# XDS in DNA. How?

Abstraction?

Data Access Layer (DAL)?

What design for inclusion in DNA?

# Abstraction?

- It would be nice to have more abstraction in the scheduler requests and responses
  - At the moment they are too Mosflm specific
  - Clean things (many unused elements)
- Or should we have Data Processing specific requests and responses?
  - It could be a temporary solution
  - But it doesn’t meet the design goal of abstraction

# Data Access Layer?

- Will Store
  - User input data through the GUI
    - Type of experiment
    - Target space group and symmetry
    - Strategy chosen…
  - Experimental contextual Information coming from the BCM (ex: highest recordable resolution)
  - Main results from Data Procession by the scheduler (ex: estimation of the highest diffracted resolution by method X)

# Data Access Layer?

- Will provide a uniform information access
  - Wright now there are at least 4 source of detector information (DNA internal from the BCM, DiffractionImage, Mosflm, Best). +XDS has its own
  - The detector object in dna_common.xsd is very limited
    - Only 2 attributes: “type” (“ADSC”, “MARCCD”…) and “suffix”
    - What about pixel size, number of pixels, gain, distance, orientation, binned mode, read-out time…

# XDSauto

- Pure python (uses only standard library module), less than 2500 lines
  - Automatic Laue group determination
  - Multithreaded integration possible on clusters
  - Automatic scaling and exports
- Symmetry information stored in dictionaries
- 4 main classes XDS, XParam, Lattice, DataCollectInfo
- LGPL licence

# XDSauto IO

- INPUTS can come from 3 sources
  - From the image headers as read by the DiffractionImage module
  - From xdsSetupDB: a xds specific Goniostat/Detector DB module
  - From some defaults parameters and/or arguments comming at runtime

# XDSauto IO

- OUTPUT: some wrappers are interpreting the XDS log file (*.LP) to produce
  - An xdsauto.log and xdsauto.html files
  - A limited implementation of the index_response
- EXPORTS:
  - Reflections in various formats: MTZ, CNS, SHELX, SOLVE, EPMR, AMoRe, FALL
  - Options for merged/unmerged, anomal/normal
  - Standard input files for some programs are written
  - Can inherite from a free reflection set

# XDSauto IO

- EXPORTS (needs Numerical Python module):
  - Orientation matrices, (miss)setting angles can be imported, exported to Mosflm, Denzo conventions
  - xds2mos, xds2dnz, mos2dnz, mos2xds, dnz2mos
- SETUP: Only 2 optional shell variables.
  - PYTHONPATH to the xupy python modules.
  - XDSHOME, points to the XDS execs (xds, xscale, xdsconv). If XDSHOME is not set, the default PATH variable is used.

# Summary

- What design goal for inclusion of XDS in DNA?
- I don’t want to work on a temporary solution
- Scheduler needs major improvement to have a “clean” inclusion of XDS
- We need a more abstracted **data model** and **scheduler framework** (some thing like XIA)
- (Include Numerical python as dependency for numerical calculations)