MXCuBE 3 web application, on the way to next generation experiment control

M. Eguiraun[1], A. Milan-Otero [1], F. Bolmsten[1], J. Nan[1] , M. Thunnissen[1] , V. Hardion[1], D. Spruce[1] , M. Guijarro[2] , M. Oscarsson[2] , A. Beteva[2] , D. de Sanctis[2], G. Leonard[2], J. Meyer[2], A. Gotz[2]

[1] MAX IV Laboratory, Fotongatan 2, 225 94 Lund, Sweden.

[2] ESRF The European Synchrotron, 71 Av des Martyrs, 38000 Grenoble, France.

E-mail: mikel.eguiraun@maxiv.lu.se

Abstract.

Macromolecular Xtallography Customized Beamline Environment (MXCuBE) is a software platform that provides users of beamlines at synchrotrons an easy to use graphical environment. From one side it hides the complexity of the beamline hardware, facilitating normal operation, while on the other side provides routines for automated complex data collection strategies. The third evolution of this software is under development as part of the MXCuBE collaboration. A prerelease version has already been used in experiments at MAXIV and ESRF, the facilities leading the development. The main evolution compared to the previous versions is the transition to a web based environment, which is expected to facilitate remote data collection and on-line data analysis, among other things. This article explains the main features and technical details of MXCuBE v3.

1. Introduction

The MXCuBE project started in 2005 at ESRF [1, 2], with the objective of providing a unified and user-friendly software to the macromolecular crystallography (MX) beamlines. The software was designed to pace the increase in performance and automation at the MX ESRF beamlines. From the initial task of collecting single crystal diffraction data, MXCuBE evolved to include more advanced functionalities, enabling the assessment of diffraction characteristics of samples, complex and automatic data collection, and recording X- ray emission spectra and subsequent analysis. Furthermore it offered the possibility to control the beamline remotely; directly from the home laboratory (Remote Access).

In 2010, a collaboration for the development of MXCuBE started among the major synchrotron facilities in Europe with the aim of further developing MXCuBE. Today it is actively supported by the following partners: ESRF, Soleil, MAX IV, HZB, EMBL, Global Phasing Ltd, DESY and ALBA. And several new partners are considering to join the collaboration.

The current stable version of MXCuBE (release v 2.2) is developed in Python using the PyQt toolkit and nowadays represents the state-of-the art in terms of GUI for MX diffraction experiments. It enables the implementation of new data collection methods [3, 4] and hands-off automatic data collection. However, the continuous evolution of structural biology beamlines and data collection protocols necessitated the development of novel control software, that not

only could keep up both with scientific drivers and user needs, but that could evolve with new technology. From this basis the project of MXCuBE v3 started.

One of the main reasons to implement MXCuBE 3 as a web application is to take advantage of the recent advances in user interface design coming from web technologies. As any web application, MXCuBE 3 runs in a web browser making maintenance and deployment of the client easy and facilitates user access. The distributed nature of web applications along with simple client installation makes MXCuBE 3 ideal for remote access usage. MXCuBE already supports the web-based LIMS (Laboratory Information Management System) for protein crystallography, ISPyB [5]. A seamless integration can further be achieved thanks to the use of a common platform, resulting possibly in a better performance and smoother operation. Taking advantage of this new technology the user interface has been completely redesigned, with the aim of enhancing the user experience by means of a feedback gathering from users, an improved experiment queue operation and sample management, and a better integration with LIMS system.

The main technologies in use are python-flask web framework [6] for tL he backend, and React JavaScript library [7] for the front-end, enhanced with several third party libraries for both components. Low-level control is achieved via Tango, Sardana and custom protocols, by means of the so-called *Hardware objects*, which are self-contained pieces of software which links to the underlying instrumentation control. These libraries are being reused from the previous versions of MXCuBE, hence compatibility between different versions is guaranteed.

MXCuBE v3 development is currently lead by MAXIV and ESRF. The first milestones defined in 2015 have already been achieved, when the first data collection experiment was successfully performed in June 2016 during MAX IV new MX beamline Biomax [8] commissioning. The next months will be dedicated to increase the stability of the application and to add new features that will be needed when the user operation starts. MXCuBE v3 will not only be the experiment control environment at Biomax beamline in MAXIV and the ESRF MX beamlines in the near future, but the experience acquired will serve for future software developments.

The structure of this paper is as follows: first the MXCuBE v3 development plan and the main goals of the project are described, the next section deals with the technologies adopted, and finally some conclusions will be listed.

2. MXCuBE v3 Development

A preliminary feasibility study was requested by the MXCuBE collaboration board, and the conclusions and experience gained from that study defined the technology stack to be used for the development. The real kick-off of the project happened in September 2015, at that time, MAXIV and ESRF defined together the development plan, specifying the main features and milestones, in addition, each milestone was subdivided in a set of work packages.

MAX IV decided to directly use MXCuBE v3 for the BioMax beamline from the very beginning of the beamline operation. Although this might look like a risky approach, the fact that the schedule of the installation of the beamline components was distributed over several months, and the commitment of both institutes allowed a successful development. Moreover, the old MaxLab as well as the ESRF beamlines have been used extensively for testing the developments (avoiding disruption of user operation).

In June 2016, shortly before the inauguration of the MAX IV facility, the first diffraction data was collected at the BioMax beamline. For such first experiment, the most important feature to implement was the so called 'Standard Data Collection', which is the most basic data acquisition. It implies configuring the beamline in order to get a suitable photon beam, as well as configuring the data collection by centering the crystal in the beam path and the oscillation to be performed by the diffractometer. For a more comprehensive list and description of the main features and milestones see the github page of the project [9].

Frequent meetings are held for planning and estimating the work, following a kind of distributed scrum (Agile methodology), where the review, planning and estimation of work is done every month. Technical discussions and face to face meetings for specific topics are also organised when needed. The project relies, apart from physical meetings, on Github and Skype for communication between the teams.

3. From Qt to Web

Although making MXCuBE available for the web is a major development effort, the whole existing Hardware Object layer is kept unchanged and is common among the different versions. In this way the application reuses the existing libraries for beamline control. The focus is on bringing the user interface to the web making the transition from MxCuBE v2 to MxCuBE v3 straightforward.

This section explains the main characteristics of the Hardware Objects library, as well as the main technologies in use in MXCuBE v3.

3.1. Hardware Objects

The MXCuBE Hardware Repository holds the description of the devices and equipments of the beamline as a set of XML files. Each file represents at least one Hardware Object, which is a self-contained piece of software that links the graphical layer to the underlying instrumentation control software. The most common communication protocols for accelerator control are available, e.g. Tango, EPICS, Tine, Sardana, Spec or EMBL Exporter. Hardware Objects can be composed in order to represent more complex equipment, like a diffractometer. And and they can also call methods from another Hardware Object, for example an object that handles the configuration of the beamline could retrieve motor positions from another Hardware Object. The asynchronous communication has been implemented using the concept of signals and slots.

3.2. User Interface

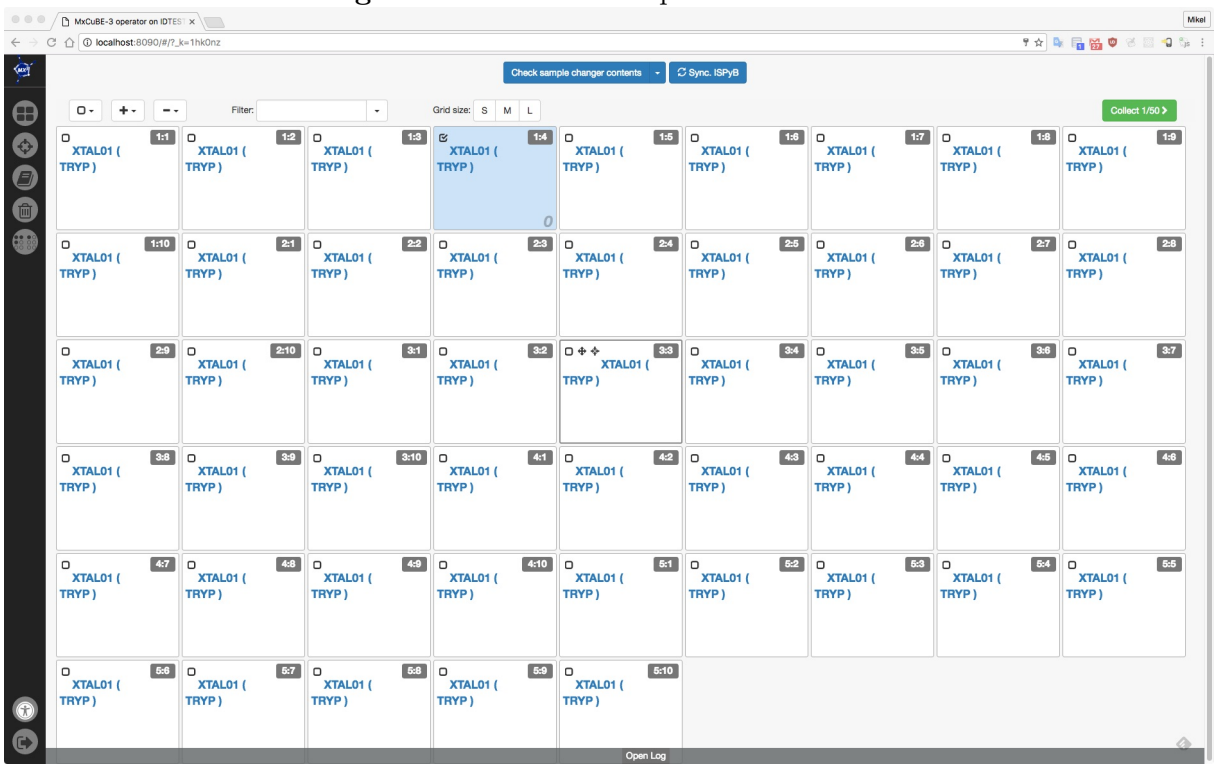
Since this new development moves from using Qt libraries into a web environment one of the first decisions (and discussions) was the user interface. The previous interfaces fulfil very well their mission, however a questionnaire was sent to nearly two hundred beamline users in order to detect any potential usability problems as well as to listen to suggestions. The overall feedback was positive, however there were clear needs to make improvements, such as the usability considering different user profiles, but also the advanced configuration and management of the experiment.

There are two main modes of operation with MXCuBE: automatic and manual mode. The first one consists of selecting a set of samples and selecting the appropriate workflow that will be applied to all of the samples, i.e. an automated, complex and sequential predefined data collection. For this purpose, the interface should present an easy way of inspecting all available samples, and associate a workflow only to the ones the user is interested in.

On the other hand, in a manual operation the scientist usually have a close view of the sample, he/she manipulates the diffractometer, centres the sample and manually selects the most appropriate parameters for the data collection. Several collection types are also available, and characterisation routines could give the user an initial proposal for the diffraction experiment. In any case, one could pre-configure all the experiment and then manually proceed with each sample.

Hence, since there are two main operation modes, the interface layout is split into different views, decoupling the sample lists and experiment configuration, see Figure 1, from the sample view. The new interface is intended to cope with increased sample throughput (several hundreds of samples each day), representing samples as cards, which is considered more appropriate compared to a hierarchical information display in a big tree as it was before. A card would allow

Figure 1. MXCuBE Sample Grid Interface.



to condensate information of sample details and performed analysis in a small space, and offers more possibilities regarding user interaction.

The Figure 2 shows the sample view together with the experiment tree for the current sample, basically the one that is mounted and a glance of what is coming. Ideally, the switching back and forth between the two views should be minimised, additionally, the design should avoid duplication of information whenever is possible. At the present stage of development the two abovementioned views have been implemented, which provide the most basic and important capabilities needed in the beamline operations. Additional views are currently in development.

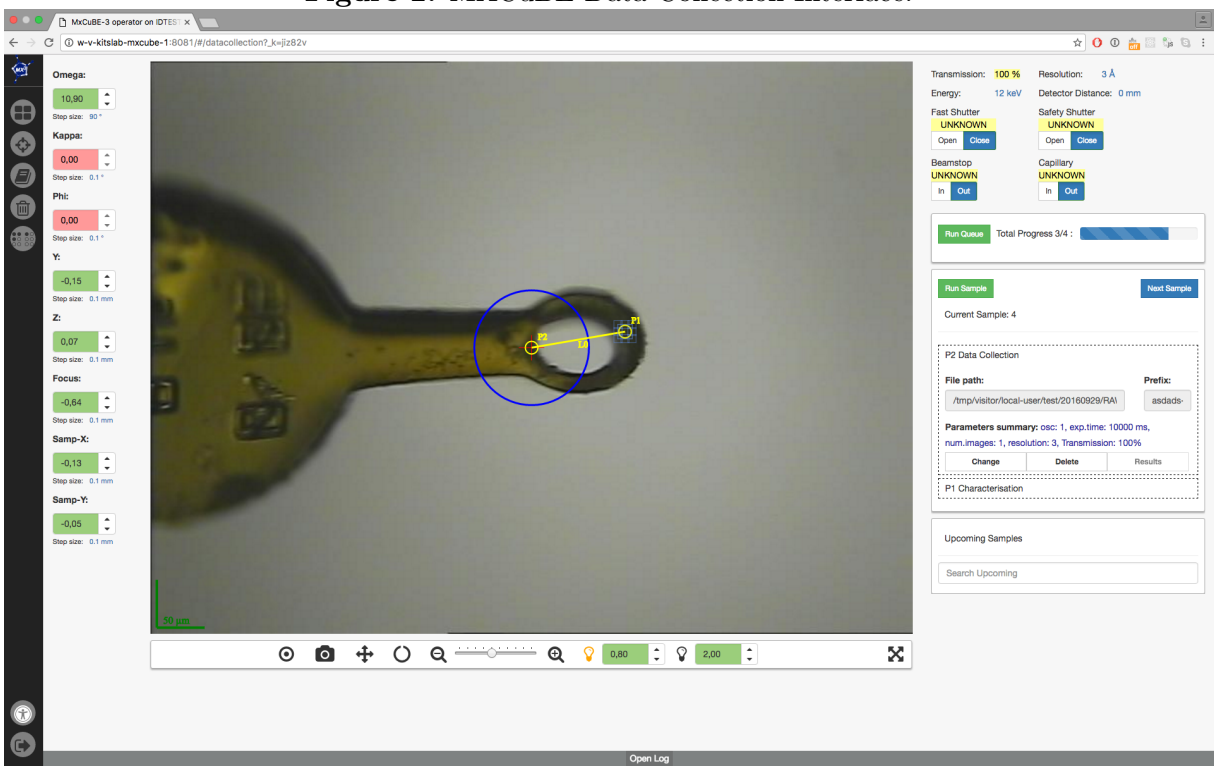
3.3. Backend

The backend server is based on Python Flask web server [6]. It is a Web Server Gateway Interface (WSGI) *microwebframework*, in which the core of the sever is designed to be simple but extensible, thus the developer needs to implement many services commonly found in heavy web servers. There are many flask extensions available, for example for dealing with databases, user authentication, form validation, templates, and so on. The backend also uses the gevent-flask extension to enable asynchronous network communication, in order to respond to concurrent requests in a very efficient manner.

Flask has been proven to be simple and effective to use, and integration within any WSGI-compliant application container can be done with little effort. Although at the moment, in the development phase, the Flask builtin web server is used.

The backend has been implemented following a Rest-like HTTP API calls. Where calls requested by the client are made via the standard HTTP methods (GET, POST, PUT, DELETE), and in the response apart from the result of the operation some data is included.. The different URLs are defined in a way that the functionality is easily understandable. For bidirectional asynchronous communication, MXCuBE v3 uses *socketio*, which is a library

Figure 2. MXCuBE Data Collection Interface.



for communication that adapts automatically to the available protocols to ensure the best functionality, going from Websockets to AJAX polling. In the current state of the development, socketio is mainly used for re-emitting the hardware objects internal signals to the client, for example when a data collection has finished its execution.

3.4. Front end

In order to provide an enhanced user experience the front end has been completely redesigned. For that goal we rely the development on React Javascript library, which it is a library that takes advantage of a component-based design.

3.4.1. React React is an open-source Javascript library for building user interfaces using the concept of components [7]. Components make it possible to create independent entities encapsulating functionality, much like widgets in a traditional desktop UI framework. React is mainly concerned with the View part in the classic Model View Controller (MVC) design pattern and for more complicated applications another library needs to take care of the Model and Controller in MVC. It has fast become one of the most popular way to achieve a single page application (SPA).

In React the UI is described as a collection of components, where each component encapsulates code making it easier to reuse and test than in normal web-development. The main objective of a component is to provide a view for rendering data as HTML. The development of React was started by Facebook and is used both in Facebook and Instagram but was open-sourced in 2013 and is now used by many other websites.

Redux Redux is an open-source Javascript library that handles the state of an application, [10]. It is one of the most common libraries to use with React when developing complex applications because it takes care of both the Model and Controller (in the MVC pattern) and is therefore a good fit with React. It does this by keeping the state of the application in one place and only letting actions mutate it. By using this architecture, it is possible to get logging, hot reloading and time travel for debugging purposes with minimal effort.

One of the drawbacks of React is the fact that it does not recommend direct component to component communication, but it does not provide a solution either. Redux solves the problem by storing all your application state in one place, called a *store*. React components then dispatch state changes to the store, not directly to other components. An action can be triggered from the components themselves or from an outside source such as the server. Once the state is changed, it is passed down to the components of React and rendered as HTML. This unidirectional data flow makes the application much more predictable and easier to understand.

4. Conclusions

The existing and successful MXCuBE collaboration has been strengthened with the development of MXCuBE v3. New tools and procedures have been established by different partners with the common goal of providing good software for our users. The current stage of MXCuBE has already prove that it is in good rails with real experiments and helping with BioMax beamline commissioning. A lot of effort has been devoted to proper planning and technology analysis, however, there have been changes compared to what it was planned in 2015, but the MXCuBE team is really committed with the project and quickly adapts to changes. Furthermore, currently both Qt and Web versions of MXCuBE are in development, and this has lead to a very positive feedback where new features and ideas have been shared. The next months are devoted to the addition of the features in order to be ready for welcoming users, as well as to continue improving the current status of the software.

Acknowledgments

The authors are very grateful to the KITS and MX (MAXIV) and BCU and SB (ESRF) teams for their outstanding support during the development.

References

[1] Gabadinho J. et al. MxCuBE: a synchrotron beamline control environment customized for macromolecular crystallography experiments. J Synchrotron Radiation. September 2010, doi: 10.1107/S0909049510020005.
[2] MXCuBE project page, http://mxcube.github.io/mxcube/
[3] Zander et al. MeshAndCollect: an automated multi-crystal data-collection workflow for synchrotron macromolecular crystallography beamlines. Acta Crystallographica Sec. D, Vol. 71, Nov. 2015.
[4] D. de Sanctis et al., Facilitating best practices in collecting anomalous scattering data for de novo structure solution at the ESRF Structural Biology Beamlines. Acta crystallographica. Section D, Structural biology, 2016; 72 (Pt 3) doi:10.1107/S2059798316001042
[5] S. Delageniere et al., ISPyB: an information management system for synchrotron macromolecular crystallography. Bioinformatics. 2011 Nov 15; doi: 10.1093/bioinformatics/btr535.
[6] Python Flask web framework, http://flask.pocoo.org/
[7] React: a javascript library for building user interfaces, https://facebook.github.io/react/
[8] Thunnissen M. et al. The macromolecular crystallography beamlines BioMAX and MicroMAX at the MAX IV laboratory. Acta Crystallographica Section A: Foundations and Advances. 2015.
[9] MXCuBE v3 in github, https://github.com/mxcube/mxcube3
[10] Redux, a predictable state container for JavaScript apps. http://redux.js.org/