ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-TUBPL05

MXCuBE3 - BRINGING MX EXPERIMENTS TO THE WEB

M. Oskarsson, M. Guijarro, D. de Sanctis, A. Beteva, G. Leonard, ESRF The European Synchrotron, Grenoble, France
M. Eguiraun, J. Nan, F. Bolmsten, A. Milan-Otero, M.Thunnissen, MAX IV Laboratory, Lund, Sweden

Abstract
Originally conceived at ESRF and first deployed in 2005 MXCuBE, Macromolecular Xtallography Customized Beamline Environment, has with its successor MXCuBE2, become a successful international collaboration. The aim of the collaboration is to develop a beamline control application for macromolecular crystallography (MX) that are independent of underlying instrument control software and thus deployable at the MX beamlines of any synchrotron source. The continued evolution of the functionality offered at MX beamlines is to a large extent facilitated by active software development. New demands and advances in technology have led to the development of a new version of MXCuBE, MXCuBE3, The design of which was inspired by the results of a technical pre-study and user survey. MXCuBE3 takes advantage of the recent development in web technologies such as React and Redux to create an intuitive and user friendly application. The access to the application from any web browser further simplifies the operation and natively facilitates the execution of remote experiments.

INTRODUCTION

MXCuBE is a control software developed for Macromolecular Crystallography beamlines. Since its first version, deployed in 2005 at The European Synchrotron (ESRF), MXCuBE has continuously evolved to facilitate user experiments by hiding the complexity of the beamline hardware and low-level control environment. In 2013 MXCuBE2 succeeded the original version and became the core of an international collaboration. MXCuBE2 was completely redesigned to permit the operation of new hardware, with particular relevance given to sample changer robots and new generation X-ray detectors. The development further enabled the automation of MX beamlines and maximised sample throughput [1]. Since then, the constant evolution of MX data collection methods and the increasing popularity of remote data collection, created new demands on the control and acquisition software, that converged into a new graphical user interface. This paper presents the architecture, preliminary usage feedback, and experiences learned from the development of the next version of MXCuBE, version 3. MXCuBE3 takes advantage of the recent development in web technologies such as ECMAScript 6 (ES6), React and Redux which the authors believe provides an environment suitable for implementing a complex web application such as MXCuBE3.

ORGANIZATION

The MXCuBE collaboration currently consist of eight institutes (ESRF, Soleil, MAX IV, HZB, EMBL, Global Phasing Ltd., DESY and ALBA) actively developing the software. A few further institutes are currently considering becoming full members of the collaboration. Developers and scientists meet in joint scientific and development workshops twice every year to share their respective progress and agree on the future goals of the collaboration. The collaboration enhances and speeds up the development of MXCuBE, many sites share similar needs and instruments and can thus quickly adapt to already existing solutions. Users of all MXCuBE sites are further presented with a familiar user interface, which decreases the learning curve and increases the portability of experiments across the different facilities.

A Memorandum of Understanding (MoU) has been signed by the partners in the collaboration. The collaboration comprises steering-, scientific- and a developer's committees. The development of MXCuBE3 was promoted and supported by the collaboration, however the development and deployment effort for MXCuBE3 is being jointly made by ESRF and MAX IV. Development started in September 2015 with a comprehensive roadmap and specific goals established during an initial meeting. The project uses an agile, scrum like, process. Planning meetings are held every two weeks and development workshops every three months. The project is available on GitHub [2].

BACKGROUND

MXCuBE 2 has become the leading software used to collect data for MX-experiments at European synchrotrons [3]. As the MXCuBE project is an international collaboration and the software is used on different sites across the world, the aim of the project is to provide user friendly software which is easy to adapt to various control systems and hardware environments.

The evolution of MX beamlines, the increase in sample throughput and the introduction of new collection procedures, have introduced new demands on the software [4-7]. The application and its interface have grown increasingly more complex to be able to handle the new requirements, as a side effect decreasing usability of the application [8]. At the same time key software libraries used in the User Interface (UI) are getting outdated and difficult to maintain. Efforts to update these key libraries have been made and a version of MXCuBE with more recent UI libraries is available [9]. However the main issue of keeping pace with these software libraries and maintaining the environment on which they run remain. As software technology evolve new libraries and methods for developing UIs have emerged that the authors believe could alleviate some of these issues.

The MXCuBE community have during this time therefore been discussing the requirements of, version 3 of MXCuBE. The ESRF Structural Biology group surveyed the user community to get their opinion on features they would like to improve or add. This was used to define the key features of MXCuBE3:

- Facilitating maintenance and installation of both client and server
- Support for new sample changers and larger sample quantities
- Improve the integration with LIMS database like ISPyB
- Enhancing the possibility of integration with third party data collection strategy calculation software
- Improve overall user experience, using a more recent user interface design
- A scalable interface that adapts better to available screen sizes
- Improve remote access performance

MXCuBE3 as a Web Application

Web application technologies have matured as streaming services for video and audio such as, Netflix, Spotify and Deezer have become increasingly popular. Companies like Facebook and Google have introduced libraries for frontend development like React and AngluarJS that further facilitate the transition to a web based environment.

One of the major advantages with web applications and perhaps the reason for the popularity of the services mentioned above is the ease of access. A web application can easily be accessed from almost any computer or mobile device without any additional software installation required. Bringing MXCuBE to the web facilitates seamless integration with already existing web based services like the LIMS system ISPyB. A key feature of MXCuBE is the feature, where a user logs in and performs an experiment remotely. Staff or other users can, depending on their access rights, login to the beamline but only to observe what is being done or communicate, via a chat mechanism, with the user in control. Implementing MXCuBE as a web application enables the remote access feature of MXCuBE almost by design. MXCuBE3 can simply be rendered in a browser with native elements directly on the client, whereas other means of remote access have to rely on specific remote access applications and compression schemas (i.e. NX-Client) [10]. The authors believe that this will greatly improve performance and ease of access to the application.

BACKEND

The backend of MXCuBE has been divided into two layers, beamline control and web service. The beamline control layer provides access to beamline instrumentation and procedures via HardwareObjects (HO) [11-12]. MXCuBE2 already introduced a clear separation between user interface and beamline control, following MVC design pattern principles. As for MXCuBE2, in MXCuBE3 the beamline control layer is implemented by Hardware Objects. In order to speed up development by reducing regression and facilitate testing, the same version of HO is used as in MXCuBE2. The Web service layer implements a REST API to the beamline control layer that is used by the client to perform the various operations requested by the user.

Beamline Control Layer

The beamline control layer consists of a set of Hardware Objects which implement a device or procedure. Hardware Objects are Python classes configured by an eXtensible Markup Language (XML) configuration file. Each configuration file contains the information necessary to initialize the corresponding Hardware Object. The Hardware Objects provide access to beamline instrumentation through the beamline control system and implement the higher-level operations required by MXCuBE. The beamline control layer provides an abstraction for control systems such as SPEC, EPICS, TINE, TANGO and Sardana, Hardware Objects are in this way control system agnostic.

Hardware Objects support composition, to represent more complex entities of hardware. In this case, the configuration XML file defines references between objects. A referenced Hardware Object is associated with a “role”, which is used to access the corresponding instance. Hardware Objects are singleton objects thus only one instance of each Hardware Object is instantiated at the time, and this instance is shared across the entire application.

To create a common instrument and procedure API which permits cross site adaption, each Hardware Object inherits an abstract base class that describe the particular instrument or procedure. The model is flexible enough to be able to adapt to the hardware and beamline specificities of different synchrotron sites with little effort. There are currently over 20 base classes defined in the beamline control layer.

Web Service Layer

The REST service layer, developed in python, provides an API for the clients to the underlying beamline control layer through REST-full web services. These services are implemented on top of Flask, a Web Server Gateway Interface (WSGI) compatible micro web framework. The Flask framework contains the Werkzeug development web server which is provided as the default server for MXCuBE3. However, the application can also be deployed on a WSGI compatible container including Green Unicorn, uWSGI or Apache mod_wsgi. A thin utility layer is used to facilitate the access to the beamline control layer and adds features that are new and exclusively used in MXCuBE3 (see Figure 1).

The main role of the web service layer is to receive calls and relay them to either the client or the beamline control layer it's thus primarily handling IO operations. Considering the IO intensive nature of MXCuBE3 and python's global interpreter lock (GIL), the authors have chosen to use coroutines, provided by Gevent, instead of threads to handle concurrency. Gevent is a coroutine-based networking library that uses greenlet to provide a high-level API on top of the libev event loop.

In MXCuBE3, a bidirectional event broker based on SocketIO was implemented to be able to send asynchronous events from and to the client. SocketIO provides a bidirectional communications channel capable of, if needed, degrading to the protocols available on the connected clients. A special version of Flask, called Flask-SocketIO is used to enable native support for SocketIO within the web framework. A session is used to keep track of information related to a particular user, this session is kept in a Redis data structure store. The Redis data structure is written to disk and can be restored in case of a server fault.

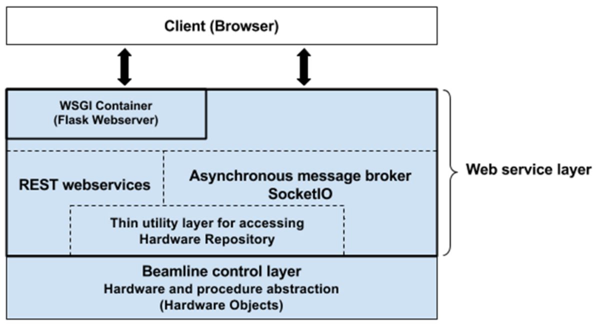

Figure 1: Application architectural overview, with beamline control layer, web service layer and client.

FRONT END

The user interface is written in Javascript, HTML5 and CSS. Javascript provides single threaded event driven programming model which is well suited for UI development [13]. Javascript is an implementation of the ECMAscript standard. Recent versions of ECMAScript have introduced constructs more suitable for writing complex applications than earlier versions. Most notably classes and modules was introduced in ES6, released in 2015, the most recent version is ECMAScript 8 (released in 2017). However the different versions of ECMAscript are supported to varying degree in each browser, browser support for ES6 is still incomplete. ES6 code can be translated, "transpiled", into ECMAScript 5 code which has more consistent support across browsers. A software called Babel is used to perform the translation.

The browser itself does not have a concept of modules or dependencies, so a build tool called webpack is used to package, "bundle", the various files into code that the browser can execute. Webpack handles the various project assets including Javascript, images, fonts, and CSS via loader plugins. The code is bundled and further optimized, "minified" and a set of static files are produced (see Figure 2). Using Babel (ES6) and Webpack makes it possible to define and use components and modules via the export and import statements.

Webpack, Babel and various other third party libraries used for the build process require a Javascript runtime. Node.js is used to provide the runtime for build and development environment. Node.js also provides a package manager npm (node package manager) with very large repository of third party libraries.

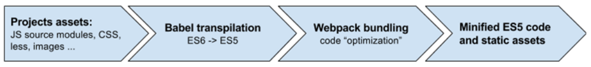

Figure 2: User interface build steps.

A library for user interface development called React provides means to create reusable user interface components. The components enable encapsulation of interaction logic and display similar to widgets in user interface frameworks for desktop applications. The user interface can be expressed in a HTML like syntax called JSX (JavaScript XML) where react components and HTML element can be included as tags. React optimises the rendering of the application and provides component life cycle for each component via a Virtual-DOM. The browser events are replaced with synthetic events within the Virtual DOM which is used to keep track of components that have changed internal state. Batch updates of all components are performed and a minimal set of changes to apply to the original DOM is calculated. This is done to minimise the number of repaint operations that are required to update the user interface.

In many user interface frameworks including React the state of a component is often contained within a data structure tightly coupled to the component itself. This means that the state of the application is distributed over the application and can become difficult to extend and debug. React can be used together with a state management library such as Redux to solve this limitation. Redux makes the state mutations more predictable by defining an application wide state, referred to as a store, and imposing certain restrictions on how the store is updated. The Redux store is an immutable data structure and can only be updated by dispatching an action that describes the state transfer. The state transformation is defined by a pure function called a reducer that takes the current state and the action as parameters and returns the new state. The components listen for changes to the store and are updated accordingly. The user interface is composed by several of these reusable React components that reflect the changes made to the underlying Redux store.

Enabling Remote Access Experiments

In a Remote Access experiment at a synchrotron source a beamline user carries out an experiment from a different location i.e from the user's home laboratory. The samples are stored in the beamline sample automounter by synchrotron staff, and the remote users have control of the endstation once the experiment hutch is interlocked.

Remote Access experiment presents some challenges to the beamline control software:

- how to manage a remote user's access control
- how to communicate with the remote user
- how to follow what a remote user is doing
- how to help remote user in case of problems
- how to take back control over remote user in case of emergency
- how to ensure acceptable real time performance (latency, real-time video and information messages)

One of the main drivers for MXCuBE3 as a web application was the need for an improved remote access functionality. The main points of improvement from previous versions of MXCuBE are:

- A user is not required to install additional software (i.e. NoMachine NX client or similar) to connect to the synchrotron computing system
- Improved real time performance, responsiveness of the user interface and video streaming frame rate.
- User action monitoring to enable remote assistance

As a web application, MXCuBE3 inherently supports multiple clients without requiring the installation of any software apart from a recent web browser on the client side. The UI elements can be directly rendered in the browser decreasing the amount of information that needs to be transferred. Particular effort has been put into the sample video streaming and MXCuBE3 offers the possibility to stream the beamline camera video as MPEG1. Users may need to select different parts of their samples for data collection, requiring a lot of interaction with the video stream. Improving the video streaming will therefore have a big positive impact on remote access experiment performance.

During remote access to the beamline control system the first user that logs into the application with a user account that currently has the right to the beamline becomes what is referred to as a Master. The Master user is the only user that can work on the beamline. Master users have the possibility to grant subsequent logins from the same account to observe what is currently being done on the beamline. Such users become Observers, the Observer users can ask to become Master, if accepted by the current Master the roles switch and the current Master becomes an Observer. This system guarantees that only one user at a time can use the beamline. It also ensures data confidentiality since the Master user has to grant the access for a user to become Observer. The current ongoing procedure is paused if the current Master client loses connection to the system.

MXCuBE3 takes advantages of the application wide state stored in the Redux store, to implement full user interface duplication for Observers. The application wide state of the Master is propagated to the Observers. Actions affecting the Redux store while the Master is using the application are captured and sent to the MXCuBE3 server using redux-persist. The MXCuBE3 server relays these actions to the Observer clients, which update their state accordingly.

User Interface

The user interface has two principal views for setting up and running an experiment, Data Collection and Sample Overview.

The Sample Overview shows the details for a set of samples. Each sample is represented as a card that contains information with the results of the data collections already performed on that sample along with the measurements that will be collected (see Figure 3). The sample information is obtained from ISPyB and synchronised with the sample changer contents. Users can apply a filter on name or location to display a particular sample or subset of samples. This provides the user with an easy way to get an overview of the experiments performed or yet to be done. It further gives the user an easy way to set up data collections on multiple samples at once and run them sequentially in an automatic fashion (referred to as pipeline mode).

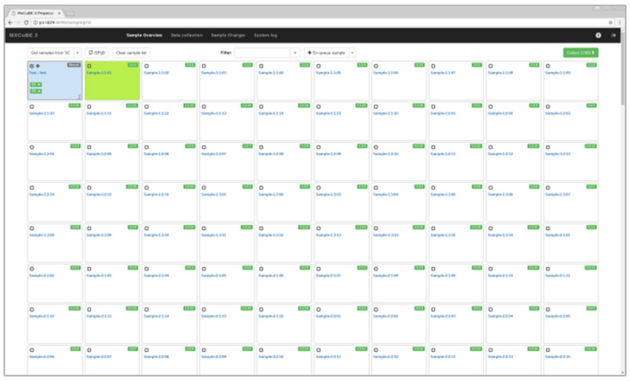

Figure 3: In Sample Overview crystals are represented as cards containing sample data

The Data Collection view provides the user with the necessary controls to perform an interactive data collection or sample realignment. The user can move the sample via the goniometer motor controls, located to the left of the interface, and center the crystal on the rotation axis by using a 3-click centring procedure. The user can further interact with the goniometer motors, to reposition the sample, by using hot keys and clicking in the sample video (see Figure 4). By interacting directly with the sample video, users can access tools to save centered positions, create lines between saved positions or draw a grid to collect diffraction data over an area. The user is presented with context menu listing the available data collection methods available for the type of selection made (point, line or grid). A top-bar displays experimental values of the beamline that define the diffraction experiment, such as energy and detector distance. Data collections on the currently mounted sample are added to a queue shown on the right side and their status is updated while they are performed.

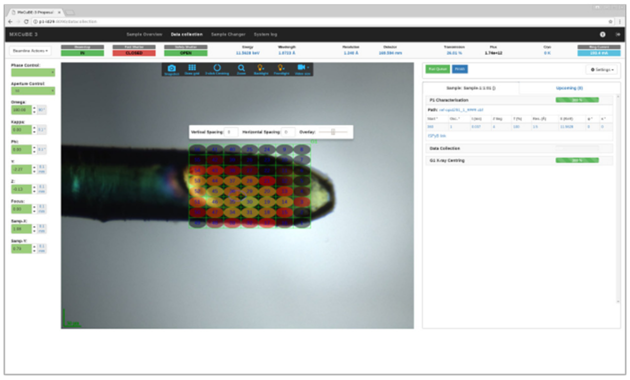

Figure 4: Data collection view, with grid data collection and result overlay, motor controls to the left, and data collection queue to the right.

RESULTS

MXCuBE3 has been actively developed during the last two years. The users experience with MXCuBE2, which has been in operation during the last five years, inspired the design and the functionality of the new interface. A pre-release of MXCuBE3 with a reduced feature set was used both for the commissioning of BioMAX beamline at MAX IV, and in subsequent user operation. First user feedback was very positive and encouraging. Users regarded the interface as much easier to use compared to previous versions of MXCuBE and control software in use at other synchrotron sources. Even less experienced users easily found their way applying the different data collection methods. This commissioning phase and user input have been valuable in subsequent improvement of the user experience.

Another remarkable result is the synergetic effort that two independent facilities were able to invest in the development of MXCuBE3, despite the different short term needs. The project roadmap have evolved from the initial planning to accommodate the new priorities. MXCuBE3 is a successful project thanks to the effort made by the team to keep the communication channels active and proactively collaborate.

CONCLUSION AND FUTURE WORK

In the short term, a feature-complete version of MXCuBE3 will be deployed at the ID29 beamline at the ESRF, then at its other MX beamlines. This is expected to happen during October 2017, and will mark the official release of MXCuBE3. At the same time the MAX IV BioMAX beamline will also upgrade to the latest version.

Deployment on different end-stations is expected to drive the finalisation of the user interface and the implementation of currently available data collection methods, eventually leading to MXCuBE3.1, which is foreseen for the second quarter of 2018.

In the meantime we expect that other partners of the collaboration to profit from the development to install MXCuBE3 at their sites. In a longer term, MXCuBE3 should evolve to integrate more novel data collection methods, in particular in the field of serial crystallography.

REFERENCES

[1] Arzt, S. et al. (2005). Prog. Biophys. Mol. Biol. 89, 124–152. [PubMed]

[2] MXCuBE 3 GitHub, `https://github.com/mxcube/mxcube3`

[3] Daniele de Sanctis, Marcus Oscarsson, Alexander Popov, Olof Svensson, Gordon Leonard Acta Crystallogr D Struct Biol. 2016 Mar 1; 72(Pt 3): 413–420. Published online 2016 Mar 1. doi: 10.1107/S2059798316001042

[4] Zander, U., Bourenkov, G., Popov, A. N., de Sanctis, D., Svensson, O., McCarthy, A. A., Round, E., Gordeliy, V., Mueller-Dieckmann, C. & Leonard, G. A. (2015). Acta Cryst. D71, 2328–2343.

[5] Weinert T, Olieric N, Cheng R, Brünle S, James D, Ozerov D, Gashi D, Vera L, Marsh M, Jaeger K, Dworkowski F, Panepucci E, Basu S, Skopintsev P, Doré AS, Geng T, Cooke RM, Liang M, Prota AE, Panneels V, Nogly P,

Ermler U, Schertler G, Hennig M, Steinmetz MO, Wang M, Standfuss J. Nat Commun. 2017 Sep 14;8(1):542. doi: 10.1038/s41467-017-00630-4

[6] Martin-Garcia JM, Conrad CE, Nelson G, Stander N, Zatsepin NA, Zook J, Zhu L, Geiger J, Chun E, Kissick D, Hilgart MC, Ogata C, Ishchenko A, Nagaratnam N, Roy-Chowdhury S, Coe J, Subramanian G, Schaffer A, James D, Ketwala G, Venugopalan N, Xu S, Corcoran S, Ferguson D, Weierstall U, Spence JCH, Cherezov V, Fromme P, Fischetti RF, Liu W. IUCrJ. 2017 May 24;4(Pt 4):439-454. doi: 10.1107/S205225251700570X. eCollection 2017 Jul 1

[7] Roedig P, Ginn HM, Pakendorf T, Sutton G, Harlos K, Walter TS, Meyer J, Fischer P, Duman R, Vartiainen I, Reime B, Warmer M, Brewster AS, Young ID, Michels-Clark T, Sauter NK, Kotecha A, Kelly J, Rowlands DJ, Sikorsky M, Nelson S, Damiani DS, Alonso-Mori R, Ren J, Fry EE, David C, Stuart DI, Wagner A, Meents A. Nat Methods. 2017 Aug;14(8):805-810. doi: 10.1038/nmeth.4335. Epub 2017 Jun 19.

[8] Uwe Mueller, Marjolein Thunnissen, Jie Nan, Mikel Eguiraun, Fredrick Bolmsten, Antonio Milàn-Otero, Mathias Guijarro, Markus Oscarsson, Daniele de Sanctis & Gordon Leonard (2017) MXCuBE3: A New Era of MX-Beamline Control Begins, Synchrotron Radiation News, 30:1, 22-27, DOI: 10.1080/08940886.2017.1267564

[9] I. Karpics, G. Bourenkov, M. Nikolova, and T. R. Schneider, "Graphical user interface and experiment control software at the MX beamlines at EMBL Hamburg", in *Proc. NOBUGS'16*, Copenhagen, Denmark, October 2016, paper 10.17199/NOBUGS2016.91, pp. 53–58.

[10] NX Technology, `https://en.wikipedia.org/wiki/NX_technology`, accessed 2017 Oct 3

[11] M. Guijarro, Hardware Repository, `https://github.com/mxcube/HardwareRepository`

[12] Gabadinho J, Beteva A, Guijarro M, *et al.*, "MxCuBE: a synchrotron beamline control environment customized for macromolecular crystallography experiments", *Journal of Synchrotron Radiation*, 2010;17(Pt 5):700-707. doi:10.1107/S0909049510020005.

[13] Event Dispathing Thread, `https://en.wikipedia.org/wiki/Event_dispatching_thread`, accessed 2017 Oct z