# Developers’ meeting – discussion summary

# MXCuBE project meeting, Trieste, September 2018

# DRAFT

The developers discussions happened over the refactoring session and the developers’ meeting proper. Action points were agreed in the joint session with the steering committee.

### New conclusions

The main point was that the UI-API is *not* after all enough to serve as the basis of an implementa-tion. The V3 branch is already set up to make use of the API, but the Qt branch would need to know also what behaviour must be triggered by each change. The reason is a hitherto unappreciated difference in architecture: in the V3 branch the entire state of the system (including e.g. whether a collection can currently be started or which sample is currently selected for adding to the queue) is kept on the beamline/hardware side. The user interface is completely dumb and merely reflects the beamline state and accepts commands. In the Qt branch a lot of business logic is done by interaction and signals between UI components (bricks). This logic would have to be extricated from the UI before any change-over to using the UI-API.

### Where are we?

This, of course, reopens the discussion on how to proceed. The goal is still to make MXCuBE easier to develop and maintain, reducing duplication and improving clarity. As Vicente pointed out, we are still at a point where we can make changes that break backwards compatibility and expect all participants to adapt. But there are many levels that need addressing (and, as Mathias pointed out, trying to do everything you risk accomplishing nothing):

* **UI-API**. A specification of which functions can be called by a (dumb) UI and what their effect should be. This would allow the two different UIs to share the same body of underlying code, and would ensure a beneficial separation between user interface code and business logic. It is not tied to using a web technology, but is a simple interface specification.
* **Application layer**. The application layer would incorporate the behaviour of the program: the states of the system, the transitions between them, and include the rules for what signals must be emitted and received to make things happen. Preferably it would allow individual objects to be implemented in isolation, with communication to other objects through well-defined signals. The UI-API would be the interface specification of the UI-facing side of the application layer. The application layer would ideally be shared between all branches and sites. It could be based on a state machine, a server (like it currently is for V3), or a set of objects.
* **Core functionality.** Some central functionality could/should be fully shared, like the application layer, and should maybe be integrated with it. These are blocks like the Session, the Queue, the Beamline-setup/configuration/HardwareObjectRepository system, and the LIMS (ISPyB) connection, code that does not correspond to actual hardware and does not have highly site-specific implementations.
* **Abstract or Generic hardware objects**. It would be an advantage to have uniform function specifications and common behaviours for the main building blocks of a beamline (beam, diffractometer, detector, sample changer, processing). This would give a standard way for other types of objects to communicate, give a central location for common functions, and make it easier to understand and share code from other sites. Site-specific functionality could then be handled in subclasses, or additional classes.
* **Branch merging**. Currently the hardware objects used by the two main branches of MXCuBE (V3 and Qt4) are in separate branches that have drifted apart over several years. Any closer collaboration clearly requires that these be merged. It would be highly desirable to remove unused objects and preferably to merge near-duplicates, so that there is less code to keep track of when making a change, or looking for pre-existing functions.
* **Working procedures**. To get any lasting benefits, people must do the necessary work, use the agreed coding, and avoid splitting into site-specific branches again. Unfortunately clean, documented and shared code is slower to write in the short term, and people tend to work mostly on their own beamline, and under heavy time pressure. The master branch was originally seen as a place to play and try out things; MXCuBE3 avoided it in order to have a relatively stable base for their user interface changes. Others have avoided it because it simply changed too fast. A specific git-based workflow could in theory deal with these problems – if properly followed. Maybe more important would be to build in incentives to get the merging and consultation done at regular intervals. More meetings? Regular releases, which serve to specify what code is officially integrated, and what each beamline is conforming to? Some policing?

It has been a specific problem lately (possibly made worse by the summer season) that pull requests can take a very long time to get accepted and merged. This has likely been worse for the Hamburg group, who have produced a lot of new code, and who do not have an equally active partner group on the same branch that can do mutual vetting. After some discussion (formal time limits, relaxing the requirements for accepting code, …) it was decided to rely on the sense of responsibility of the individual developers, and to make sure that all relevant developers actually have permission to approve pull requests.

### Decisions

The conclusions of the developers’ meeting were presented to the joint meeting with the steering committee. The decision to break off the work on implementing the UI-API was met with surprise, but accepted. The developers’ group was asked for a roadmap to make the necessary changes, but no such roadmap is yet available; what can and should be done is still being discussed. As pointed out by Peter Keller there is an element of discovery in this that does not lend itself to precise roadmaps.

* The first action, as decided by the developers’ meeting, is to merge the different hardware objects branches into a single shared code branch. This, it was estimated, would be a clearly feasible task. The decision was approved by the steering committee.
* It was decided to execute this merge at a face-to-face developers’ meeting, to take place as soon as practicable. The meeting is scheduled for 15-16 November at ESRF Grenoble.
* It was further decided that the series of monthly developers’ web meetings had had a positive effect and should be continued.
* The next MXCuBE meeting will be held at MaxLab in Lund in the week 19-24 March
* The developers’ group will consider how to proceed with harmonisation / refactoring / code improvement (as discussed under ‘Where are we’, above), with particular attention to the development of an application layer, and to future collaboration procedures. The developers’ group must present a plan, together with a roadmap and timing estimates to the steering committee no later than one month before the next MXCuBE meeting. This timing requires homework to be done *before* themid-November face-to-face meeting, so that substantial discussions can start at the meeting.
* There is a special need for documentation, especially documentation that allows new entrants to set up MXCuBE, as well as to start coding. It was decided that new member groups and people doing such setting-up should be tasked with putting their questions in writing, and gathering the answers in a central location, for instance a Github Wiki, in order to build a manual. Groups that had recently done a lot of setting up, such as Elettra, should start by organising and contributing their existing notes.
* Rasmus Fogh (ACTION) is tasked with writing an executive summary of the work on the UI-API for the steering committee.