MXCuBE developers' meeting 22 March 2021

Participants:

- Marcus Oscarsson, Antonia Beteva, Jean-Baptiste Florial, Daniele de Sanctis (ESRF)
- Meghdad Yazdi, Elimir Yagudin (MAX IV)
- Ivars Karpics (EMBL Hamburg) – observer only
- Jordi Andreu (ALBA).
- Lais do Carmo (LNLS)
- Rasmus Fogh (Global Phasing)

Minutes: Rasmus

Last meeting

Minutes approved

Additions to agenda

no additions

Site status

The site status overview was skipped at this meeting

mxcubecore refactoring

The new PR by MO, merges in the renaming PR by IK and otherwise consists in copying the HardwareRepository to a new location, and renaming the old HardwareRepository to mxcubecore. This will require some less usual git commands (like filterbranch and forced push) but has the advantage to preserve issues, history etc. That is the **only** thing happening in PR #613. The developers note that this kind of operations are bad practice in general, but justified and absolutely necessary in this specific case.

Semantic versioning and releases

It is agreed that the mxcubecore repository will follow semantic, 3-digit versioning in the future. The implications were discussed.
MXCuBE is unusual in that there are few direct users, all of whom have their own separate development branch, and all of whom participate in the developers' meetings. The main way of working is a matter of merging master branch advances with local changes, rather than adopting a new 'release'. This means that the most common branch-and-release patterns may be less appropriate in this case. It was discussed whether to move to a system with stable and development branches and regular, official releases. The counterargument was that this would require a lot of backports that might not be done in practice, and that there could be heavy pressure on synchrotrons to make local changes very fast. JA underscored the necessity of ensuring **backwards compatibility** when making changes, so that it would be simple to upgrade versions. In the end the following system was agreed:

1. Major version change. Required for all breaking changes to the API (though not for breaking changes in site-specific code). Major version changes, and consequently breaking change, would require extensive discussion and agreement, and should happen every few years at most.
2. Minor (feature) version change. This would be required for anything that added or changed API capabilities. Note that new functions in the API would need a default implementation, as new abstract methods would break the API. New parameters would need a default value. Feature version changes **must** be agreed by the monthly developers' meeting, and pull requests should be marked if they require a version bump. Multiple feature changes could be grouped into a single version bump.
3. Patch level change. These are for bug fixes and minor modifications (including larger site-specific modifications). This was not discussed in detail; one possibility would be to bump the patch level any time a (group of) pull requests were merged.

Site-specific development would be free to run in its own branch, but it is strongly recommended that new versions of the main branch should be merged in regularly.

There seems to be agreement that the very first version, to be defined Real Soon Now, would be 0.0.1, since the refactoring of the queue system clearly presages further breaking changes before we can reach version 1.x.y

ACTION: JA agrees to write a draft procedure description (which may supersede these minutes) and put it on Googledoc.

Queue refactoring

The meeting discussed the queue refactoring proposal, somewhat hampered by the lack of IK as an active participant – and by the fact that very few had read the discussion document beforehand.

The discussion can be summarised (based mainly on MO’s overview and JA’s summing-up): For each Queue entry there should be a data model object (MO ‘data model’), specified by a meta description (MO: ‘schema’). There were some minor objections to the terminology, particular ‘data model’. It was agreed that each queue entry should have one specification/schema, with no need for subclassing or composition at this level. There should be the possibility of e.g. site-specific implementations to add attributes, which amounts to subtyping. It was proposed that we might use a framework like Marshmallow or Pydantic to generate object code from a schema description. The distinction was made between the schema specification and the specification of default values. There was no agreement whether the data model objects would need to be mutable, or indeed whether a single object of each type might often suffice.

Miscellaneous

AB raised the proposal to get rid of some unnecessary abstract classes, notably AbstractFlux, whereas RF claimed that AbstractFlux was very necessary indeed. The issue was postponed to discussion elsewhere, e.g. as a github issue.

Next Meeting

The date for the next developers’ meeting was left for MO to organise, towards the end of April.