



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



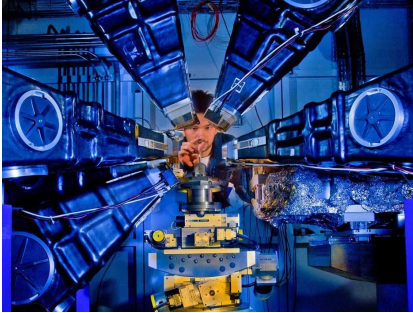
ESRF-EBS

Beamline Instrumentation Support Software



BLISS project goals

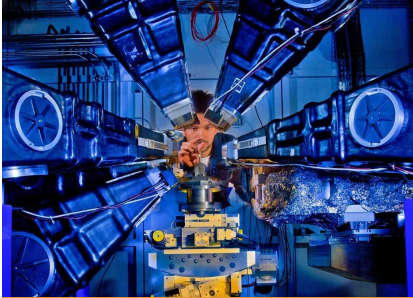
BLISS project goals



State-of-the-art beamline control

Advanced scans, Trajectories,
Data management

BLISS project goals



State-of-the-art beamline control

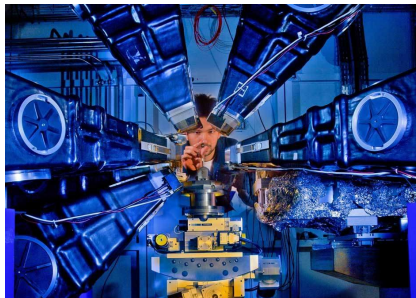
Advanced scans, Trajectories,
Data management

Integrated environment

Configuration, Command Line Interface, Live
Data Display



BLISS project goals

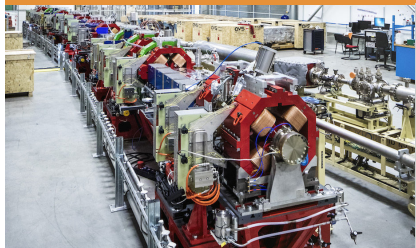


State-of-the-art beamline control

Advanced scans, Trajectories,
Data management

Integrated environment

Configuration, Command Line Interface, Live
Data Display

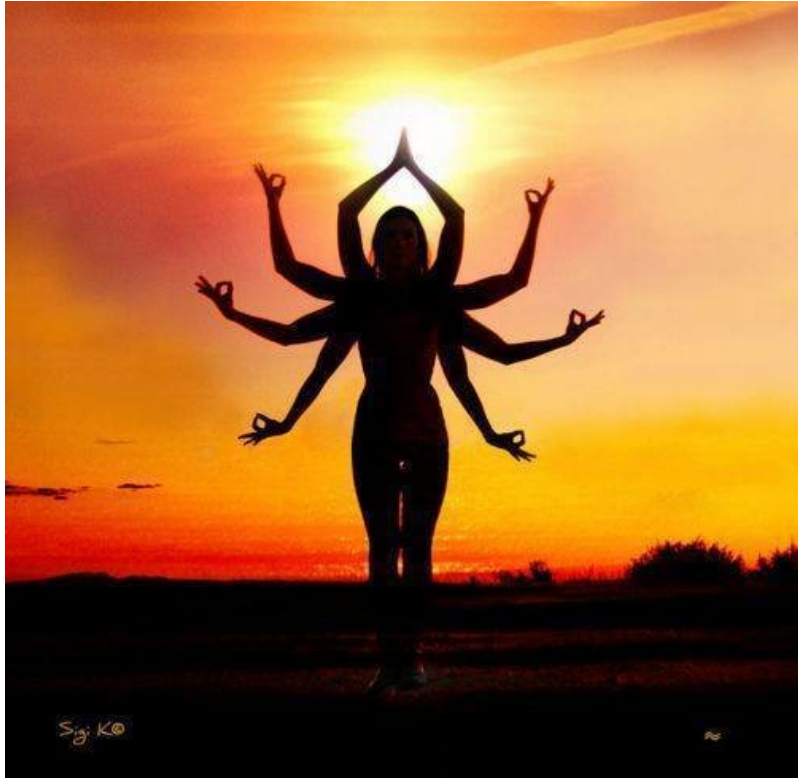


Ready for new challenges

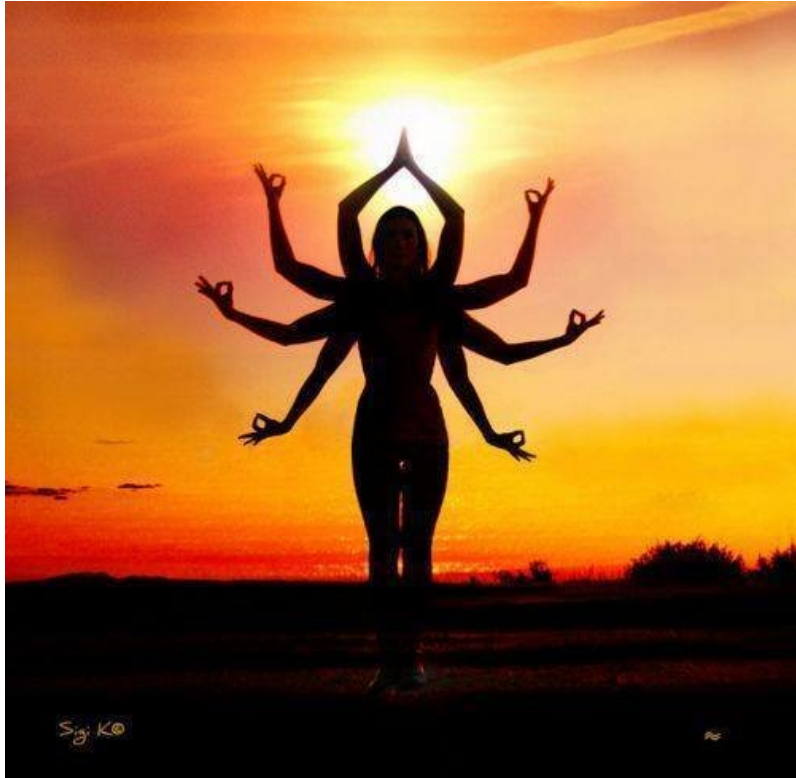
Online data analysis, live feedback,
extensibility of the system



BLISS philosophy

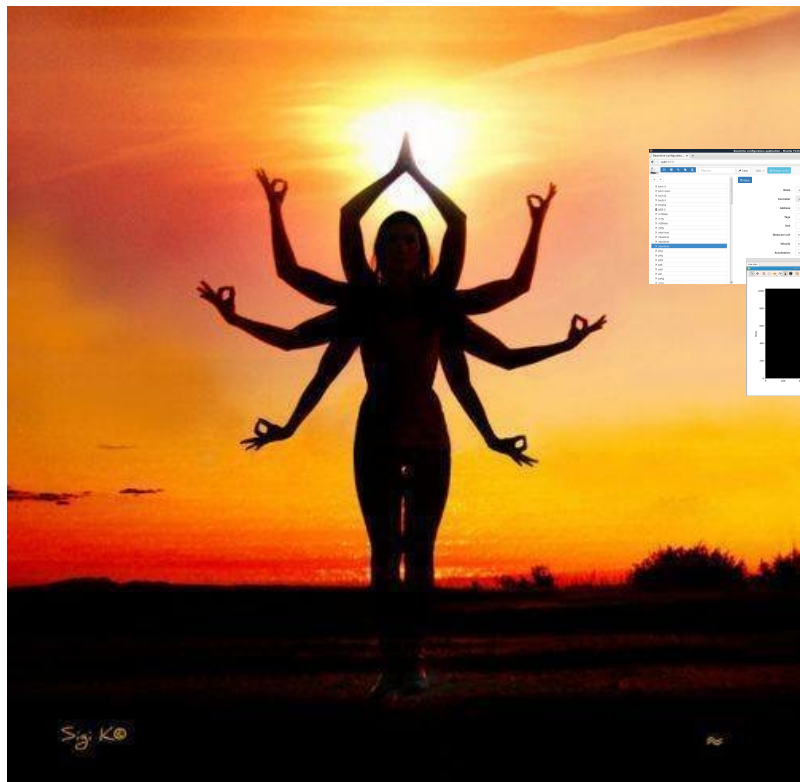


BLISS philosophy



Python (3.7=>) everywhere 

BLISS philosophy



Python (3.7=>) everywhere



**Software library first,
with a set of tools built upon**

- Command Line Interface
- Configuration application
- Online visualization

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**



BLISS philosophy

Python (3.7=>) everywhere



**Software library first,
with a set of tools built upon**

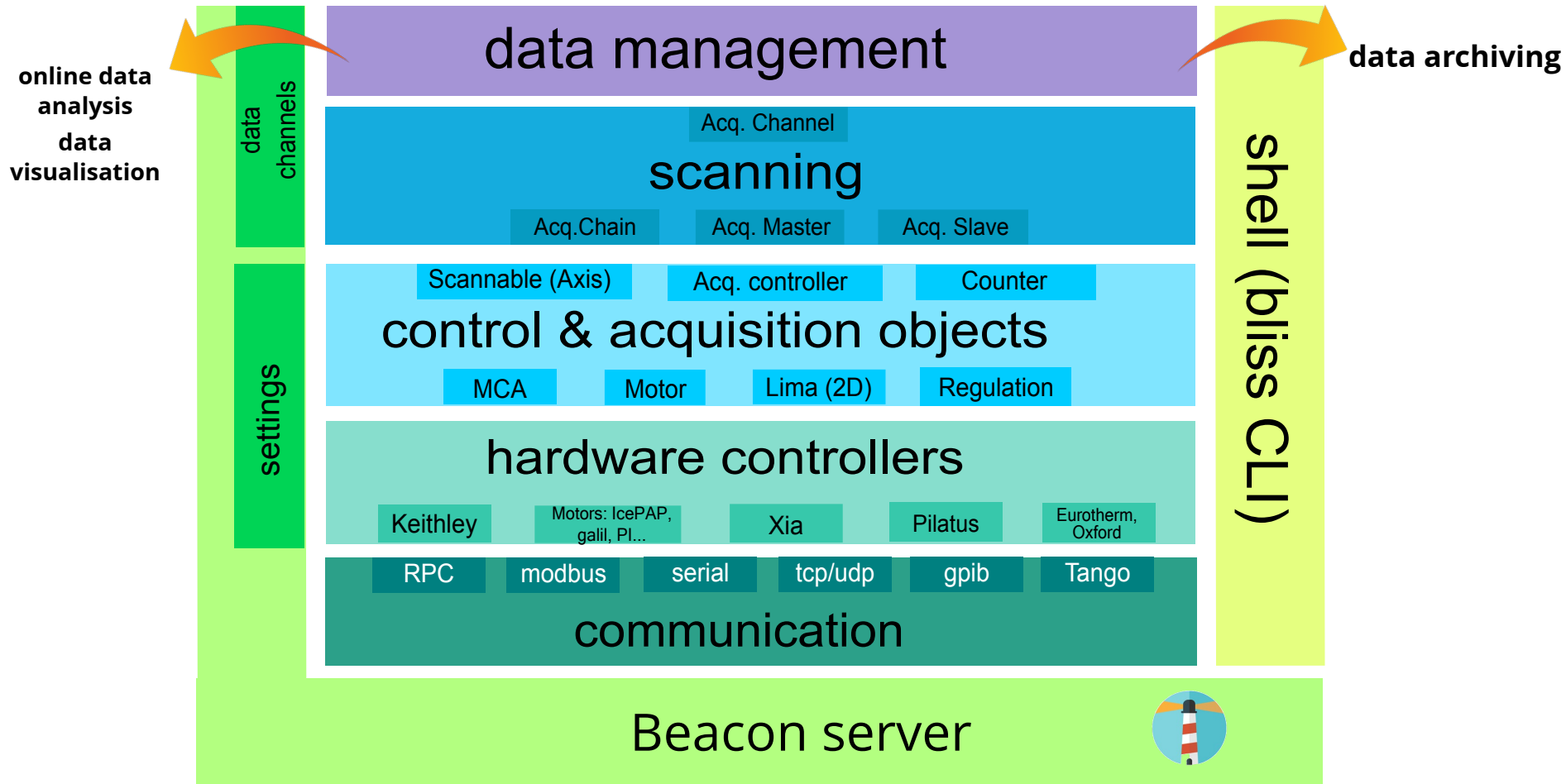
- Command Line Interface
- Configuration application
- Online visualization

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

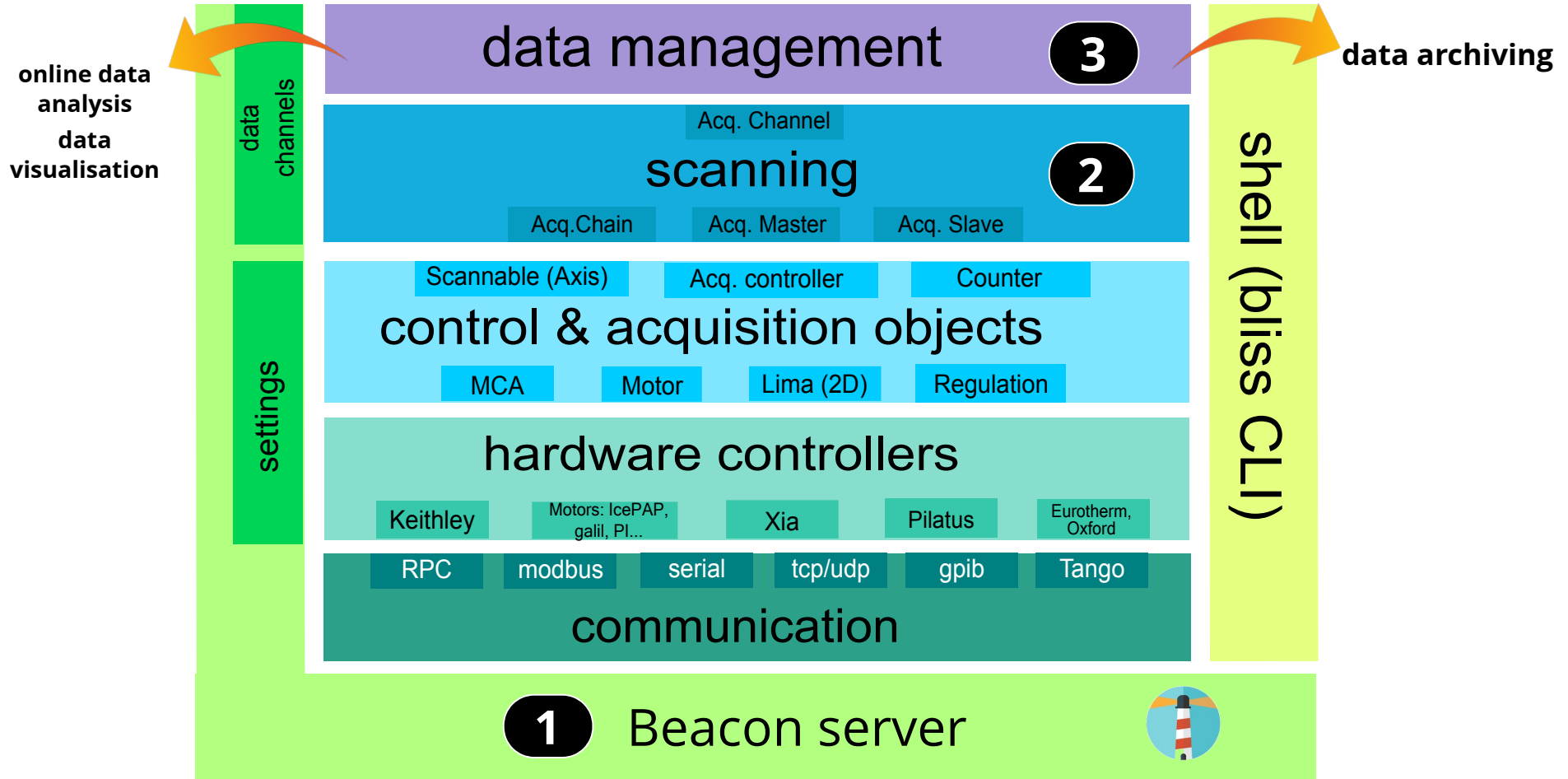
**Producer/Consumer model for Data Management,
to decouple acquisition and data storage**



BLISS modular architecture



BLISS modular architecture

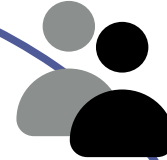




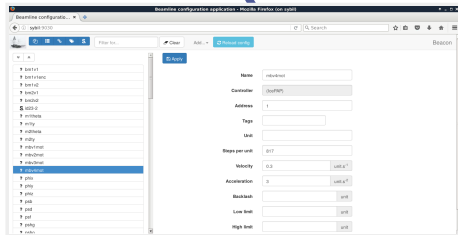
Devices & sequences configuration in YAML format



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



configuration



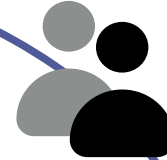
Web interface for configuration editing



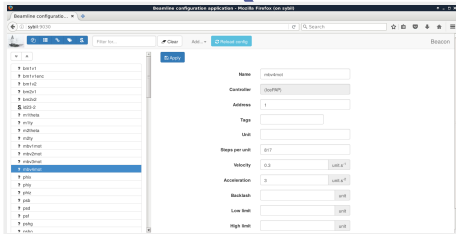
Devices & sequences configuration in YAML format



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



configuration

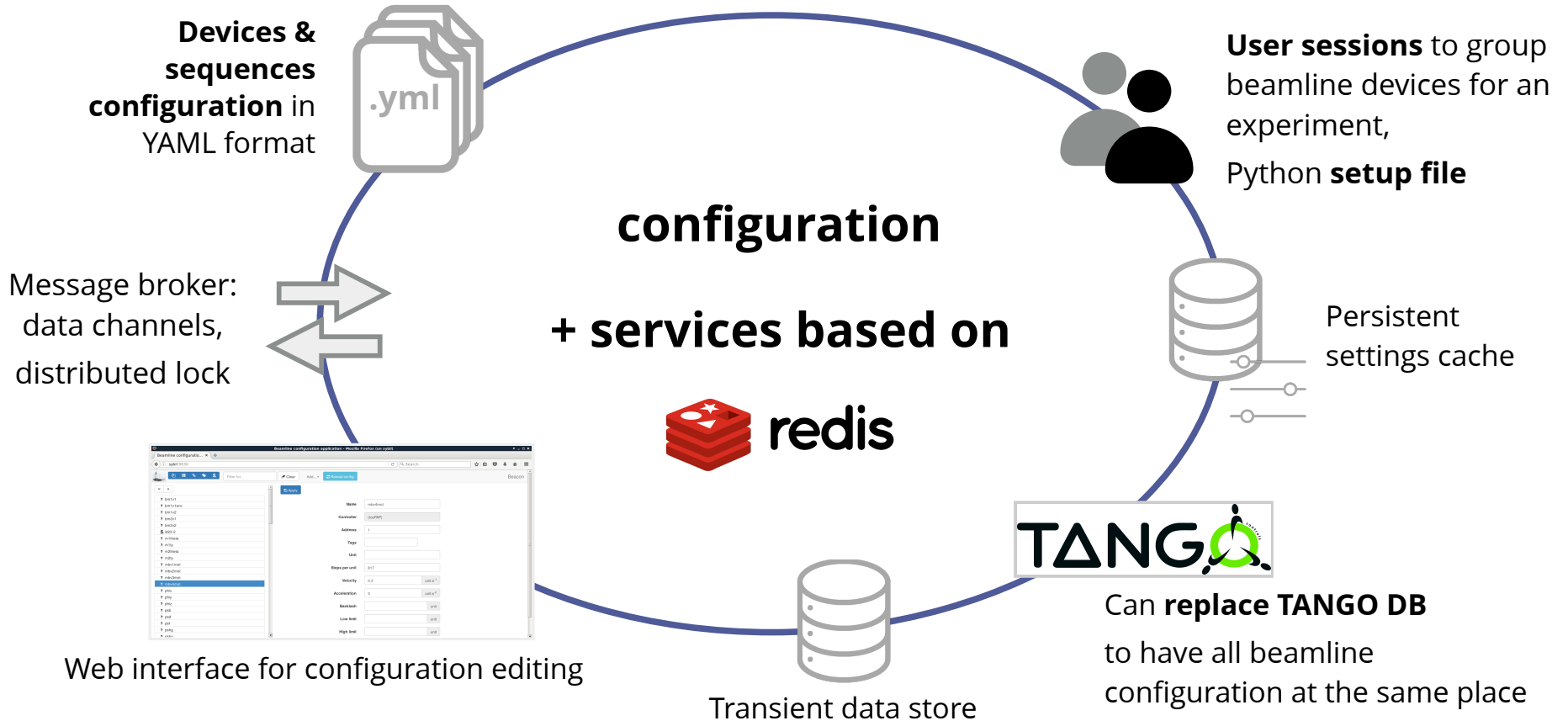


Web interface for configuration editing



Can **replace TANGO DB**

to have all beamline configuration at the same place







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)





Acquisition Chain

tree with master and
slave nodes



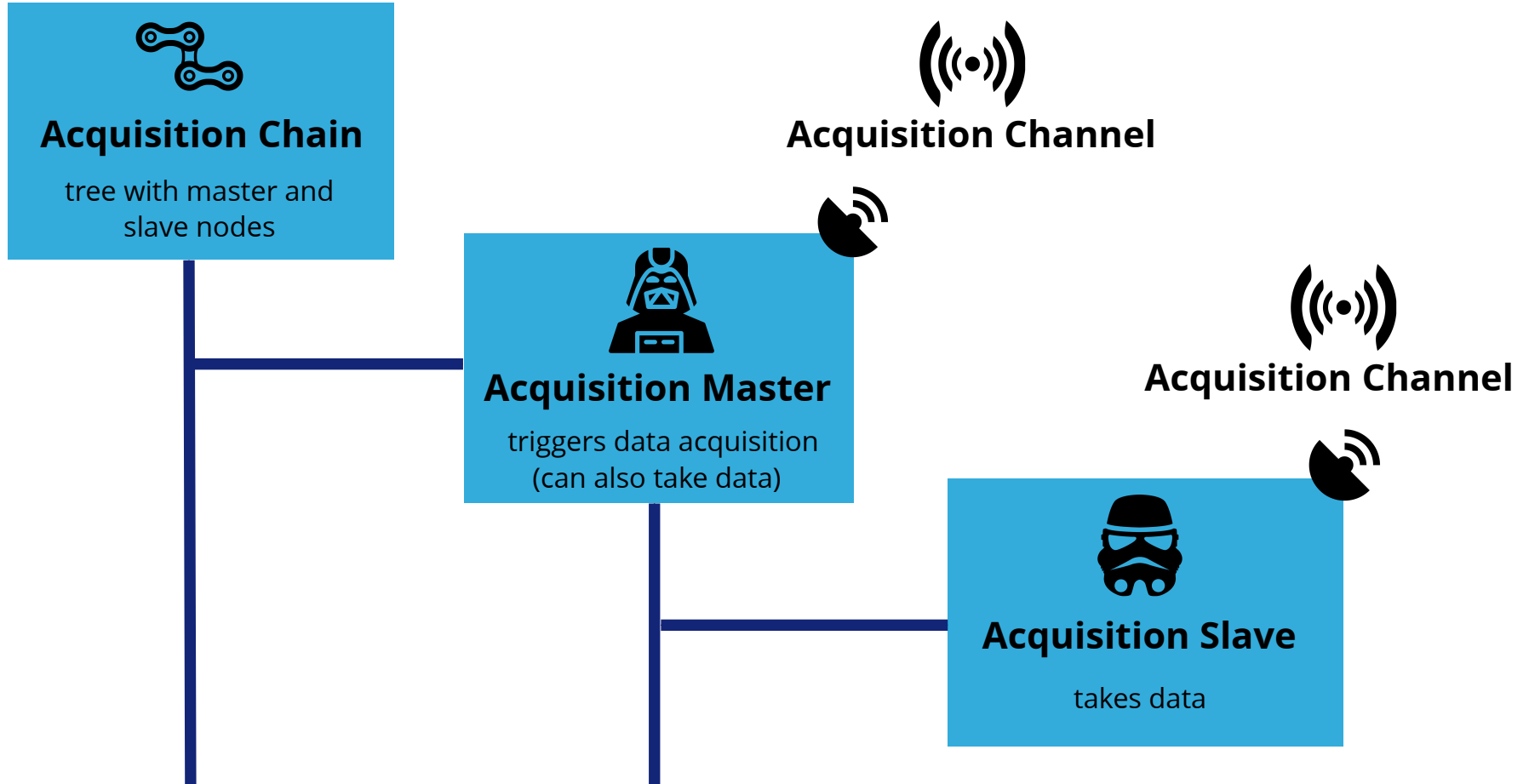
Acquisition Master

triggers data acquisition
(can also take data)



Acquisition Slave

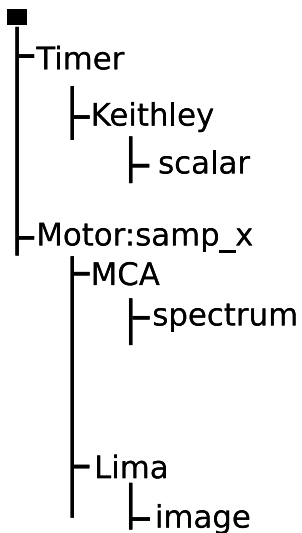
takes data



2

BLISS scan example

Continuous scan with a motor triggering MCA and 2D detector acquisition, while a timer triggers diode readings



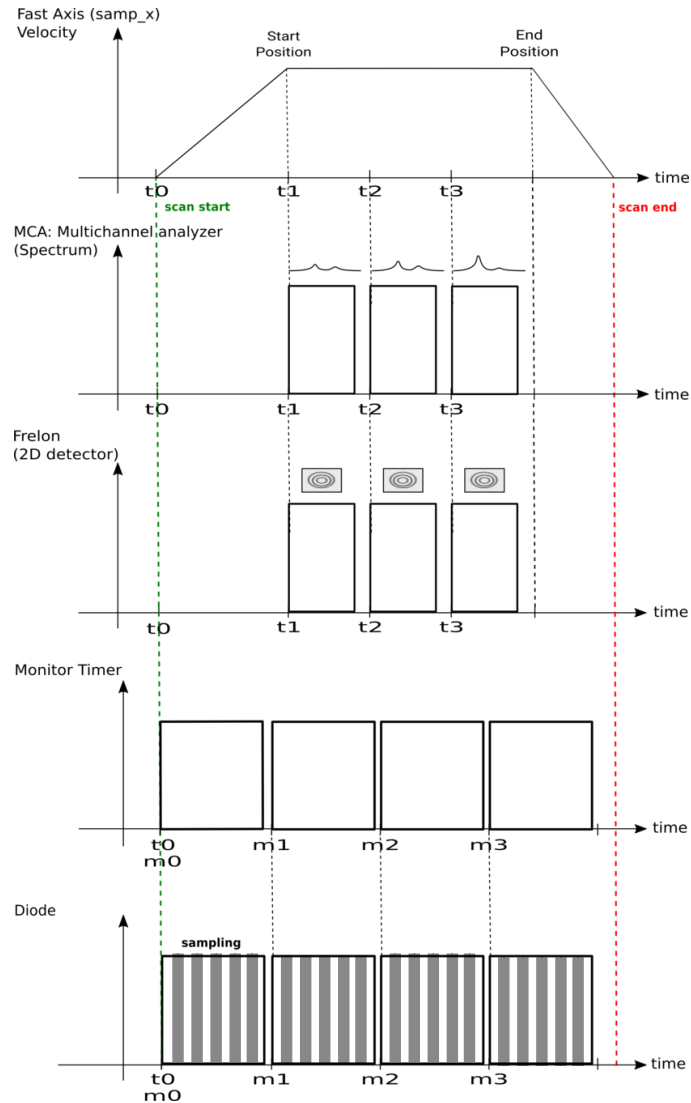
```

def cscan(motor, start, stop, npoints, time):
    # create a new chain
    chain = AcquisitionChain(parallel_prepare=True)
    # create the monitor timer
    # npoints == 0 mean infinite
    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



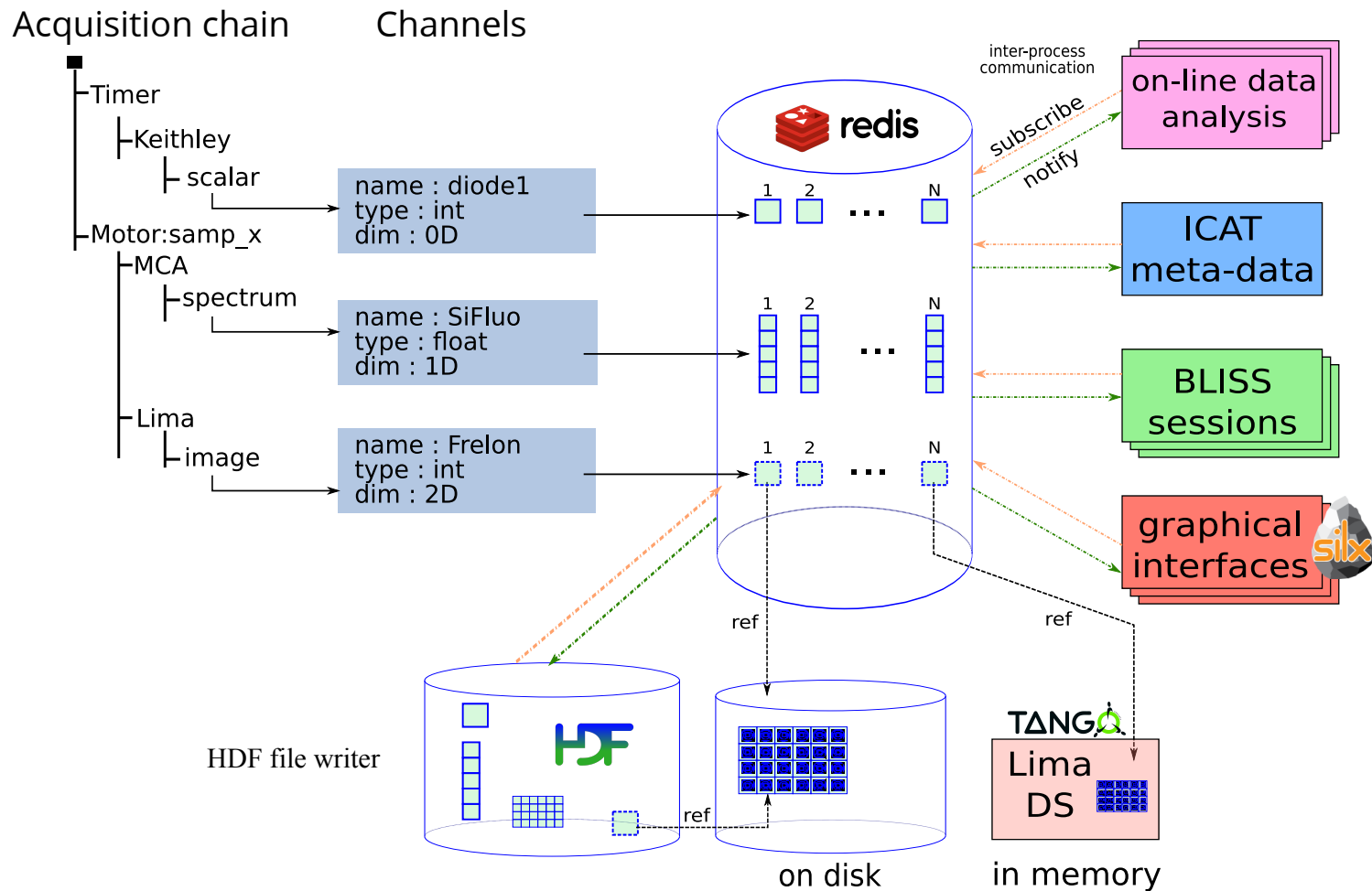
- scalar values are **stored directly**
- bigger data (images, spectra) is **just referenced**
- configurable time to live (TTL)

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



backlog
(tasks)

The screenshot shows a GitLab Kanban board with five columns: 'backlog', 'Development', 'Pending Integration', 'Integration', and 'Done'. The 'backlog' column contains a list of tasks such as 'Motion framework: take into account presentation feedback', 'Motion framework, write user documentation about Encoder objects', 'Calc. axes in scan: export real motor positions in hDFS file and data', 'xNicon plotting', 'Temperature framework renaming', 'Scan info', and 'Temp Image'. The 'Development' column contains tasks like 'Print live plot: 1D display improvement', 'Home search', 'New Wago module and interlocks, TANGO device server', 'lima login not working with simulator', 'Scan display refactoring', 'fullnames containing : and .', 'gitlab-ci: add package building to CI pipeline server', ':counters namespace not accessible from command line', and 'Ctrl space => shell blocks'. The 'Pending Integration' column contains '3 log-table calc: motor controller' and '1903 sets acceleration to 0.001 on axes (at least on IB31)'. The 'Integration' column contains 'Add new MCA: Mythen'. The 'Done' column contains 'aEscan: correct size cm, cmn, peak, etc.'. A blue arrow points from the 'backlog' column to the 'Done' column.

new features



bug fixes

BLISS development methodology

Kanban



backlog
(tasks)

The screenshot shows a GitLab Kanban board with five columns: 'todo', 'development', 'testing', 'production', and 'done'. The 'todo' column contains a list of tasks, some with checkboxes. The 'development' column contains several task cards with titles like 'Scan display refactoring', 'Home search', and 'New Wago module and interlocks, TANGO device server'. The 'testing' column has cards like '3 log-table calc: motor controller' and 'Add new MCA: Mythen'. The 'production' column has a card '1903 sets acceleration to 0.001 on axes (at least on IB31)'. The 'done' column has a card 'aECoor: correct size cm, cmn, peak, etc.'. Each card includes a title, a description, and a status indicator.

Prioritized
todo

new features



bug fixes

BLISS development methodology

Kanban



backlog
(tasks)

The screenshot shows a GitLab Kanban board with five columns representing different stages of the development process. The first column, 'Prioritized todo', contains a list of tasks with checkboxes and priority indicators. The second column, 'Analysis phase', shows tasks with a 'Analysis' label and a diagram of people discussing. The third column, 'Development', contains tasks with 'Development' labels. The fourth column, 'Pending Integration', contains tasks with 'Pending Integration' labels. The fifth column, 'Done', contains tasks with 'Done' labels. The board is titled 'Issue Board' and has a search bar at the top.

Prioritized
todo

Analysis
phase

new features



bug fixes

BLISS development methodology

Kanban



backlog
(tasks)

The screenshot shows a GitLab Kanban board with five columns representing different stages of the development process:

- Prioritized todo:** Contains a list of tasks such as "Motion framework: take into account presentation feedback", "Motion framework, write user documentation about Encoder objects", "Calc. axes in scan: export real motor positions in hDFS file and data", "xNicon plotting", "Temperature framework renaming", "Scan info", and "Temp Inpur".
- Analysis phase:** Contains tasks like "Print live plot: 1D display improvement", "Home search", "New Wago module and interlocks, TANGO device server", and "lima login not working with simulator".
- Coding:** Contains tasks such as "Scan display refactoring", "fullnames containing ' and '", "gitlab-ci: add package building to CI pipeline server", ".counters namespace not accessible from command line", and "Ctrl space => shell blocks".
- Testing/Deployment:** Contains tasks like "3 log-table calc: motor controller", "Add new MCA: Mythen", "1903 sets acceleration to 0.001 on axes (at least on B31)", and "shexapod".
- Done:** The final column on the right, representing completed tasks.

Prioritized
todo

Analysis
phase

Coding

new features



bug fixes

BLISS development methodology

Kanban



backlog
(tasks)

The screenshot shows a GitLab Kanban board with five columns representing different stages of the development process. Each column contains several task cards with titles, issue numbers, and status indicators. The columns are: 'Prioritized todo' (with a horse cart icon), 'Analysis phase' (with a meeting icon), 'Coding' (with a person at a computer icon), 'Integration tests' (with a document and magnifying glass icon), and 'Done' (with a checkmark icon). The board is titled 'Issue Board' and includes search and filter options.

Prioritized
todo

Analysis
phase

Coding

Integration
tests

new features



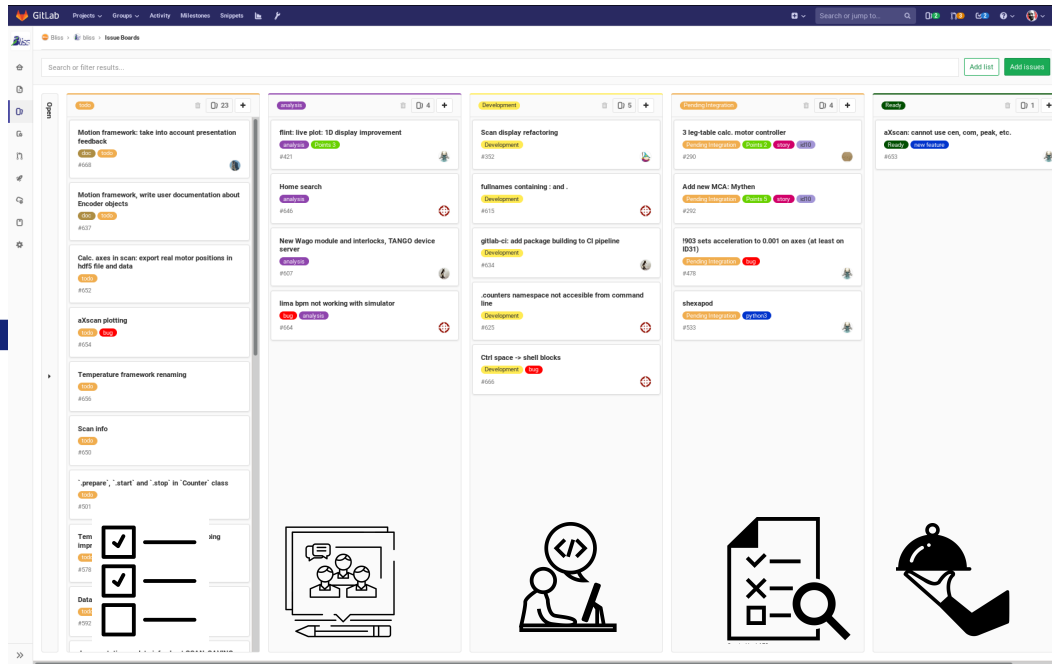
bug fixes

BLISS development methodology

Kanban



backlog
(tasks)



Prioritized
todo

Analysis
phase

Coding

Integration
tests

Ready to
merge

new features



bug fixes

BLISS project development tools

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

Continuous
Integration:

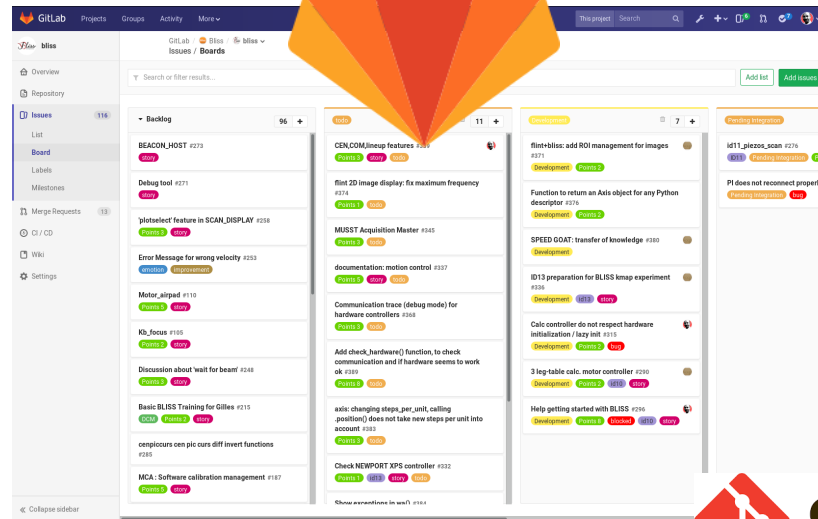
gitlab-ci



pytest

Documentation:

MkDocs



Deployment:



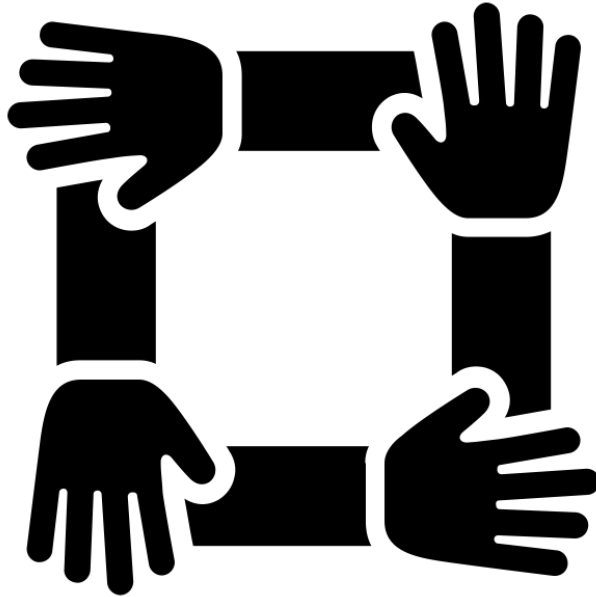
ANSIBLE



git

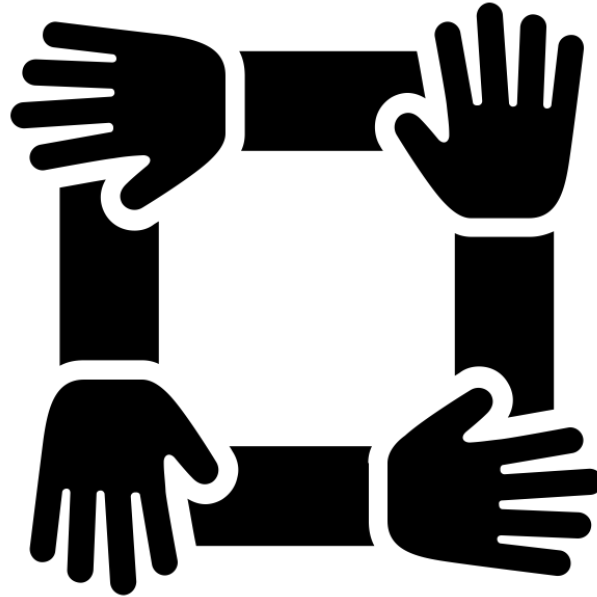
Code formatting: Black

Sharing knowledge within the BLISS team



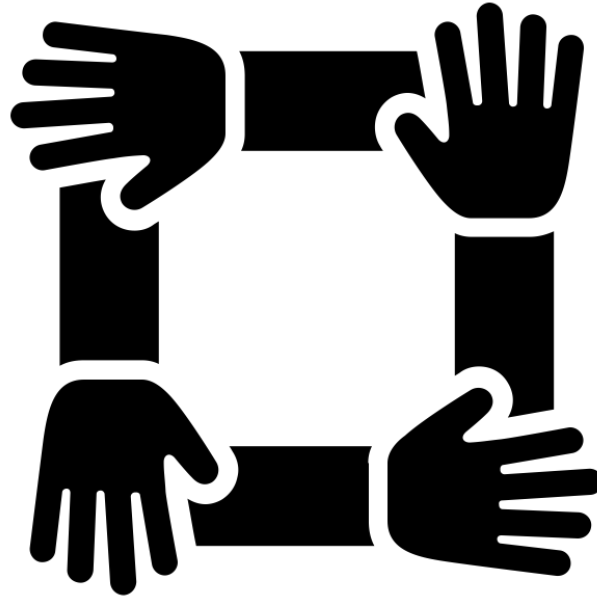
Sharing knowledge within the BLISS team

Daily stand-ups



Sharing knowledge within the BLISS team

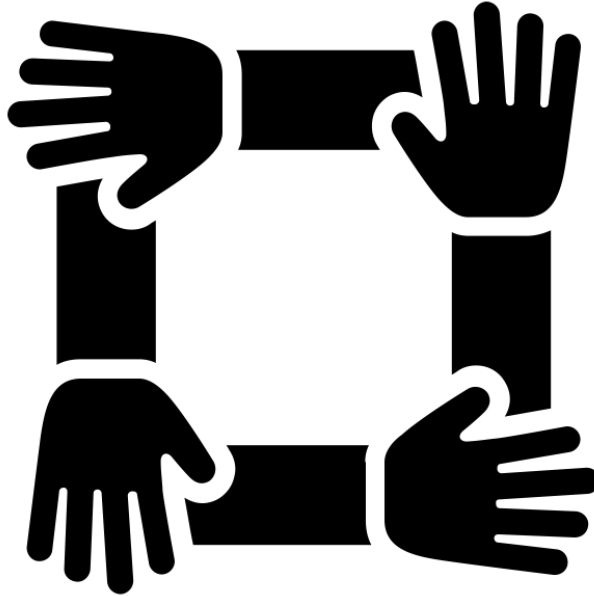
Daily stand-ups



"Stop and Solve" meetings

Sharing knowledge within the BLISS team

Daily stand-ups

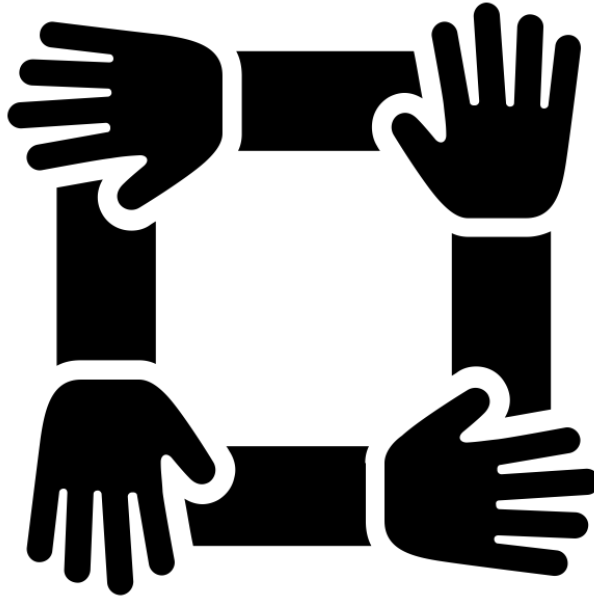


"Stop and Solve" meetings

Pair programming

Sharing knowledge within the BLISS team

Daily stand-ups



"Stop and Solve" meetings

Pair programming

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)



Only merge code with
documentation

Systematic code
review

BLISS team ensures Quality Assurance

Automatic tests pipeline
(980+ tests)



Only merge code with
documentation

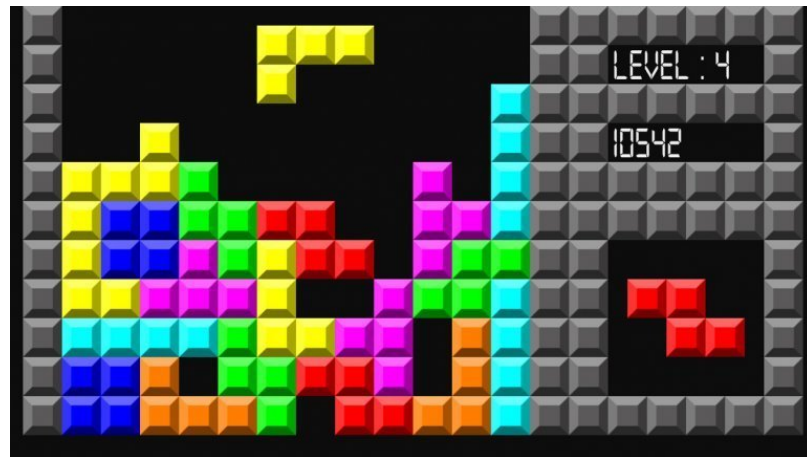
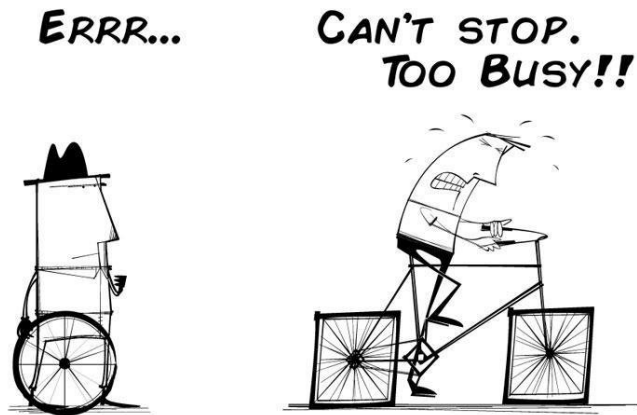
Systematic code
review

Integration tests on
beamlines (real
hardware)

Dealing with the technical debt

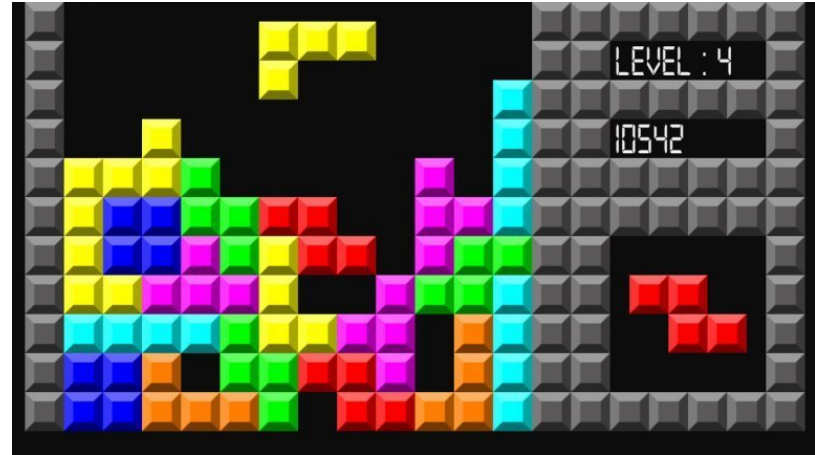
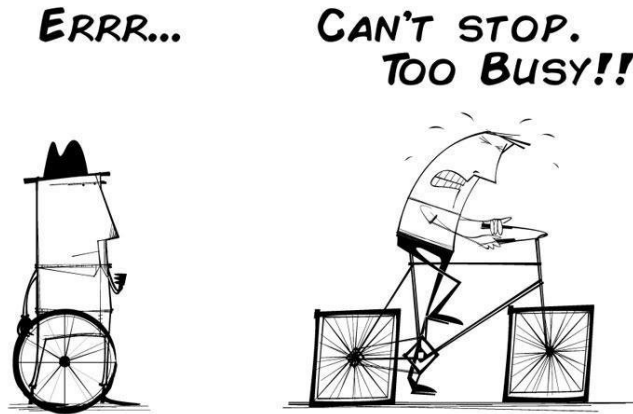
Dealing with the technical debt

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



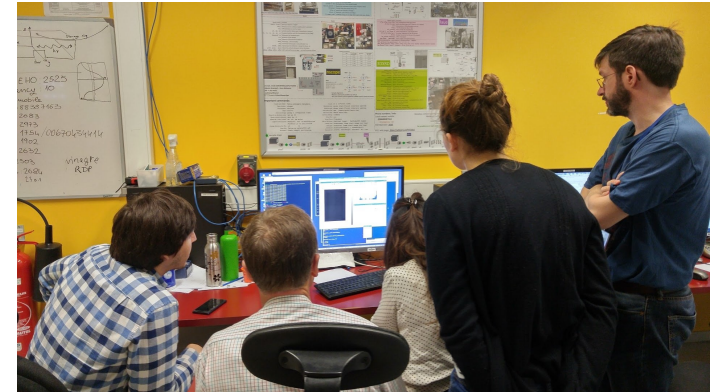
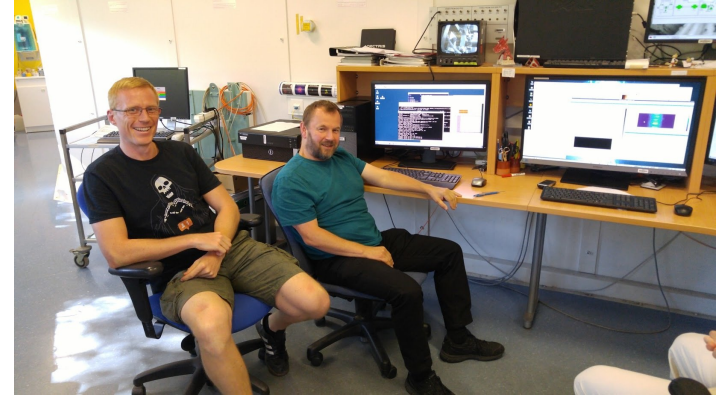
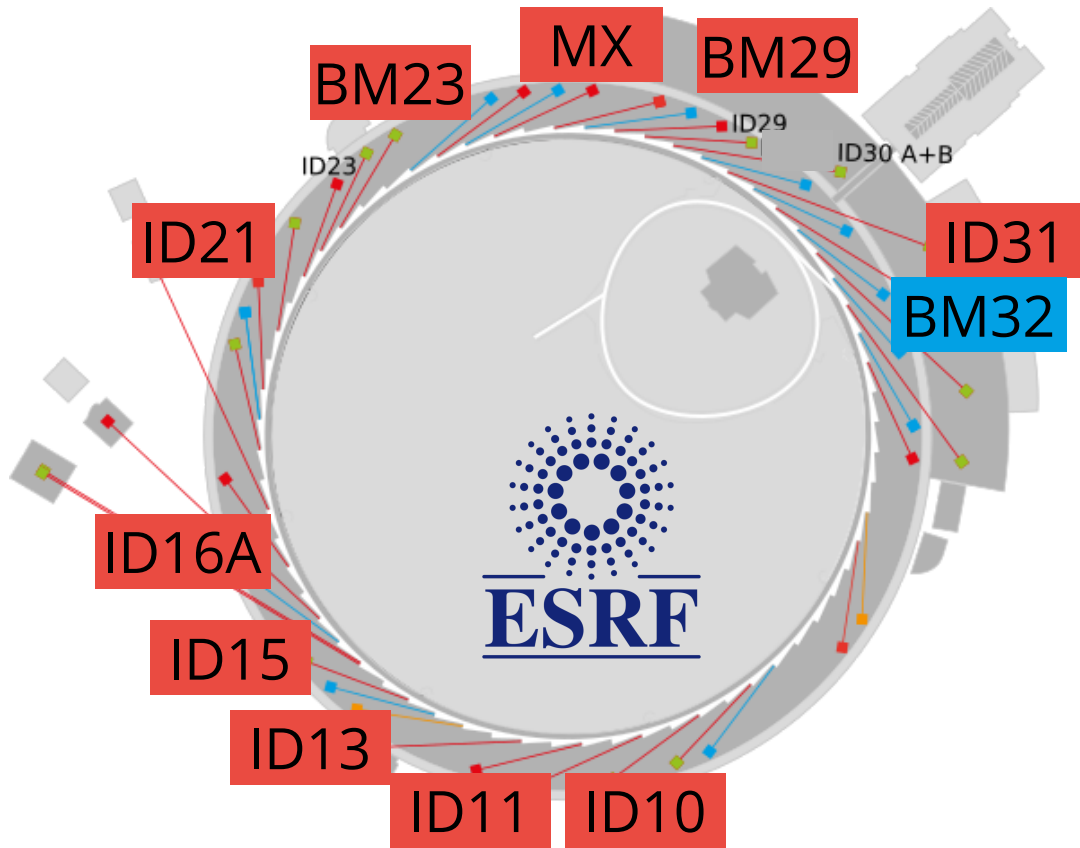
Dealing with the technical debt

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

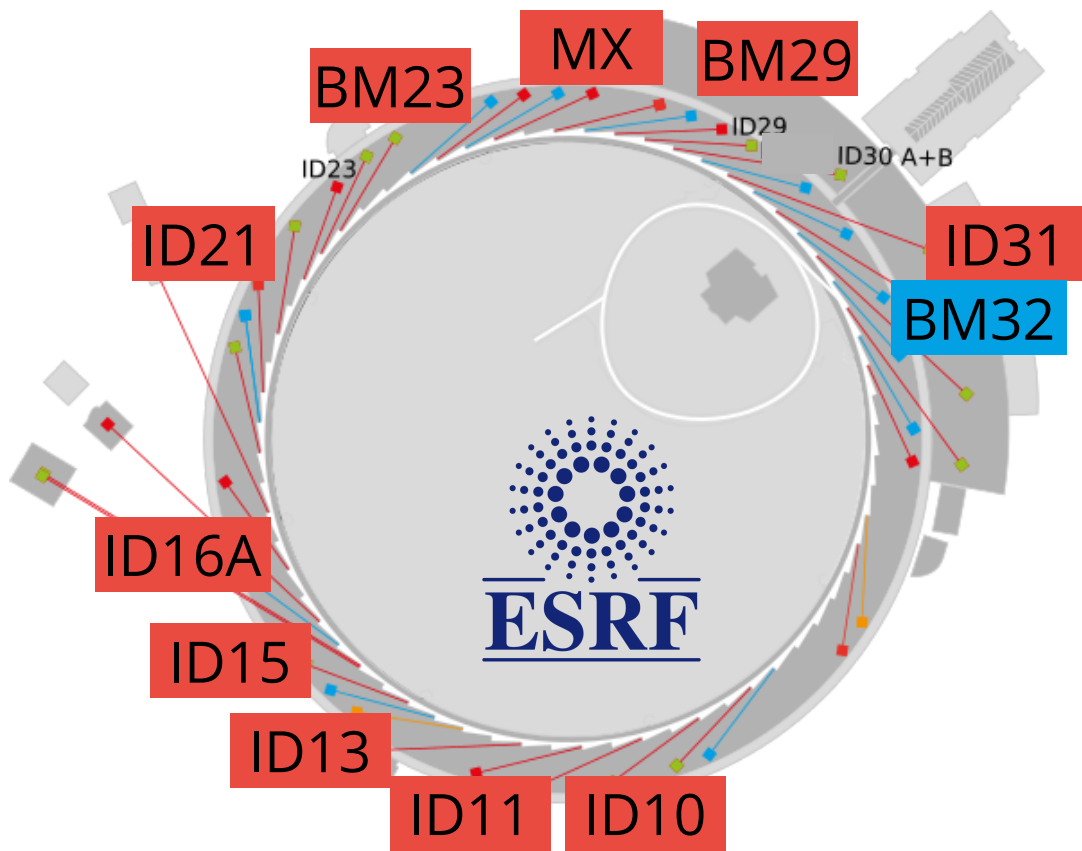


Refactoring helps to fill the holes,
but with time penalty

BLISS deployment (2020)



BLISS deployment (2020)



In 2018, first tests with users: MX, ID10, ID11,

ID13, ID31



Conclusion

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