Implementing mesh scan, interleaved collection and other potential methods in MXCuBE

Ivars Karpičs, EMBL-HH

EMBL

Content

- MeshScan and XrayCentring: A case study
- Interleaved collection
- AbstractCollect
- Weakness and strangeness of the MXCuBE data model

MeshScan and XrayCentring

- Previous solution with AdvancedGroupQueueEntry was withdrawn due to the hard synchronization between the execution steps.
- All code removed from master branch (not all ideas live).
- A more straight forward implementation is suggested:

1. Define methods in beamline-setup.xml

```
<!-- advanced methods are defined as a list with method names.
     Each name is then converted to class name.
     For example Mesh scan -> MeshScan, Xray centring -> XrayCentring
     and used as a queue_entry. If queue entry is missing
     then queue skip exception will be raised.
-->
<advancedMethods>["MeshScan", "XrayCentering"]</advancedMethods>
```

2. Define how the queue entry model is created (Qt4_create_advanced_widget.py)

```
dc = queue_model_objects.DataCollection([acq],
                         sample.crystals[0],
                         processing_parameters)

dc.set_name(acq.path_template.get_prefix())
dc.set_number(acq.path_template.run_number)
dc.set_experiment_type(EXPERIMENT_TYPE.MESH)
dc.set_requires_centring(False)
dc.grid = grid

exp_type = str(self._advanced_methods_widget.\
    method_combo.currentText())
if exp_type == "MeshScan":
    dc.run_processing_parallel = "MeshScan"
    tasks.append(dc)
elif exp_type == "XrayCentering":
    xray_centering = queue_model_objects.XrayCentering(\
       dc, sample.crystals[0])
    dc.run_processing_parallel = "XrayCentering"
    tasks.append(xray_centering)
self._path_template.run_number += 1

return tasks
```

Implementing MeshScan and XrayCentring

3. Define class in queue_model_objects.py:
- MeshScan is DataCollection with MeshScan exp type.
- For others new class needs to be created:

```
class XrayCentering(TaskNode):
    def __init__(self, ref_data_collection=None, crystal=None):
        TaskNode.__init__(self)

        self.set_requires_centring(False)
        if not ref_data_collection:
            ref_data_collection = DataCollection()

        if not crystal:
            crystal = Crystal()

        self.reference_image_collection = ref_data_collection
        self.crystal = crystal
```

4. Define class in queue_entry.py (most important part):

```
class XrayCenteringQueueEntry(BaseQueueEntry):
    """
    Defines the behaviour of an Advanced scan
    """
    def __init__(self, view=None, data_model=None,
                 view_set_queue_entry=True):

        BaseQueueEntry.__init__(self, view, data_model, view_set_queue_entry)
        self.mesh_qe = None
        self.helical_qe = None
        self.in_queue = False

    def execute(self):
        BaseQueueEntry.execute(self)

    def pre_execute(self):
        BaseQueueEntry.pre_execute(self)
        mesh = self.get_data_model()
```

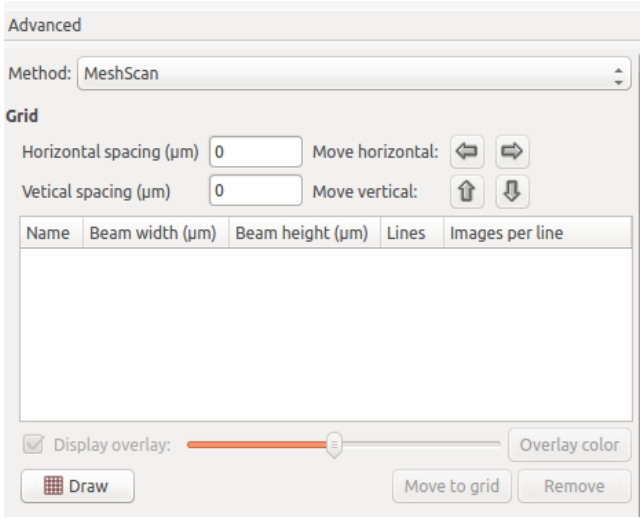
```
MODEL_QUEUE_ENTRY_MAPPINGS = \
    {queue_model_objects.DataCollection: DataCollectionQueueEntry,
     queue_model_objects.Characterisation: CharacterisationGroupQueueEntry,
     queue_model_objects.EnergyScan: EnergyScanQueueEntry,
     queue_model_objects.XRFSpectrum: XRFSpectrumQueueEntry,
     queue_model_objects.SampleCentring: SampleCentringQueueEntry,
     queue_model_objects.Sample: SampleQueueEntry,
     queue_model_objects.Basket: BasketQueueEntry,
     queue_model_objects.TaskGroup: TaskGroupQueueEntry,
     queue_model_objects.Workflow: GenericWorkflowQueueEntry,
     queue_model_objects.XrayCentering: XrayCenteringQueueEntry}
```

Implementing MeshScan and XrayCentring

5. Define class in Qt4_queue_entry.py:

```
class XrayCenteringQueueItem(TaskQueueItem):
    def __init__(self, *args, **kwargs):
        TaskQueueItem.__init__(self, *args, **kwargs)


MODEL_VIEW_MAPPINGS = \
    {queue_model_objects.DataCollection: DataCollectionQueueItem,
     queue_model_objects.Characterisation: CharacterisationQueueItem,
     queue_model_objects.EnergyScan: EnergyScanQueueItem,
     queue_model_objects.XRFSpectrum: XRFSpectrumQueueItem,
     queue_model_objects.SampleCentring: SampleCentringQueueItem,
     queue_model_objects.Sample: SampleQueueItem,
     queue_model_objects.Basket: BasketQueueItem,
     queue_model_objects.Workflow: GenericWorkflowQueueItem,
     queue_model_objects.XrayCentering: XrayCenteringQueueItem,
     queue_model_objects.TaskGroup: DataCollectionGroupQueueItem}
```

```
if not self._advanced_methods:
    self._advanced_methods = self._beamline_setup_hwobj.get_advanced_methods()
    if self._advanced_methods:
        for method in self._advanced_methods:
            self._advanced_methods_widget.method_combo.addItem(method)
    else:
        self.setEnabled(False)
```



- Depending from method customize Qt4_create_advanced_widget.
- Possibility to develop some other examples during the developers workshop.

Potential collection methods

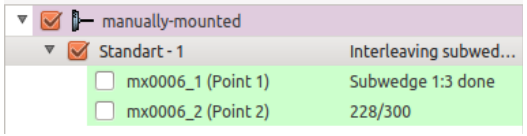
- The number of collection methods may grow in the future.
- It would be vice to implement some abstraction layer now before it gets to crowded.
- Possible solutions:
  1. Implement all methods in queue_entry (already now 1.6k code lines)
  2. Generic queue_entry with minimal set of collection methods:
     a. DataCollection (oscillation, helical line and mesh)
     b. Characterisation, EnergyScan and XRFSpectrum.

  and imports for specific methods:

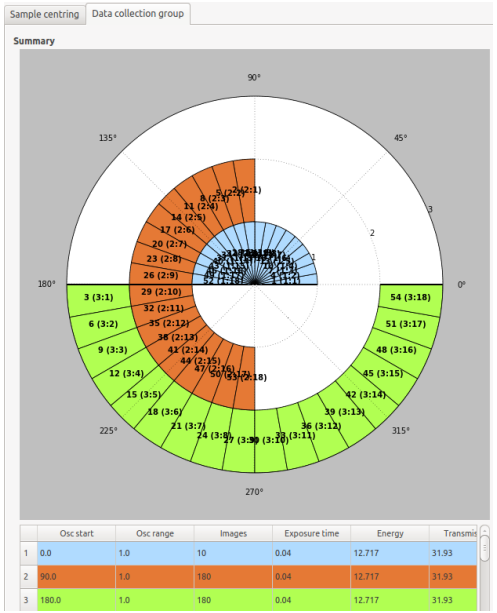
```
from collections import namedtuple
from queue_model_enumerables_v1 import *

from EMBL_queue_entry import EMBLCustomMethod
```

Interleaved collection

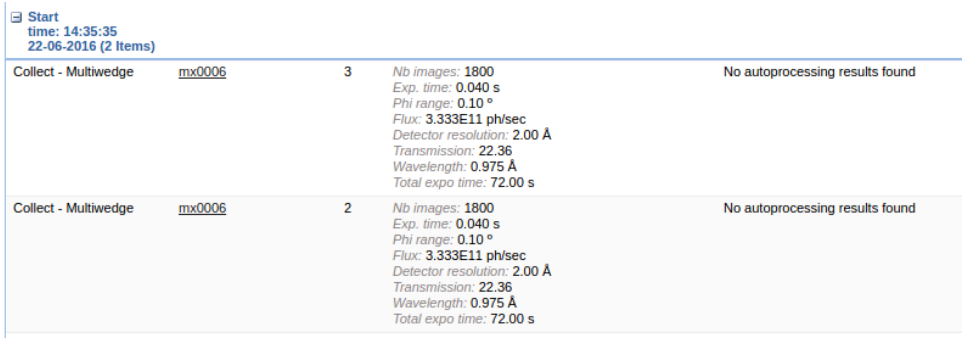
1. Defines how several data collections are executed.
2. Added more log information during the execution:




```
[2016-06-22 14:35:59] Executing interleaved collection (subwedge 1:3, from 201 to 300, osc start: 110.00, osc total range: 10.00)
[2016-06-22 14:35:59] Collection: Preparing to collect
[2016-06-22 14:35:59] Collection: Storing data collection in LIMS
[2016-06-22 14:35:59] Collection: Creating directories for raw images and processing files
[2016-06-22 14:35:59] Collection: Getting sample info from parameters
[2016-06-22 14:35:59] Collection: Moving to centred position
[2016-06-22 14:36:00] Collection: Setting transmission to 28.200
[2016-06-22 14:36:00] Collection: Setting energy to 12.717
[2016-06-22 14:36:00] Collection: Setting resolution to 2.000
[2016-06-22 14:36:00] Collection: Updating data collection in LIMS
[2016-06-22 14:36:00] Collection started
```



3. Storing information in ISPyB



A need to display more information about each subwedge in ISPyB.

AbstractCollect

- Rework of AbstractMulticollect leaded to a new AbstractCollect.
- No wedge loop.
- Clean implementation: do_collect from around 400 lines to 76 lines and abstract methods.

```
def do_collect(self, owner):
    """
    Actual collect sequence
    """
    log = logging.getLogger("user_level_log")
    log.info("Collection: Preparing to collect")
    self.emit("collectReady", (False, ))
    self.emit("collectOscillationStarted", (owner, None, \
              None, None, self.current_dc_parameters, None))

    # ----------------------------------------------------------------------
    self.open_detector_cover()
    self.open_safety_shutter()
    self.open_fast_shutter()

    # ----------------------------------------------------------------------
    self.current_dc_parameters["status"] = "Running"
    self.current_dc_parameters["collection_start_time"] = \
         time.strftime("%Y-%m-%d %H:%M:%S")

    log.info("Collection: Storing data collection in LIMS")
    self.store_data_collection_in_lims()

    log.info("Collection: Creating directories for raw images and processing files")
    self.create_file_directories()

    log.info("Collection: Getting sample info from parameters")
    self.get_sample_info()

    #log.info("Collect: Storing sample info in LIMS")
    #self.store_sample_info_in_lims()

    if all(item == None for item in self.current_dc_parameters['motors'].values()):
        # No centring point defined
        # create point based on the current position
        current_diffractometer_position = self.diffractometer_hwobj.getPositions()
        for motor in self.current_dc_parameters['motors'].keys():
            self.current_dc_parameters['motors'][motor] = \
                 current_diffractometer_position.get(motor)

    log.info("Collection: Moving to centred position")
    self.move_to_centered_position()
    self.take_crystal_snapshots()
    self.move_to_centered_position()
```

```
if "transmission" in self.current_dc_parameters:
    log.info("Collection: Setting transmission to %.3f",
             self.current_dc_parameters["transmission"])
    self.set_transmission(self.current_dc_parameters["transmission"])

if "wavelength" in self.current_dc_parameters:
    log.info("Collection: Setting wavelength to %.3f", \
             self.current_dc_parameters["wavelength"])
    self.set_wavelength(self.current_dc_parameters["wavelength"])

elif "energy" in self.current_dc_parameters:
    log.info("Collection: Setting energy to %.3f",
             self.current_dc_parameters["energy"])
    self.set_energy(self.current_dc_parameters["energy"])

if "resolution" in self.current_dc_parameters:
    resolution = self.current_dc_parameters["resolution"]["upper"]
    log.info("Collection: Setting resolution to %.3f", resolution)
    self.set_resolution(resolution)

elif 'detdistance' in self.current_dc_parameters:
    log.info("Collection: Moving detector to %f",
             self.current_dc_parameters["detdistance"])
    self.move_detector(self.current_dc_parameters["detdistance"])

log.info("Collection: Updating data collection in LIMS")
self.update_data_collection_in_lims()
self.data_collection_hook()
# ----------------------------------------------------------------------

self.close_fast_shutter()
self.close_safety_shutter()
self.close_detector_cover()
```

ParallelProcessing

1. Prepares xml input for EDNA.
2. Standart pipe to start EDNA Dozor plugin.
3. Cyclic output file polling and extracting results. Weakest point because based on file system read.
4. There was an attempt to use xmlrpc to transfer results (work not finished).

Weakness and strangeness of the MXCuBE data mode

+ Implementation of new collection methods and strategies is possible.

- With a current implementation would be difficult to continue if new methods are developed.
- Length of queue_entry is more than 1.6k. 1K suggestion from PEP.
- All sites has to test and have a common agreement on the queue model.
- Testing may take some time.

1. Site specific imports may resolve the testing issue.
2. If a site specific method is well tested and widely used it could move to main queue_entry.

Weakness and strangeness of the MXCuBE data mode

Still some links to Qt in the Hardware level

```
def pre_execute(self):
    BaseQueueEntry.pre_execute(self)
    self.get_view().setOn(True)
    self.get_view().setHighlighted(False)

    self.data_analysis_hwobj = self.beamline_setup.data_ar
    self.diffractometer_hwobj = self.beamline_setup.diffra
    #should be an other way how to get queue_model hwobj:
    self.queue_model_hwobj = self.get_view().listView().\
         parent().queue_model_hwobj
```

```
for edna_dc in edna_collections:
    path_template = edna_dc.acquisitions[0].path_template
    run_number = self.queue_model_hwobj.get_next_run_number(path_template)
    path_template.run_number = run_number

    edna_dc.set_enabled(False)
    edna_dc.set_name(path_template.get_prefix())
    edna_dc.set_number(path_template.run_number)
    self.queue_model_hwobj.add_child(new_dcg_model, edna_dc)
```

Suggestion: use BeamlineSetup as a container for all hwobj

```
<object class="BeamlineSetup" role="BeamlineSetup">
  <object href="/energy" role="energy"/>
  <object href="/eh1/resolution" role="resolution"/>
  <object href="/eh1/diff-omega" role="omega_axis"/>
  <object href="/eh1/diff-kappa" role="kappa_axis"/>
  <object href="/eh1/diff-kappaphi" role="kappa_phi_axis"/>
  <object href="/eh1/detector" role="detector"/>
  <object href="/mini-diff" role="diffractometer"/>
  <object href="/attenuators" role="transmission"/>
  <object href="/beam-info" role="beam_info"/>
  <object href="/fast-shut" role="fast_shutter"/>

  <object href="/sc-generic" role="sample_changer"/>
  <object href="/plate-manipulator" role="plate_manipulator"/>

  <!-- <object href="/sc-generic" role="sample_changer"/>
  <object href="/plate-manipulator" role="plate_manipulator"/> -->

  <object href="/Qt4_graphics-manager" role="shape_history"/>
  <object href="/mxcollect" role="collect"/>
  <object href="/session" role="session"/>
  <object href="/dbconnection" role="lims_client"/>
  <object href="/energyscan" role="energyscan"/>
  <object href="/xrf-spectrum" role="xrf_spectrum"/>
  <object href="/data-analysis" role="data_analysis"/>
  <object href="/parallel-processing" role="parallel_processing"/>
```

```
def init(self):
    """
    Framework 2 init, inherited from HardwareObject.
    """

    self.sample_changer_hwobj = None
    self.plate_manipulator_hwobj = None

    for role in self.getRoles():
        self._get_object_by_role(role)

    self._object_by_path['/beamline/energy'] = self.energy_hwobj
    self._object_by_path['/beamline/resolution'] = self.resolution_hwobj
    self._object_by_path['/beamline/transmission'] = self.transmission_hwobj

    self.advanced_methods = []
    try:
       self.advanced_methods = eval(self.getProperty("advancedMethods"))
    except:
       pass
```

Thank you for your attention!