MXCuBE3 @ ESRF

Daniele de Sanctis
daniele.de_sanctis@esrf.fr
MXCuBE1 was used for a long time at ESRF
Eventually installed at Soleil, MAXLab, Bessy, EMBL-HH
But presented some limitations
  ○ Capacity of sample changer increased
  ○ Microbeam capabilities needed to be exploited
  ○ Complexity of data collection increased
  ○ Hard to install elsewhere, too many ESRF dependencies (spec in primis)

These common needs paved the way to the design of MXCuBE2 and the begin of the MXCuBE collaboration
THE MXCUBE COLLABORATION

2005

2012

2015

2019

On MXCuBE the sun never sets

https://github.com/mxcube

Partners meet twice a year in round robin (jointly with ISPyB)
Collaboration very dynamic both on core and front end
MXCuBE2 was released in May 2012
Designed in (ESRF) FW2 based on Qt3
Eventually deployed at other sites
Ported to Qt4 by EMBL-HH

But in the meantime:
- New generation of pixel detectors
- Higher capacity sample changers
- New tools for automated data collection (workflows)
- More demand for Remote access

Collaboration between ESRF and MAXIV initiated the MXCuBE3 project
Design a new interface in web technology
Preserve a common backend with MXCuBE core (Hardware repository)
MXCuBE3 User Interface (Browser or other client)

MXCuBE3 Web Application layer (server)

BlissFramework
Qt 3/4

Beamline control layer
Hardware and procedure abstraction (Hardware Objects)

Control System and Device servers
(Bliss, SPEC, EPICS, Tine, Tango, Sardana)

React
Redux

Websockets
FROM SCIENTIFIC DRIVERS TO WEB

- High-throughput data collection
  - Fully exploit automation
  - Apt to perform more elaborated data collection for complex experiments
- Adaptable to any hardware and control environment
  - Independent from the underlying control system
- Scalable with time
- Interface with external experiment descriptors BES and Global Phasing Workflows

Why web?
- Allows for a more graphical experience with direct interaction with the samples
- Lighter interface
- Remote by design
- No need of extra software installation on the client side
- Modern technology
- It scales well on any screen size
- Fast to modify and maintain
- Smooth integration with ISPyB
At ESRF Shutdown (December 2018) MXCuBE3 was running on ID29 and ID23-2

ESRF-EBS restarted in complete remote user operation in August 2020

MXCuBE3 used at **ID23-1, ID23-2, ID30A1, ID30A3, ID30B**, (soon again on ID29) for 525 user sessions since the restart
- Directly accessible from any browser from https://mxcube3.esrf.fr
- Remote login with proposal account (soon with personal account)
- When experiment is scheduled and local session allows
- Users from same experiment session can login simultaneously, only one in control
- Users can exchange control without local intervention
MXCuBE3 LAYOUT

Beamline setup

Goniometer motors control
Sample joystick

Data collection queue

User logging

Chat

Shift + dblclick → Move to beam
R+scroll → Rotate spindle
Z+scroll → zoom
Only present pucks are displayed

Smart filtering (name, position, …)

Each card a sample

Sample name retrieved from ISPyB

Data collections performed are shown per each sample
Sample(s) Position, line, grid

Data collection method

This is the basis to build any complex data collection sequence (automatically when combined with workflows)

Point (3D or 2D) Line Grid
ENERGY SCAN

Energy Scan

Path: /data/visitor/15/16/23/eh/1/2021317/RAW_DATA/empty/empty-pin/
Filename:
Subdirectory: empty/empty-pin/
Prefix: pin-empty

Element

Element

Edge

Run Now  Add to Queue
HELICAL

MXCuBE3 (mx1816)

Beamline Actions →

Wavelength: 0.8721 Å
Resolution: 2.497 Å
Transmission: 20.61 %
Flux: 2,500+11 ph/s
Cryo: 99.18 K

Sample:
Path: /data/visitor/mx15/01/02/2021/03/11/Raw_DATA/INS_X24/INS_X24/
Subdirectory:
Prefix: INS_X24
Filename: INS_X24_RUN[2].[IMG]

Acquisition
Oscillation range: 0.0502°
Oscillation start: 91.000°
Exposure time (s): 0.006

Queue Samples (0)
Sample: INS_X24
Queue Collection
Queue Data Collection

Log messages:
Show
Processing

Run Now
Add to Queue
Interleave multiple data collection

Energy
Angle (inverse beam)
Kappa
Mesh data collections methods

Mesh&collect
Mesh
X-ray centring
Successful X-ray centring generate a new centered point
No point

Shoot here, no rotation

Do everything automatic

Same methods are available from Samples for pipeline mode
FUTURE PLANS

- Integrate a new Web based diffraction image viewer
- Continue the improvement in the ergonomics
- Improve Samples list visualisation and results
- Complete implementation of DozorM and MeshBest combined with mesh results
- Port GP workflows to MXCuBE3
- Finalise integration of crystallisation plate screening
- Implement novel SSX data collection methods
- Controller for “liquid” based delivery systems, pumping, mixing
- Fixed targets, new type of mesh on periodic supports

https://github.com/marcus-oscarsson/braggy

Dozor score and png available in ISPyB
Develop a graphical interface for timing mode:
- Defines trigger, offsets and delays
- Be able to retrieve the signals (oscilloscope mode) to monitor the events

Opening for PostDoc for ID29
https://www.esrf.eu/Jobs
MXCuBE is part of a large collaboration
  - on a common control layer and user interface
  - That makes a great effort in sharing generic components that are commonly used (ex. Detectors, Diffractometers)
  - “Standardisation” is part of a iterative process, continue refactoring
MXCuBE3 is the web front end
  - Specifically optimised for remote access
  - Facilitate all kind of MX related experiments by hiding the complexity (not removing it)
  - From user perspective it has a flat learning curve
The home of MXCuBE is https://github.com/mxcube
The MXCuBE meeting are usually opened to observers, feel free to contact if interested
ACKNOWLEDGEMENTS

ESRF - EMBL Joint Structural Biology Group
The MXCuBE collaboration
http://mxcube.github.io/mxcube/

ESRF - Marcus Oscarsson, Antonia Beteva
MAXIV
EMBL
Global Phasing
SOLEIL
BESSY HZB
ALBA
DESY
ELETTRA
LNLS
NSRRC
EXTRA SLIDES
- Built on top of the same beamline control layer as MXCuBE 2 (Hardware Objects)
- Instruments and procedures are implemented as what is called Hardware Objects
- The beamline control layer is control system agnostic and supports for instance SPEC, EPICS, Sardana, BLISS and TANGO
- Base classes define a common API for a particular instrument or procedure, which facilitates cross site adaptation
Web Service Layer

- **Defines an API** for clients to access the HardwareObjects, and relays events between Hardware Objects and clients (not necessarily a browsers)

- Thin utility layer for providing new **functionality exclusive to MXCuBE 3** and ease access to Hardware Objects

- Websockets, via SocketIO, **used to relay events from backend**

- Implemented on top of a Flask **web server, WSGI container**
• Application written in HTML 5, Javascript 6 (JS6) and CSS

• JS6 gives us the possibility to use reusable components and modules

• Problem, no browser have full JS6 support

Babel allows us to use reusable modules and classes via ES6 syntax (https://babeljs.io/)

ES6 Code is “transpiled” with babel to ES5 which have good support in most browsers
React is a library for creating user interfaces.

React makes it possible to use widgets like in traditional UI development.

Provides a way to express the UI in a markup language called JSX.

Can be used with state management library, in order to avoid per widget state.

https://facebook.github.io/react/
Application wide state, only source of data for components.

The redux store is an immutable data structure and can only be updated (replaced) by a pure function, a reducer.

The reducer function is called by dispatching an action for instance when user interacts with UI.

Provides data flow which is easy to debug.
```javascript
import React from 'react';
import { Button, ButtonGroup, OverlayTrigger, Popover } from 'react-bootstrap';
import './input.css';
import './input.css';

export default class InputSwitch extends React.Component {
  constructor(props) {
    super(props);
    this.setInput = this.setInput.bind(this);
    this.setOutput = this.setOutput.bind(this);
  }

  setInput() {
    if (this.props.undate == undefined) {
      this.props.undate = this.props.value ? 'in' : 'out';
    }
  }

  setOutput() {
    if (this.props.undate == undefined) {
      this.props.undate = this.props.value ? 'out' : 'in';
    }
  }

  shouldComponentUpdate(nextProps) {
    return nextProps.data !== this.props.data;
  }

  render() {
    const inputStyle = this.props.data === 'in' ? 'default' : 'success';
    const buttonTextStyle = this.props.value ? 'default' : 'success';
    let msgTitleStyle = 'input-bg-moving';

    if (this.props.data === 'in') {
      msgTitleStyle = 'input-bg-moving';
    } else if (this.props.data === 'out') {
      msgTitleStyle = 'input-bg-moving';
    }

    return (
      <div class="input-label">
        {this.props.labelText}
      </div>
    )
  }
}
```
- Webpack is used as a build tool to bundle the various assets, JS, CSS, LESS, Fonts and images to a set of static files that can be loaded by the browser.

- Provides a development server with “hot reloading” (changes are automatically built and app updated)

- Runtime for Javascript development provided by node.js
Context based navigation, options depends on selection

- Video is streamed as MPEG-1, perhaps adaptive MPEG-4 in the future
- Possibility to select video stream size (particularly useful for remote users)
- With auto scale option