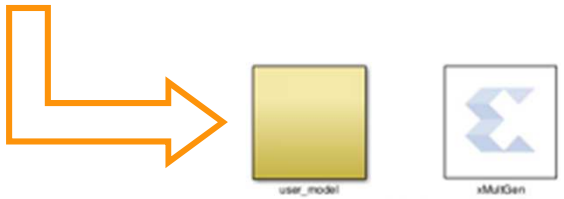
ADQ 412 (SP-Devices)

SIS8300 – ADC AMC (Struck Innovative Systems)

MicroTCA crate



XML FILE **User Registers and Memories Description** → **Software (Karabo)**



Final Design

Source Files **with script** → ISE DESIGN SUITE

Bus protocol logic generated automatically

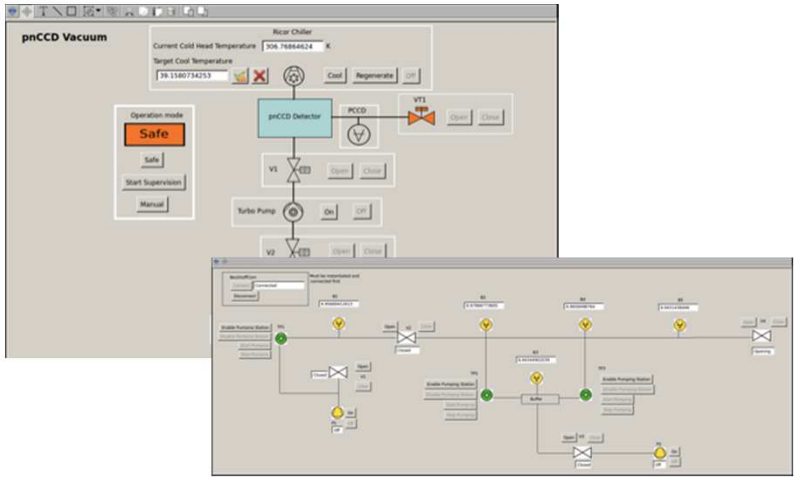
FPGA compiler/tool

Courtesy of P. Geßler

Karabo in use – Control systems

- Pump probe laser setup (laser group)
  - Runs burst mode according to XFEL (4.5 MHz intra burst)
  - Control of motors, cameras, MicroTCA based ADCs, image processing devices
  - Controls pulse arrival time in a closed loop (motor position correction depending on ADC signal)
- Vacuum setups
  - Control of e.g. differential pumping sections
  - Typically includes several valves, pumps, and PLC-interlocks for safety
  - Pressure monitoring and history trend line plotting

Courtesy of L. Wissmann



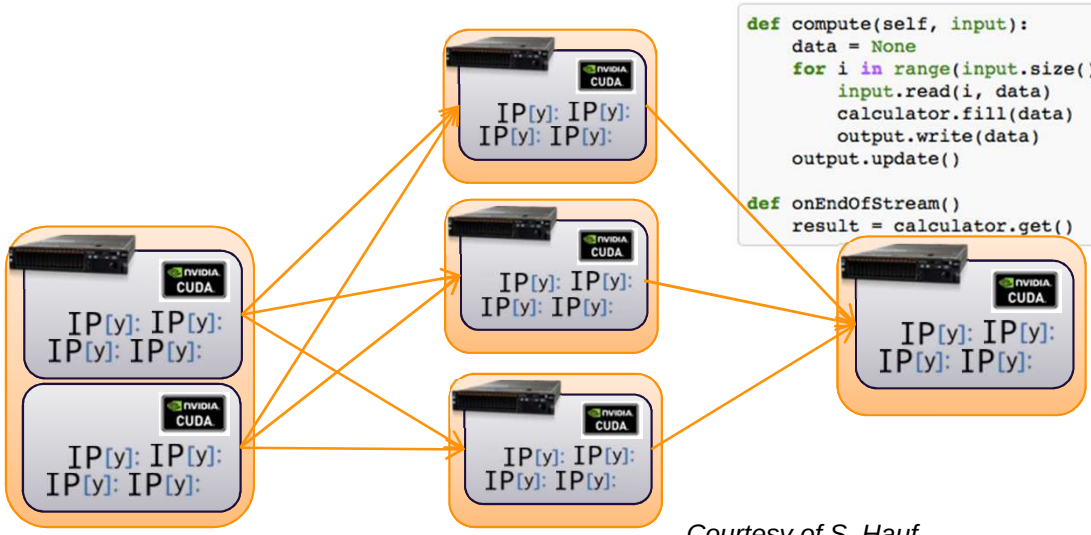
European XFEL

Karabo in use – DAQ and data management

- Large Pixel Detector (LPD) prototype testing
  - Detector control (Python based)
  - Data acquisition and storage (HDF5) (C++)
  - Online calibration processing, analysis and correction (Python and CUDA based)
  - Interaction with calibration database (Oracle)
  - Full integrated run control expected in summer

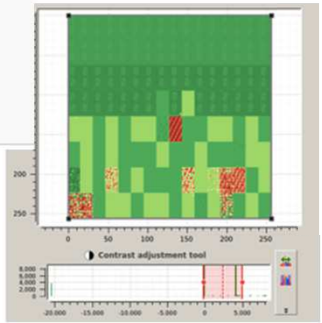
LPD quadrant prototype

Server setup necessary to harvest the data




```
def compute(self, input):
    data = None
    for i in range(input.size()):
        input.read(i, data)
        calculator.fill(data)
        output.write(data)
    output.update()

def onEndOfStream()
    result = calculator.get()
```

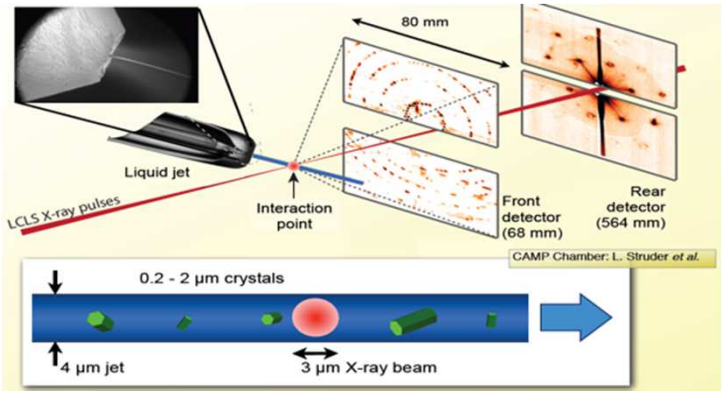


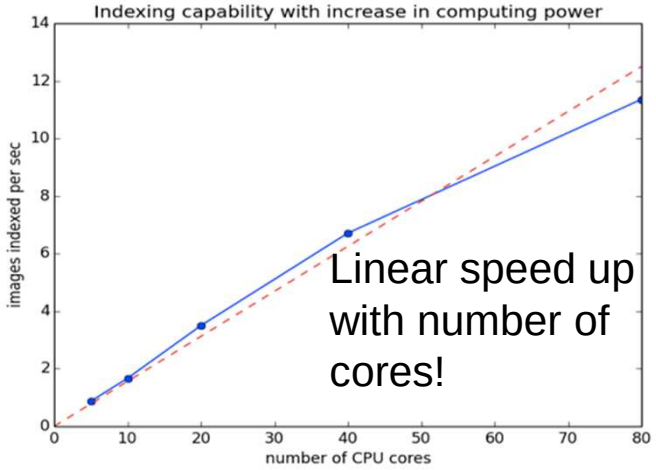
Visualization of a LPD quadrant in Karabo

Courtesy of S. Hauf

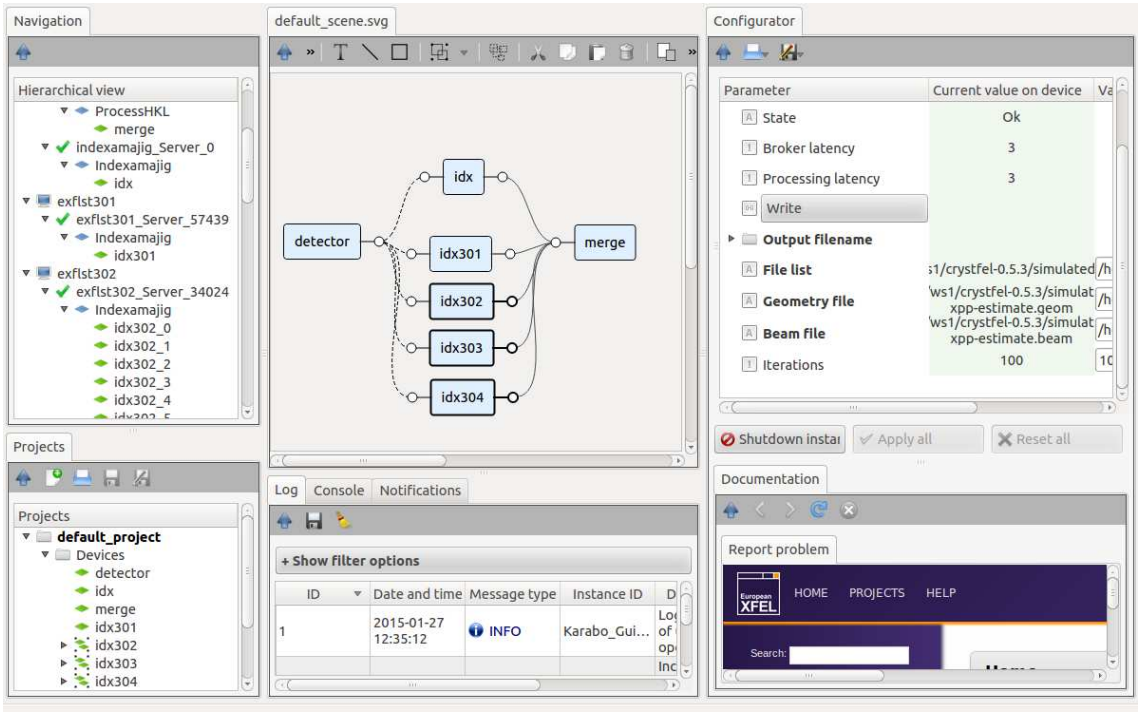
Karabo in use – Data Analysis

- Serial femtosecond crystallography
  - Aim is to build an **SFX data processing pipeline** to match the high repetition rate offered at the European XFEL
  - 5000 images/sec
  - @ 2.3 sec per image
  - 30% hit rate
  - 3500 cores needed





Courtesy of C. Yoon



European XFEL

Conclusions and Acknowledgements

- Karabo is unique in **seamlessly combining** instrument control, experiment configuration, streamed data analysis and visualization
- Its is **designed for being extended** easily and will be ready to execute your algorithms during the experiments
- Karabo proves capable to run complex installations even today
- Currently the focus is on supporting our in-house users and external collaborators -> access is on request
- Public release of the core system under the GNU GPL license will follow in 2015
- Stay tuned…

K. Weger
M. Teichmann
S. Esenov
A. Parenti
A. Beckmann
G. Giambartolomei
A. Silenzi
J. Szuba
M. Kumar
L. Maia
D. Boukhelef
S. Hauf
C. Yoon
A. Münnich
L. Wissmann
P. Geßler
K. Wrona
C. Youngman
… and many more

Thank you for your kind attention.

Contact details: burkhard.heisen@xfel.eu