Beamline experiments with BLISS @ ESRF

presented by Matias Guijarro - BLISS team

Beamline Control Unit / Software Group

ESRF, Grenoble, France
ESRF Extremely Brilliant Source (EBS) update program
ESRF Extremely Brilliant Source (EBS) update program

- 4th generation light source
- 150 M€ investment (2015-2022)
- 100x better X-ray beams
- New state-of-the-art beamlines
- Data as a Service strategy

ESRF-EBS
BLISS project goals
BLISS project goals

State-of-the-art beamline control
Advanced scans, Trajectories,
Data management
BLISS project goals

State-of-the-art beamline control
Advanced scans, Trajectories,
Data management

Integrated environment
Configuration, Command Line Interface, Live
Data Display
BLISS project goals

State-of-the-art beamline control
Advanced scans, Trajectories, Data management

Integrated environment
Configuration, Command Line Interface, Live Data Display

Ready for new challenges
Online data analysis, live feedback, extensibility of the system
BLISS philosophy
Python (3.7=>) everywhere
Python (3.7=>) everywhere

Software library first, with a set of tools built upon

- Command Line Interface
- Configuration application
- Online visualization
Python (3.7=>) everywhere

Software library first, with a set of tools built upon

- Command Line Interface
- Configuration application
- Online visualization

To provide a generic scan engine for all kind of data acquisition procedures
BLISS philosophy

Python (3.7=>) everywhere

Software library first, with a set of tools built upon

- Command Line Interface
- Configuration application
- Online visualization

To provide a generic scan engine for all kind of data acquisition procedures

Producer/Consumer model for Data Management, to decouple acquisition and data storage
## BLISS modular architecture

**1. Beacon server**
- **Data management**
  - Acq. Channel
  - Acq. Chain
  - Acq. Master
  - Acq. Slave
- **Scanning**
  - Scannable (Axis)
  - Acq. controller
  - Counter
  - MCA
  - Motor
  - Lima (2D)
  - Regulation
- **Control & Acquisition objects**
  - Acq. Control
  - Acq. Master
  - Acq. Slave

**2. Communication**
- RPC
- modbus
- serial
- tcp/udp
- gpib
- Tango

**3. Settings**
- Keithley
- Motors: IcePAP, galil, Pl...
- Xia
- Pilatus
- Eurotherm, Oxford

**4. Data archiving**
- Online data analysis
- Data visualization
- Data archiving
Beacon server

Devices & sequences configuration in YAML format

Web interface for configuration editing

User sessions to group beamline devices for an experiment, Python setup file

configuration
Beacon static configuration service

Web interface for configuration editing

Devices & sequences configuration in YAML format

User sessions to group beamline devices for an experiment,
Python setup file

Can replace TANGO DB to have all beamline configuration at the same place
Beacon server

Devices & sequences configuration in YAML format

Web interface for configuration editing

User sessions to group beamline devices for an experiment, Python setup file

Configuration + services based on redis

Message broker: data channels, distributed lock

Persistent settings cache

Can replace TANGO DB to have all beamline configuration at the same place

Transient data store
synchronization
low latency
flexibility
Acquisition Chain

tree with master and slave nodes
Acquisition Chain

tree with master and slave nodes

Acquisition Master

triggers data acquisition (can also take data)
2. BLISS scan framework

- **Acquisition Chain**: tree with master and slave nodes
  - **Acquisition Master**: triggers data acquisition (can also take data)
  - **Acquisition Slave**: takes data
BLISS scan framework

**Acquisition Chain**
- tree with master and slave nodes

**Acquisition Master**
- triggers data acquisition
  - can also take data

**Acquisition Slave**
- takes data
Continuous scan with a motor triggering MCA and 2D detector acquisition, while a timer triggers diode readings.

```python
def cscan(motor, start, stop, npoints, time):
    # create a mc chain
    chain = AcquisitionChain(parallel.prepare=True)
    # create the monitor timer
    monitor_timer = SoftwareTimerMaster(1., name="monitor_timer",
        npoints=0)
    # create acquisition device for the monitor diode
diode_device = SamplingCounterAcquisitionDevice(diode1, count_time=1.,
        npoints=0)
    # Associate them in the chain
    chain.add(monitor_timer, diode_device)
    # Now the fast acquisition
    # create a motor master for a position trigger
    master = SoftwarePositionTriggerMaster(motor, start, stop, npoints,
        time=time)
    # The spectrum device MCA
    mca_acq = McaAcquisitionDevice(mca, npoints=npoints,
        trigger_mode=McaAcquisitionDevice.GATE,
        counters=list(mca.counters))
    chain.add(master, mca_acq)
    # The image detector
    lima_master = LimaAcquisitionMaster(frelon,
        acq.nb_frames=npoints, acq.trigger_mode=EXTERNAL_GATE)
    lima_master.add_counter(frelon.image)
    chain.add(master, lima_master)
    # Finally build the scan and run it.
    scan = Scan(chain, name="cscan")
    scan.run()
    return scan
```
While a scan is running, **data is published** to the redis database provided by Beacon.
While a scan is running, **data is published** to the redis database provided by Beacon:

- scalar values are **stored directly**
- bigger data (images, spectra) is **just referenced**
- configurable time to live (TTL)
While a scan is running, **data is published** to the redis database provided by Beacon.

- scalar values are **stored directly**
- bigger data (images, spectra) is **just referenced**
- configurable time to live (TTL)

Any external process can access redis data to perform **online data analysis** or **live feedback** for example.
BLISS development
BLISS development methodology

- backlog (tasks)
- new features
- bug fixes
BLISS development methodology

Kanban

- new features
- bug fixes
- backlog (tasks)
BLISS development methodology

Kanban

backlog (tasks)

Prioritized todo

new features

bug fixes
BLISS development methodology

Kanban

backlog (tasks)

Prioritized todo

Analysis phase

new features

bug fixes
BLISS development methodology

Kanban

Prioritized todo
Analysis phase
Coding

backlog (tasks)

new features
bug fixes
BLISS development methodology

Kanban

- Prioritized todo
- Analysis phase
- Coding
- Integration tests

backlog (tasks)

new features

bug fixes
BLISS development methodology

Kanban

- Prioritized todo
- Analysis phase
- Coding
- Integration tests
- Ready to merge

new features
bug fixes

backlog (tasks)
BLISS project development tools

Issues list, Kanban board, wiki, and more on ESRF-hosted gitlab

Continuous Integration: gitlab-ci

Deployment: Ansible

Documentation: MkDocs

Code formatting: Black
Sharing knowledge within the BLISS team
Sharing knowledge within the BLISS team

Daily stand-ups
Sharing knowledge within the BLISS team

Daily stand-ups

"Stop and Solve" meetings
Sharing knowledge within the BLISS team

- Daily stand-ups
- Pair programming
- "Stop and Solve" meetings
Sharing knowledge within the BLISS team

- Daily stand-ups
- Pair programming
- "Stop and Solve" meetings
- Guidelines, good practices (very useful for newcomers)
BLISS team ensures Quality Assurance
BLISS team ensures Quality Assurance

Automatic tests pipeline
(980+ tests)
BLISS team ensures Quality Assurance

Automatic tests pipeline
(980+ tests)

Only merge code with documentation
BLISS team ensures Quality Assurance

Automatic tests pipeline (980+ tests)

Systematic code review

Only merge code with documentation
BLISS team ensures Quality Assurance

- Automatic tests pipeline (980+ tests)
- Systematic code review
- Only merge code with documentation
- Integration tests on beamlines (real hardware)
Dealing with the technical debt
Dealing with the technical debt

Technical debt = Tetris game (you can't win)
Dealing with the technical debt

Technical debt = Tetris game
(you can't win)

Refactoring helps to fill the holes,
but with time penalty
In 2018, first tests with users: MX, ID10, ID11, ID13, ID31
Conclusion

BLISS is the new Python-based DAQ and experiments control system
Conclusion

BLISS is the new Python-based DAQ and experiments control system

BLISS is being designed for the needs of the EBS beamlines
Conclusion

BLISS is the new Python-based DAQ and experiments control system

BLISS is being designed for the needs of the EBS beamlines

Half of the beamlines are being converted to BLISS during the EBS shutdown (2019-2020)
Conclusion

BLISS is the new Python-based DAQ and experiments control system

BLISS is being designed for the needs of the EBS beamlines

Half of the beamlines are being converted to BLISS during the EBS shutdown (2019-2020)

Would be happy to start collaborations around BLISS with interested people
Acknowledgements

head of Software Group: A. Gotz, head of BCU: J. Meyer

BLISS Core Development team

From left to right:
B. Formet (CEA), M. Guijarro,
P. Pancino, L. Pithan,
P. Guillou,
S. Petitdemange,
C. Guilloud

Beamline Operation team

A. Beteva, G. Berruyer, L. Claustre, S. Fisher, A. Homs, R. Homs, M-C. Lagier, A. Mauro,
C. Muzelle, S. Ohlsson, M. Oscarsson, E. Papillon, F. Sever, V. Valls, H. Witsch