Abstraction and advanced collection methods

Ivars Karpics

Content

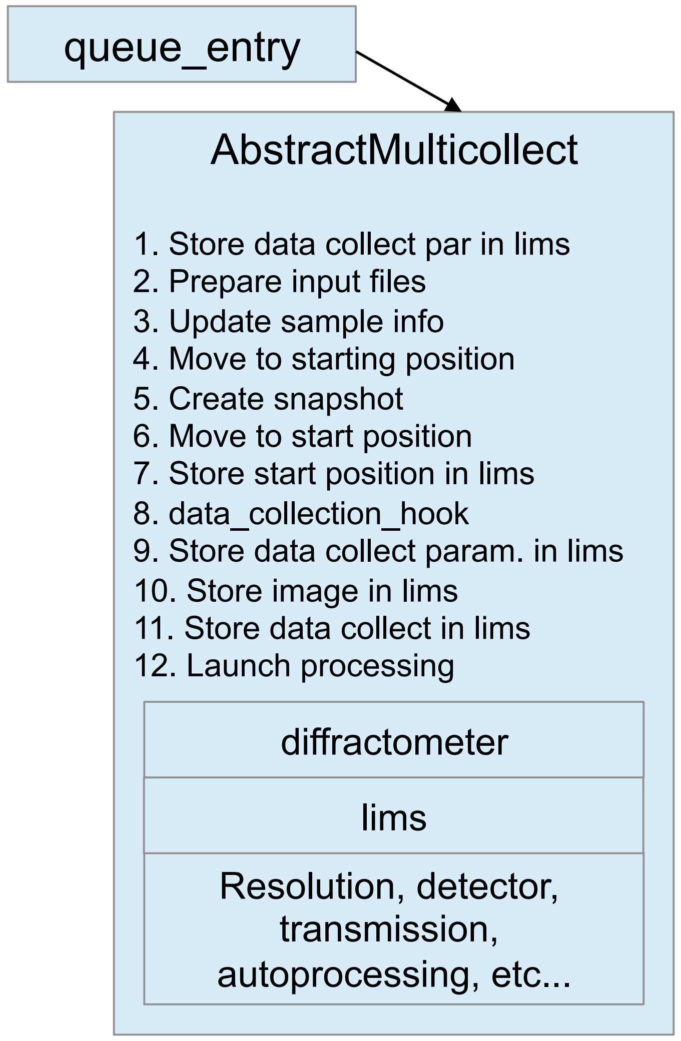
New abstract classes

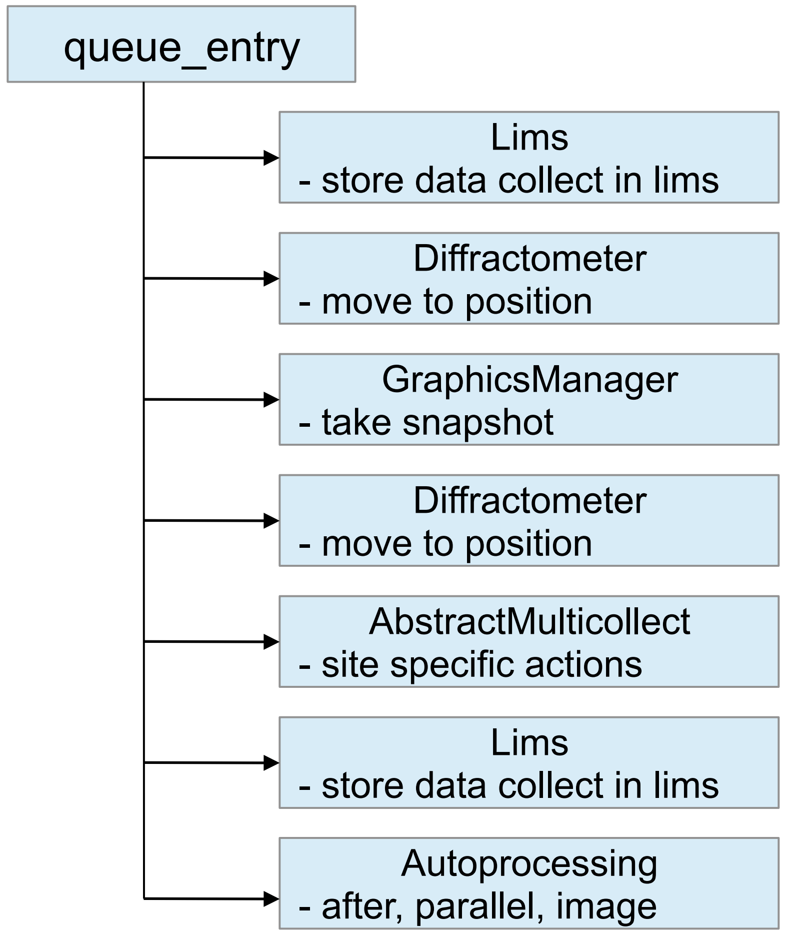
New abstract classes available in git.

- AbstractMotor
- AbstractDetector
- AbstractEnergyScan
- AbstractXrfSpectrum

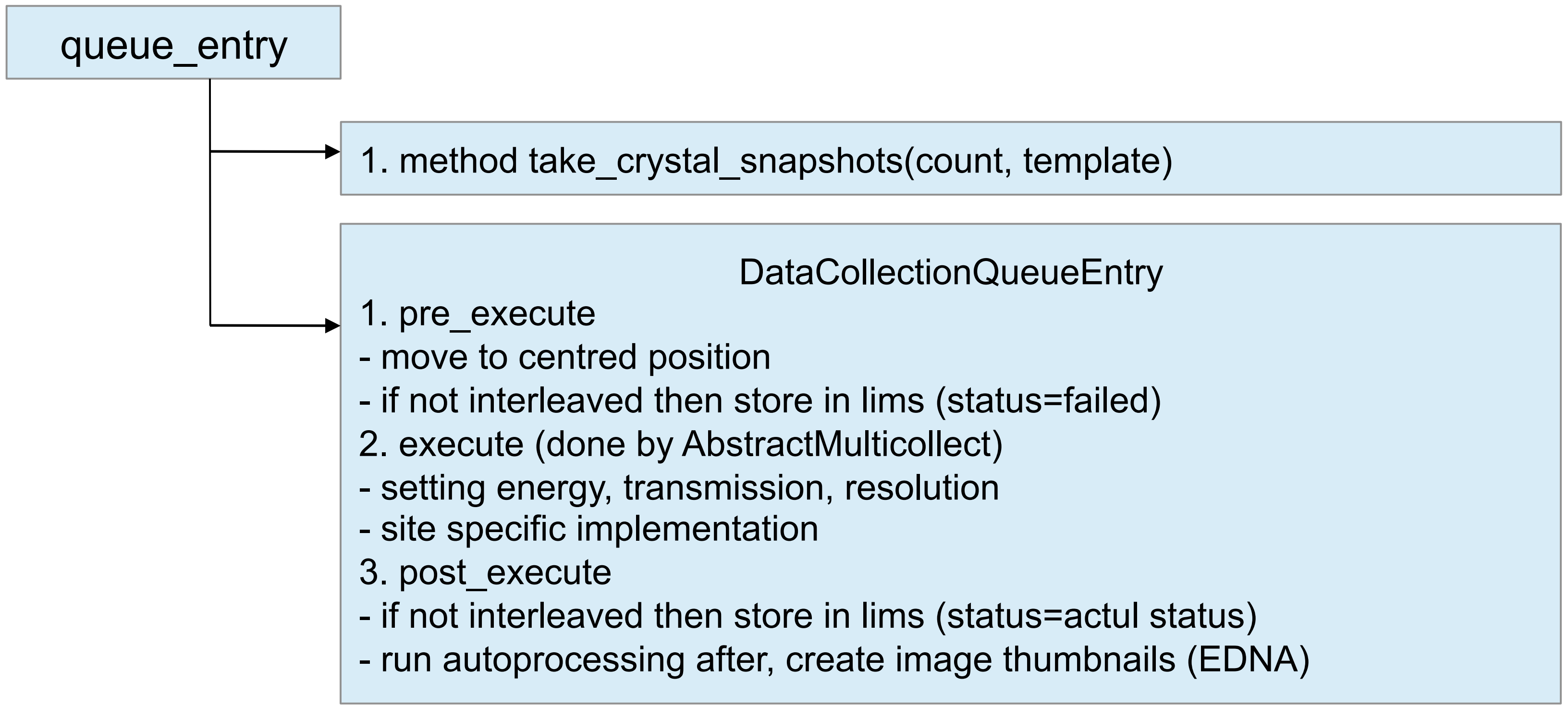
- Important to keep a common interface.
- GenericSampleChanger is a good example.
- Easier adaptation to site specific implementation.
- Avoids branching.
- GenericDiffractometer?

Rework of AbstractMulticollect and queue_entry


queue_entry
AbstractMulticollect
1. Store data collect par in lims
2. Prepare input files
3. Update sample info
4. Move to starting position
5. Create snapshot
6. Move to start position
7. Store start position in lims
8. data_collection_hook
9. Store data collect param. in lims
10. Store image in lims
11. Store data collect in lims
12. Launch processing
diffractometer
lims
Resolution, detector, transmission, autoprocessing, etc...



Rework of AbstractMulticollect and queue_entry


queue_entry

1. method take_crystal_snapshots(count, template)

DataCollectionQueueEntry

1. pre_execute
- move to centred position
- if not interleaved then store in lims (status=failed)
2. execute (done by AbstractMulticollect)
- setting energy, transmission, resolution
- site specific implementation
3. post_execute
- if not interleaved then store in lims (status=actul status)
- run autoprocessing after, create image thumbnails (EDNA)

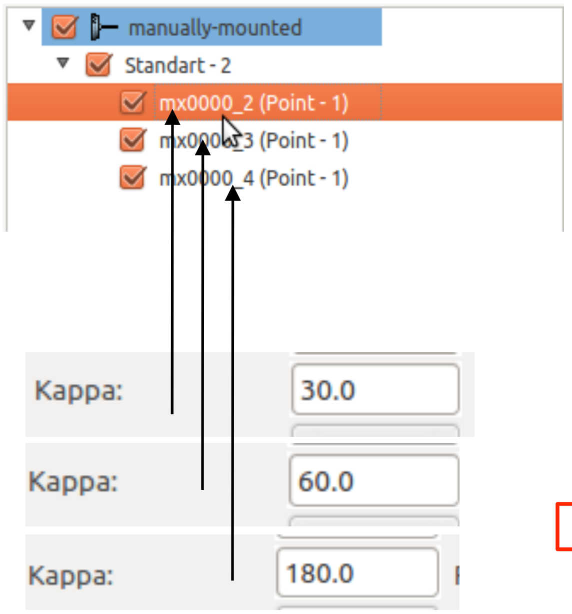
Rework of AbstractMulticollect and queue_entry

1. Added property to be able to run AbstractMulticollect without loop.
2. Queue is not made to send a list of data collection.
3. Delegate more task to queue_entry pre_executer, post_execute methods.
4. Maybe new AbstractCollect?

1. AbstractMulticollect handles to much actions.
2. With current implementation difficult or even impossible to implement more advanced methods like interleaved collections.
3. Give more actions and roles to queue_entry items.
4. Interleaved and similar collections organize in group level (TaskGroupQueueEntry).
5. Use beamline_setup_hwobj as all hwobj container for most bricks and hwobj.

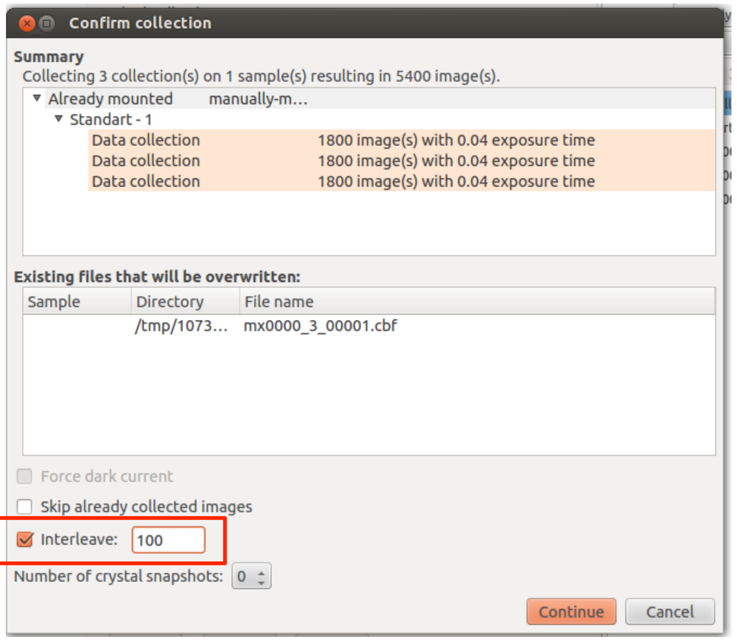
Interleave collections

1. Interleave is possible based on any parameter combination and any number of collections for different centred positions.
2. It is not a new task, but a new way how to execute a task group.
3. Modified confirmation dialogue allows to change execution type of a task group.
4. No need to create numerous new collection that freezes graphics (issue #57).



manually-mounted
Standart - 2
mx0000_2 (Point - 1)
mx0000_3 (Point - 1)
mx0000_4 (Point - 1)

Kappa:	30.0
Kappa:	60.0
Kappa:	180.0



Confirm collection

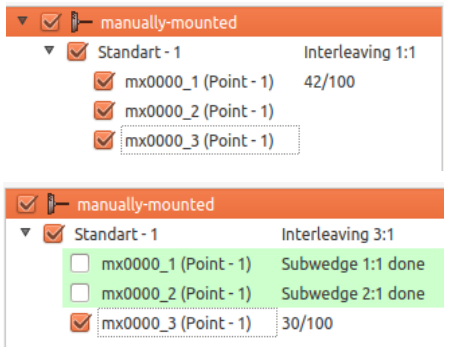
Summary
Collecting 3 collection(s) on 1 sample(s) resulting in 5400 image(s).

```
▼ Already mounted      manually-m...
  ▼ Standart - 1
      Data collection          1800 image(s) with 0.04 exposure time
      Data collection          1800 image(s) with 0.04 exposure time
      Data collection          1800 image(s) with 0.04 exposure time
```

Existing files that will be overwritten:

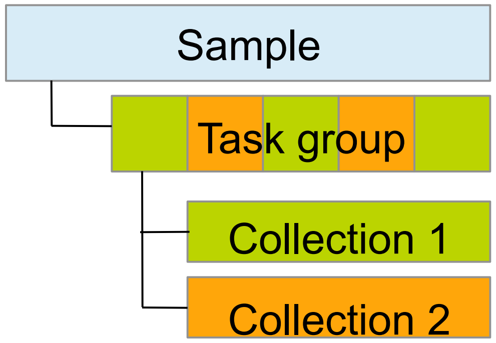
Sample	Directory	File name
	/tmp/1073...	mx0000_3_00001.cbf

☐ Force dark current
☐ Skip already collected images
☑ Interleave: 100
Number of crystal snapshots: 0



▼ ☑ manually-mounted	
▼ ☑ Standart - 1	Interleaving 1:1
☑ mx0000_1 (Point - 1)	42/100
☑ mx0000_2 (Point - 1)	
☑ mx0000_3 (Point - 1)	

☑ manually-mounted	
▼ ☑ Standart - 1	Interleaving 3:1
☐ mx0000_1 (Point - 1)	Subwedge 1:1 done
☐ mx0000_2 (Point - 1)	Subwedge 2:1 done
☑ mx0000_3 (Point - 1)	30/100



Interleave collections

Interleave is possible based on any parameter combination and any number of collections for different centred positions. Examples:

1. 1800 images (Point 1)
2. 1800 images (Point 1)

Subwedge size 100

1. sw1:1(p1)
2. sw2:1(p1)

…

35. sw1:18 (p1)
36. sw2:18 (p1)

Total:
18 sw1 (100 frames)
18 sw2 (100 frames)

1. 900 images (Point 1)
2. 1800 images (Point 2)

Subwedge size 100

1. sw1:1(p1)
2. sw2:1(p2)

…

26. sw2:17 (p2)
27. sw2:18 (p2)

Total:
9 sw1 (100 frames)
18 sw2 (100 frames)

1. 900 images (Point 1)
2. 960 images (Point 2)

Subwedge size 100

1. sw1:1(p1)
2. sw2:1(p2)

…

17. sw1:9 (p1)
18. sw2:9 (p2)
19. sw2:10 (p2)

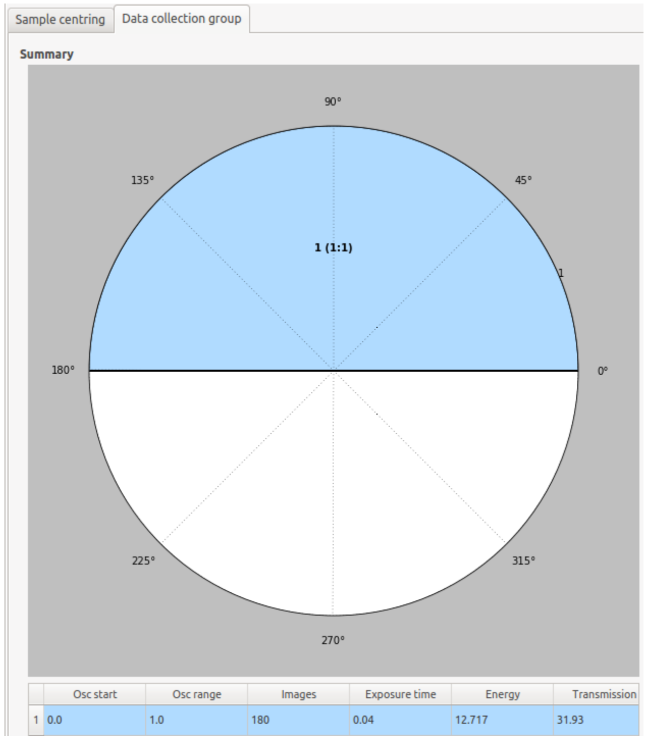
Total:
9 sw1 (100 frames)
9 sw2 (100 frames)
1 sw2 (60 frames)

Interleave collections (execution sequence)

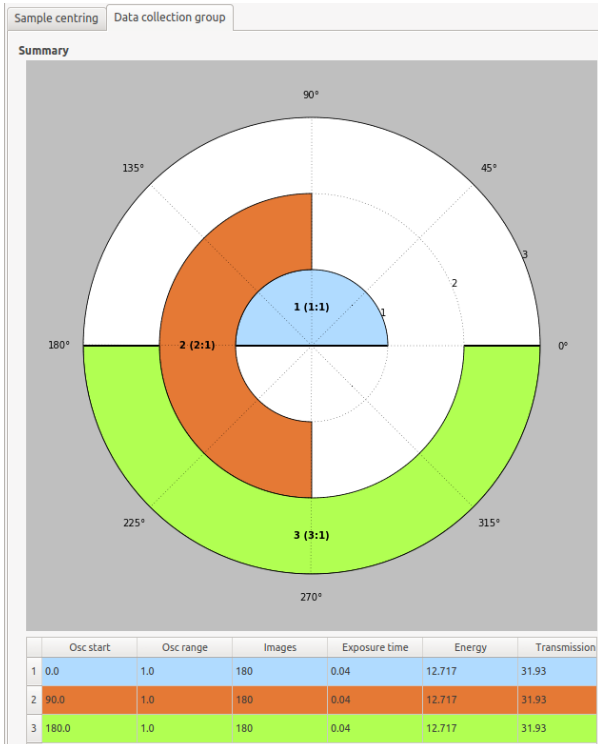
1. It is possible to interleave discreate collections. Other collection types (characterisation, energy scan, ...) will be executed as usually.
2. Before executing a task group, subwedge size is checked.
3. Based on the subwedge size and reference data collections, a list with subwedges is created.
4. Reference collections are not executed, but are stored in ISPyB.
5. Within a loop reference data collections are modified based on the subwedge parameters:
    - start_image_num
    - image_num
    - osc_start
6. No snapshots are taken before each subwedge and information about subwedge is not stored in ISPyB.
7. A link to graphical tree entry is kept to display information about current subwedge.

Data collection group

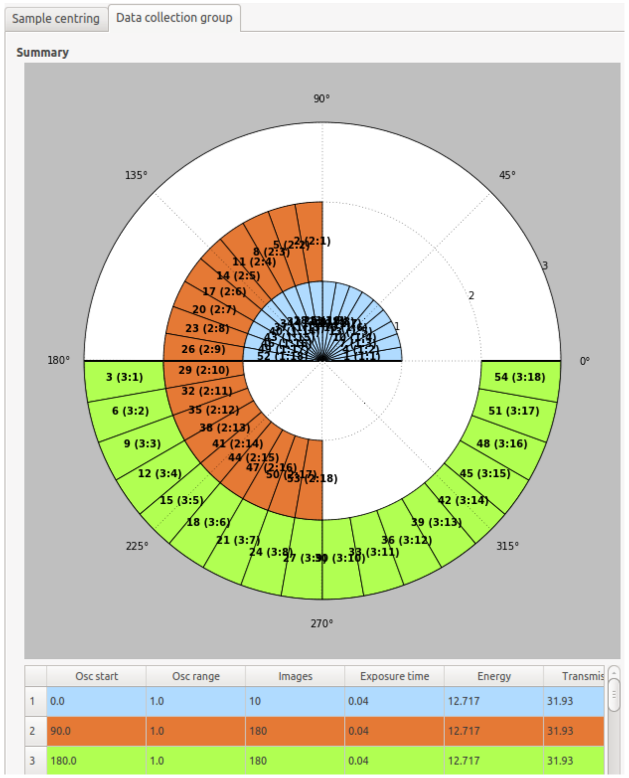
1. A new way how to represent an experiment.
2. Could also dynamically represent a queue.
3. Maybe reorganize wedges based on osc start and end.



	Osc start	Osc range	Images	Exposure time	Energy	Transmission
1	0.0	1.0	180	0.04	12.717	31.93



	Osc start	Osc range	Images	Exposure time	Energy	Transmission
1	0.0	1.0	180	0.04	12.717	31.93
2	90.0	1.0	180	0.04	12.717	31.93
3	180.0	1.0	180	0.04	12.717	31.93

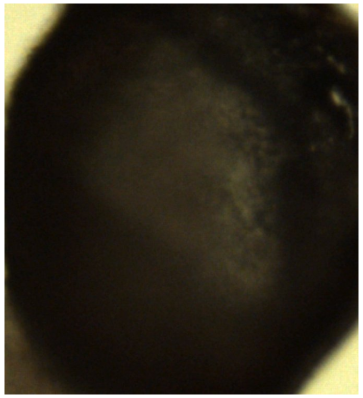
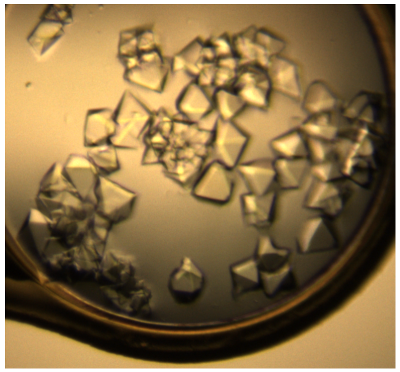
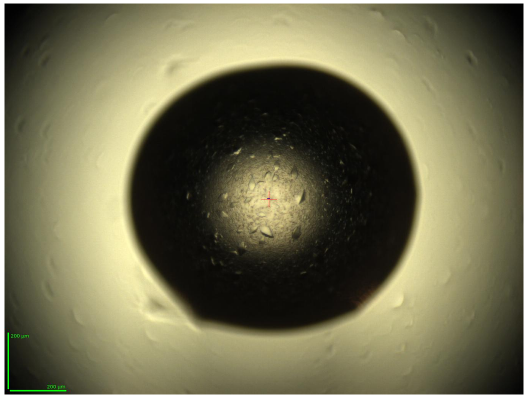


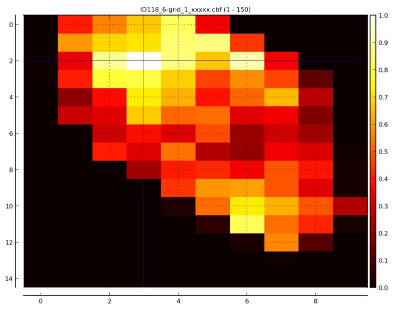
	Osc start	Osc range	Images	Exposure time	Energy	Transmis
1	0.0	1.0	10	0.04	12.717	31.93
2	90.0	1.0	180	0.04	12.717	31.93
3	180.0	1.0	180	0.04	12.717	31.93

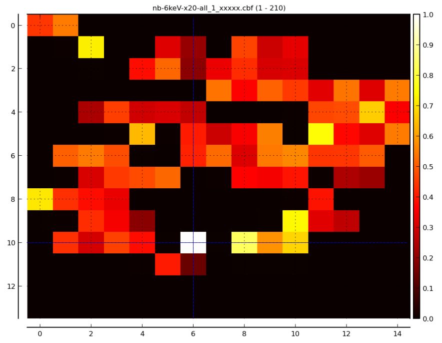
Video demo...

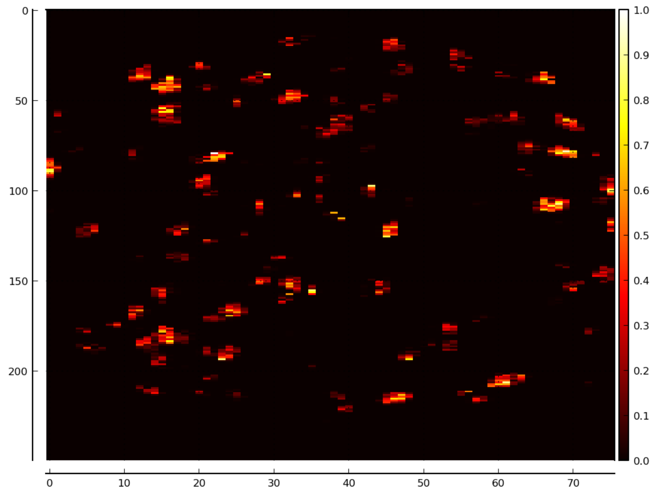
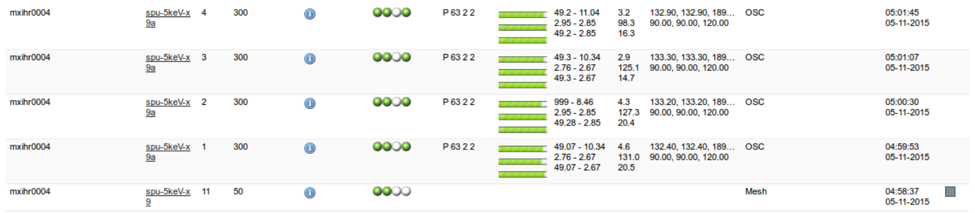
MeshScan (examples)

1. Used daily by many users to find crystals or best collection positions.
2. Mesh and scan feature (automatic centring points).
3. Large grids as SFX. For example 23 868 images, exposure time 0.05 s in less than 21 min.





mxihr0004	spu-5keV-x 9a	4	300	ⓘ	●●○●	P 63 2 2	▬ ▬ ▬	49.2 - 11.04 2.95 - 2.85 49.2 - 2.85	3.2 98.3 16.3	132.90, 132.90, 189… 90.00, 90.00, 120.00	OSC	05:01:45 05-11-2015	
mxihr0004	spu-5keV-x 9a	3	300	ⓘ	●●○●	P 63 2 2	▬ ▬ ▬	49.3 - 10.34 2.76 - 2.67 49.3 - 2.67	2.9 125.1 14.7	133.30, 133.30, 189… 90.00, 90.00, 120.00	OSC	05:01:07 05-11-2015	
mxihr0004	spu-5keV-x 9a	2	300	ⓘ	●●○●	P 63 2 2	▬ ▬ ▬	999 - 8.46 2.95 - 2.85 49.28 - 2.85	4.3 127.3 20.4	133.20, 133.20, 189… 90.00, 90.00, 120.00	OSC	05:00:30 05-11-2015	
mxihr0004	spu-5keV-x 9a	1	300	ⓘ	●●○●	P 63 2 2	▬ ▬ ▬	49.07 - 10.34 2.76 - 2.67 49.07 - 2.67	4.6 131.0 20.5	132.40, 132.40, 189… 90.00, 90.00, 120.00	OSC	04:59:53 05-11-2015	
mxihr0004	spu-5keV-x 9	11	50	ⓘ	●●○○						Mesh	04:58:37 05-11-2015	■

Implementing MeshScan

Modules needed to execute MeshScan:
1. Multicollect with capability to run a mesh scan.
2. Configured ParallelProcessing hwobj.
3. EDNA dozor plugin.

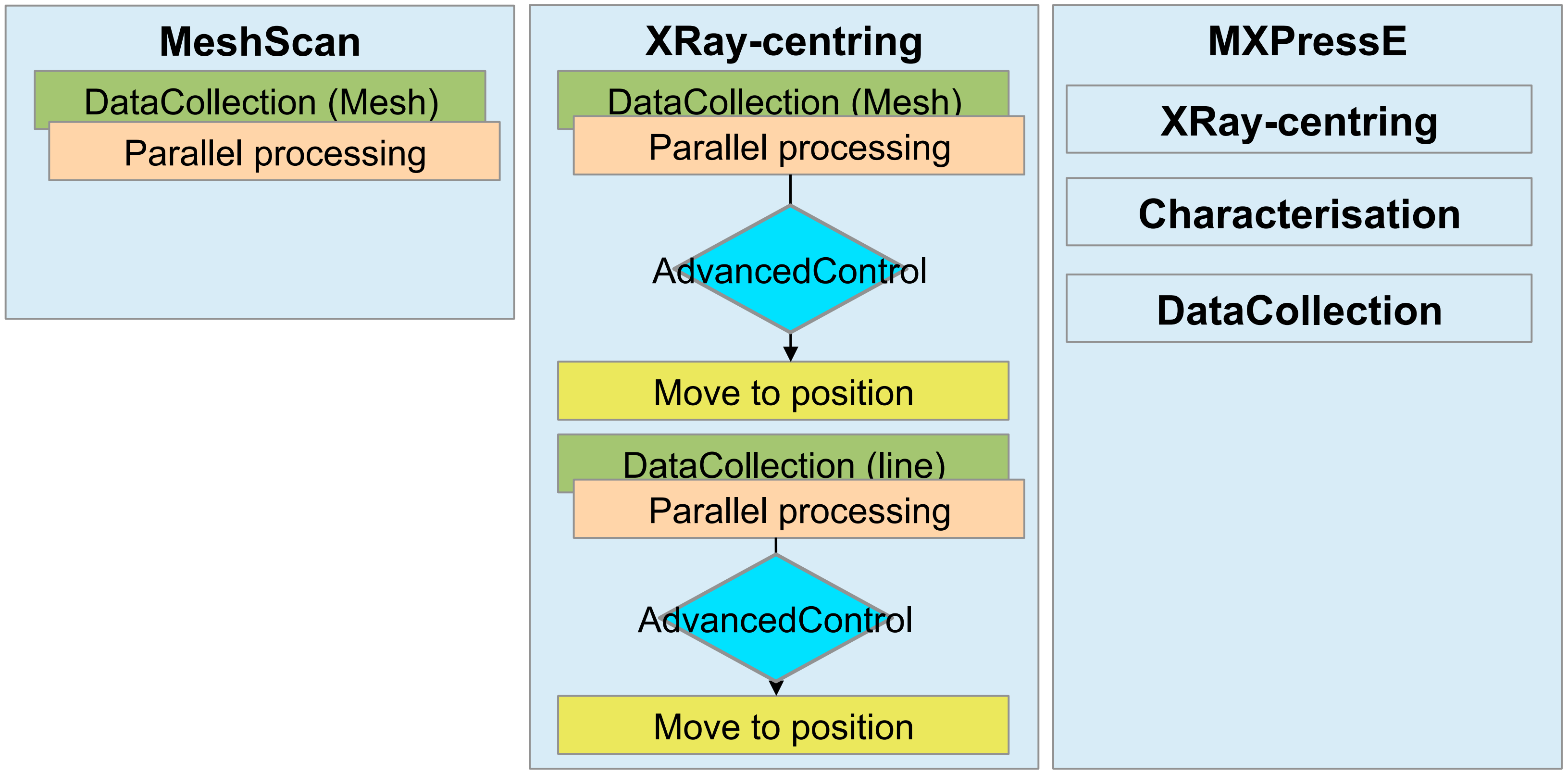
Execution sequence:
1. EDNA input file is generated.
2. Processing via script is launched.
3. Result xml files are polled and results are aligned.
4. Signals emit aligned results to brick after each xml.
5. Information about grid, workflow and best positions are stored in ISPyB.
6. Heat map as png file is generated. For helical line curve plot.
7. HTML page as a report is generated to be available in ISPyB.

ParallelProcessing.py, SimpleHTML.py and XDSDataControlDozor.py are in git.
ISPyBClient2.py is updated to be able to store all information.

Video demo…

Implementing other advanced methods

1. MeshScans, Xray-centring, BurnStrategy, MXPress type workflows
2. Use new queue entries:
- AdvancedGroupQueueEntry
- AdvancedControlQueueEntry
3. Define available advanced methods in beamline_setup



Conclusion

1. New abstract classes to generalize mxcube code and keep a common track.
2. Improved Queue and AbstractMulticollect.
3. With a testing in the beamline could be fully available within few weeks.
4. New way how to execute interleaved collection.
5. Graphical representation of multi wedge collections as polar graph.
6. New hardware objects to run MeshScans and other advanced methods.

- All these features are available and fully compatible with Qt4 gui.
- Front-end is nice to see, but lets not forget about back-end and queue execution.

TODO (to be pushed in the git):

1. 2.2. tag for Hardware objects and gui.
2. Finish interleaved collections.
3. Finish queue save/load.
4. Debug and test GUI.

Thank you for your attention