Karabo

An integrated software framework combining control, data management, and scientific computing tasks

Burkhard Heisen
October 11, ICALEPCS 2013
Karabo: The European XFEL software framework

Karabo will be used at the European XFEL

- Data rate (10 GB/s/2d-detector)
- Bunch pattern (4.5 MHz pulses @ 10Hz)
Karabo: The European XFEL software framework

Functional requirements

A typical use case:

- **Control**: drive hardware and complex experiments, monitor variables & trigger alarms.
  - Allow some control & show hardware status.
  - Show online data whilst running.
  - Setup computation & show scientific results.

- **DAQ**: data readout, online processing, quality monitoring (vetoing).
  - DM: storage of experiment & control data, data access, authentication, authorization etc.
  - SC: processing pipelines, distributed and GPU computing, specific algorithms (e.g. reconstruction).

Tight integration of applications
What is Karabo? - Let's draw an analogy

Karabo devices

Karabo framework

Karabo device-server running devices
What is Karabo? - Let's draw an analogy

Karabo devices

Karabo framework

Karabo device-server running devices
What is Karabo? - Let's draw an analogy

Karabo devices

Google

Karabo central services (archive, database)

Android

Karabo framework

Android

Karabo device-server running devices
Karabo: The European XFEL software framework

Essential components

- HV
- Pump
- Calibrate1
- Calibrate2
- Simulate
- Diode
- Camera
- Load
- Store
- Disk Storage
- Message Broker
- Device-Server Application
- Device-Server Application
- Device Server
- Application
- Message Broker
- Device
- Sub Control
- Device Instance
- GUI
- Server
- CLI(s)
- GUI(s)
- RDB
- Logger
Karabo: The European XFEL software framework

Essential components

- **Device Instance**
  - HV
  - Pump

- **Device-Server Application**
  - Calibrate1
  - Calibrate2

- **Message Broker**
  - Logger
  - GUI Server
  - CLI(s)
  - GUI(s)

- **Clustering**
  - Device Sub Control
  - IP[y]:

- **Store**
  - Load
  - Disk Storage

Burkhard Heisen (WP76)
Devices are **controllable objects**
managed by a device server

Device classes can be loaded at runtime (**plugin technology**)

Can be written in **C++** or **Python**

Devices completely **describe themselves**. Allows automatic GUI creation and auto-completion in IPython

**Runtime-extension** of properties, commands and attributes is possible

**No need** for device developers to **validate** any parameters. This is internally done taking the expectedParameters as a white-list

Properties and commands can be nested, such that **hierarchical groupings** are possible

---

```cpp
Class: MotorDevice
static expectedParameters( Schema& s ) {
  FLOAT_ELEMENT(s).key("velocity")
    .description("Velocity of the motor")
    .unitSymbol("m/s")
    .assignmentOptional().defaultValue(0.3)
    .maxInc(10)
    .minInc(0.01)
    .reconfigurable()
    .allowedStates("Idle")
    .commit();

  INT32_ELEMENT(s).key("currentPosition")
    .description = "Current position of the motor"
    .readOnly()
    .warnLow(10)
    […]

  SLOT_ELEMENT(s).key("move")
    .description = "Will move motor to target position"
    .allowedStates("Idle")
    […]
}
// Constructor with initial configuration
MotorDevice( const Hash& config ) { […] }
```
- Any device uses a standardized way to express its possible program flow
  - The state machine calls back device functions (guard, onStateExit, action, onStateEntry)
  - The GUI is state-machine aware and enables/disables buttons proactively

```c
// Ok Machine
FSM_TABLE_BEGIN(OkTransitionTable)
// SrcState  Event  TgtState  Action  Guard
Row< Started, StopEvent, Stopped, StopAction, none >,
Row< Stopped, StartEvent, Started, StartAction, none >
FSM_TABLE_END
FSM_STATE_MACHINE(Ok, OkTransitionTable, Stopped, Self)

// Top Machine
FSM_TABLE_BEGIN(TransitionTable)
Row< Initialization, none, Ok, none, none >,
Row< Ok, ErrorFoundEvent, Error, ErrorFoundAction, none >,
Row< Error, ResetEvent, Ok, ResetAction, none >
FSM_TABLE_END
KARABO_FSM_STATE_MACHINE(StateMachine, TransitionTable, Initialization, Self)
```
Devices can act as modules of a scientific workflow system

- Configurable generic input/output channels on devices
- One channel is specific for one data structure (e.g. Hash, Image, File)
- New data structures can be “registered” and are immediately usable
- Developers just need to code the `compute` and (optionally) the `endOfStream` method
- IO system is decoupled from processing system (process whilst transferring data)
- Broker-based communication transparently establishes point-to-point connection
- Any workflow device has full access to the live control-system
Once resources are available, input channels request new data from connected output channels.

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

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

Configurable output channel behavior in case no input currently available:

- throw
- queue
- wait
- drop
Once resources are available, input channels request new data from connected output channels.

Configurable output channel behavior in case no input currently available:

*throw, queue, wait, drop*
Once resources are available, input channels request new data from connected output channels.

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

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

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

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

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

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

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

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

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

Configurable output channel behavior in case no input currently available: 

*throw, queue, wait, drop*
Once resources are available, input channels request new data from connected output channels.

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

- Configurable output channel behavior in case no input currently available: throw, queue, wait, drop
Karabo: The European XFEL software framework

Workflows 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: throw, queue, wait, drop.

Pixel parallel processing (one GPU thread per pixel)
Once resources are available, input channels request new data from connected output channels.

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

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

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

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

Configurable output channel behavior in case no input currently available: throw, queue, wait, drop.
Karabo: The European XFEL software framework

Workflows 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: **throw, queue, wait, drop**

**Pixel parallel processing**
(one GPU thread per pixel)
Workflows 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: **throw, queue, wait, drop**
Workflows 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:

- throw
- queue
- wait
- drop
Once resources are available, input channels request new data from connected output channels.

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

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

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

Configurable output channel behavior in case no input currently available: throw, queue, wait, drop
Workflows 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: throw, queue, wait, drop.
Karabo: The European XFEL software framework

Workflows 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: throw, queue, wait, drop.

Pixel parallel processing (one GPU thread per pixel)
Karabo: The European XFEL software framework

Workflows 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: throw, queue, wait, drop.
Workflows 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:**
- **throw, queue, wait, drop**
Once resources are available, input channels request new data from connected output channels.

Configurable output channel behavior in case no input currently available:

- throw, queue, wait, drop
Once resources are available, input channels request new data from connected output channels.

Configurable output channel behavior in case no input currently available: throw, queue, wait, drop.
Karabo: The European XFEL software framework

Workflows 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: throw, queue, wait, drop
Once resources are available, input channels request new data from connected output channels.

Configurable output channel behavior in case no input currently available:

**throw, queue, wait, drop**
Once resources are available, input channels request new data from connected output channels.

Configurable output channel behavior in case no input currently available: **throw, queue, wait, drop**
Io whilst computing

Pixel parallel processing
(one GPU thread per pixel)

Once resources are available, input channels request new data from connected output channels.

Configurable output channel behavior in case no input currently available:

  throw, queue, wait, drop
Once resources are available, input channels request new data from connected output channels.

Configurable output channel behavior in case no input currently available:

- throw
- queue
- wait
- drop
Once resources are available, input channels request new data from connected output channels

- Configurable output channel behavior in case no input currently available: 
  - throw, queue, wait, drop
Workflows 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: throw, queue, wait, drop.

IO whilst computing

Notification about new data possible to obtain

Pixel parallel processing
(one GPU thread per pixel)
Once resources are available, input channels request new data from connected output channels.

Configurable output channel behavior in case no input currently available: **throw, queue, wait, drop**
Karabo: The European XFEL software framework

Workflows 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: **throw, queue, wait, drop**
Once resources are available, input channels request new data from connected output channels.

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

Configurable output channel behavior in case no input currently available: **throw, queue, wait, drop**
Karabo: The European XFEL software framework

Workflows 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: throw, queue, wait, drop.

Pixel parallel processing (one GPU thread per pixel)
Karabo: The European XFEL software framework

Workflows 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: throw, queue, wait, drop.

IO whilst computing

Pixel parallel processing
(one GPU thread per pixel)
Workflows 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: throw, queue, wait, drop
Workflows 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: throw, queue, wait, drop.
Workflows 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: throw, queue, wait, drop

Notification about new data possible to obtain

IO whilst computing

Pixel parallel processing (one GPU thread per pixel)
Karabo: The European XFEL software framework

**Workflows 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: **throw, queue, wait, drop**

*IO whilst computing*

*Notification about new data possible to obtain*

*Pixel parallel processing (one GPU thread per pixel)*
Karabo: The European XFEL software framework

Multi-purpose graphical user interface

Navigation

Custom composition area

Configuration

Notifications

Logging / Scripting console

Documentation

Burkhard Heisen (WP76)
Before you even start: Login

- **Username**: heisenb
- **Password**: ********
- **Provider**: LOCAL
- **Hostname**: localhost
- **Port**: 44444

**Central DB**
- **userId**
- **sessionToken**
- **defaultAccessLevel**
- **accessList**

**GUI or CLI**
- **username**
- **password**
- **provider**
- **ownIP**
- **brokerHost**
- **brokerPort**
- **brokerTopic**

**Broker-Message**
- **Header** [...]
  - __uid=42
  - __accessLevel="admin"
- **Body** [...]

**GUI-Srv**

**Device**

**Locking**
- if is locked:
  - if is __uid == owner then ok

**Access control**
- if __accessLevel >= visibility:
  - if __accessLevel >= param.accessLevel then ok

1. Authorizes
2. Computes context based access levels
Karabo: The European XFEL software framework

Property/Command composition

- Text field
- Button widget
- Display widget
- Editable widget

Drag & drop

Burkhard Hieisen (WP76)
Karabo: The European XFEL software framework

Device (workflow) composition

Workflow node (device)

Draw connection

Burkhard Heisen (WP76)
Main functionality:

- Exploring the distributed system topology (hosts, device-servers, devices, their properties/commands, etc.)
  
- `getServers`, `getDevices`, `getClasses`

- `instantiate`, `kill`, `set`, `execute` (in “wait” or “noWait” fashion), `get`, `monitorProperty`, `monitorDevice`

- Even polled interface will never really poll, but is event-driven under the hood

- Remote auto-completion which is access-role and state aware
Karabo: The European XFEL software framework

The command line interface (CLI)

```
In [2]: c.help"
Test_Conveyor_1       Test_SimulatedCamera_1
Test_DataGenerator_1   Test_SimulatedCamera_2

In [2]: c.help("Test_DataGenerator_1")

----- HELP -----
Schema: DataGenerator

   .version (STRING)
      Assignment    : OPTIONAL
      Default value : 1.0
      Description   : The version of this device class
      Access mode   : read only

   .connection (CHOICE_OF_NODES)
      Assignment    : OPTIONAL
      Default value : Jms
      Description   : The connection to the communication layer of the distributed system

   .visibility (INT32)
      Assignment    : OPTIONAL
      Default value : 0
      Description   : Configures who is allowed to see this device at all
      Access mode   : reconfigurable

   .classId (STRING)
      Assignment    : OPTIONAL
      Default value : Device
      Description   : The (factory)-name of the class of this device
      Access mode   : read only
```
Karabo: The European XFEL software framework

The command line interface (CLI)

```python
In [3]: c.get("Test_Si
Test_SimulatedCamera_1 Test_SimulatedCamera_2
areaOfInterest    connection.Jms.port
connection.Jms.acknowledgeMode
connection.Jms.acknowledgeTimeout
connection.Jms.deliveryInhibition
connection.Jms.destinationName
connection.Jms.hostname
connection.Jms.messageTimeToLive
connection.Jms.messagingDomain
connection.Jms.password
connection.Jms.ping
imageFilename
imageFormat
imageFileTimestamp
data
image.encode
image.im
image.im
image.im
image.im
image.isBigEndian

In [3]: c.get("Test_SimulatedCamera_1", "sensorHeight"
"sensorWidth"
"sensorHeight"
"sensorWidth"

Out[3]: 25.49

In [4]: c.set("Test_Sim
Test_SimulatedCamera_1 Test_SimulatedCamera_2
areaOfInterest exposureTime pixelGain triggerMode
connection.Jms.acknowledgeMode
connection.Jms.acknowledgeTimeout
connection.Jms.deliveryInhibition
connection.Jms.destinationName
connection.Jms.hostname
connection.Jms.messageTimeToLive
connection.Jms.messagingDomain
connection.Jms.password
connection.Jms.ping
imageFilename
imageFormat
imageFileTimestamp
data
image.encode
image.im
image.im
image.im
image.im
image.isBigEndian

In [4]: c.set("Test_SimulatedCamera_1", "exposureTime", 0.5)
Out[4]: (True, '')

In [5]: c.set("Test_SimulatedCamera_1", "triggerMode", "nop")
Out[5]: (False,
=Value nop for parameter "triggerMode" is not one of the valid options: Internal,Software

In [8]: c.execute("Test_SimulatedCamera_1", "acquire")
Out[8]: (True, 'Ok.Acquisition')

In [9]: c.execute("Test_SimulatedCamera_1", "stop")
Out[9]: (True, 'Ok.Ready')
```
Karabo: The European XFEL software framework

The command line interface (CLI)
Selected Features

- Cross-network **signals and slot** implementation (Qt style)
  - Native function calls providing native argument passing
  - Runtime setup of signals, slots and connect statements

- High performance **dictionary object**
  - Fully recursive key to any value mapper (also in C++)
  - Keeps insertion order and supports attributes
  - Serializes to XML, Binary, HDF5, DB, JMS-Message, etc.

- **Event driven system throughout**, incl. **event driven archiving**
  - Archiving happens completely transparent (no extra tooling needed)
  - Archiving policies are configured per property within device code

- **User centric, access controlled** system
  - **Context based authentication** (who, where, when)
  - Kerberos is integrated

- Full system available in C++ and “pythonic” Python
  - Devices and workflows can intermix both languages (access to **numpy**, **scipy**, etc.)

- Fully functional also without master or database (good for testing/developing)

- Easy install (both framework and devices) using **software bundle** architecture
Acknowledgments

Karabo contributors (internal)
K. Wrona       S. Esenov
K. Weger       J. Szuba
D. Boukhelef   I. Kozlova
A. Parenti     L. Maia
S. Hauf        P. Gessler
N. Coppola     J. Tolkiehn
J. Elizondo    H. Höhne
C. Youngman

Karabo contributors (external)
C. Yoon (CFEL)  N. D. Loh (SLAC)
T. Nicholls (STFC) K. Rehlich (DESY / DOOCS)
...

Burkhard Heisen (WP76)
Thank you for your kind attention.