GphL Workflows in Use

Peter Keller

Global Phasing Ltd.

http://www.globalphasing.com/

GΦL

Trieste, September 2019

Summary

- Scope and rationale
- Two data models:
  - Abstract Beamline Interface
  - Persistence Layer: characteristics and purpose
- Calibration

Scope of GΦL workflows

- "Traditional" MX crystallography
  - Single-crystal, or a small number of crystals
- Still a lot of room for improvement
  - MX Data collections are rarely optimal
  - "Improvement" means more informative electron density maps
  - Achieved through better experimental protocols
    - Improving the downstream processing and refinement cannot fully compensate for limitations arising from sub-optimal data collection

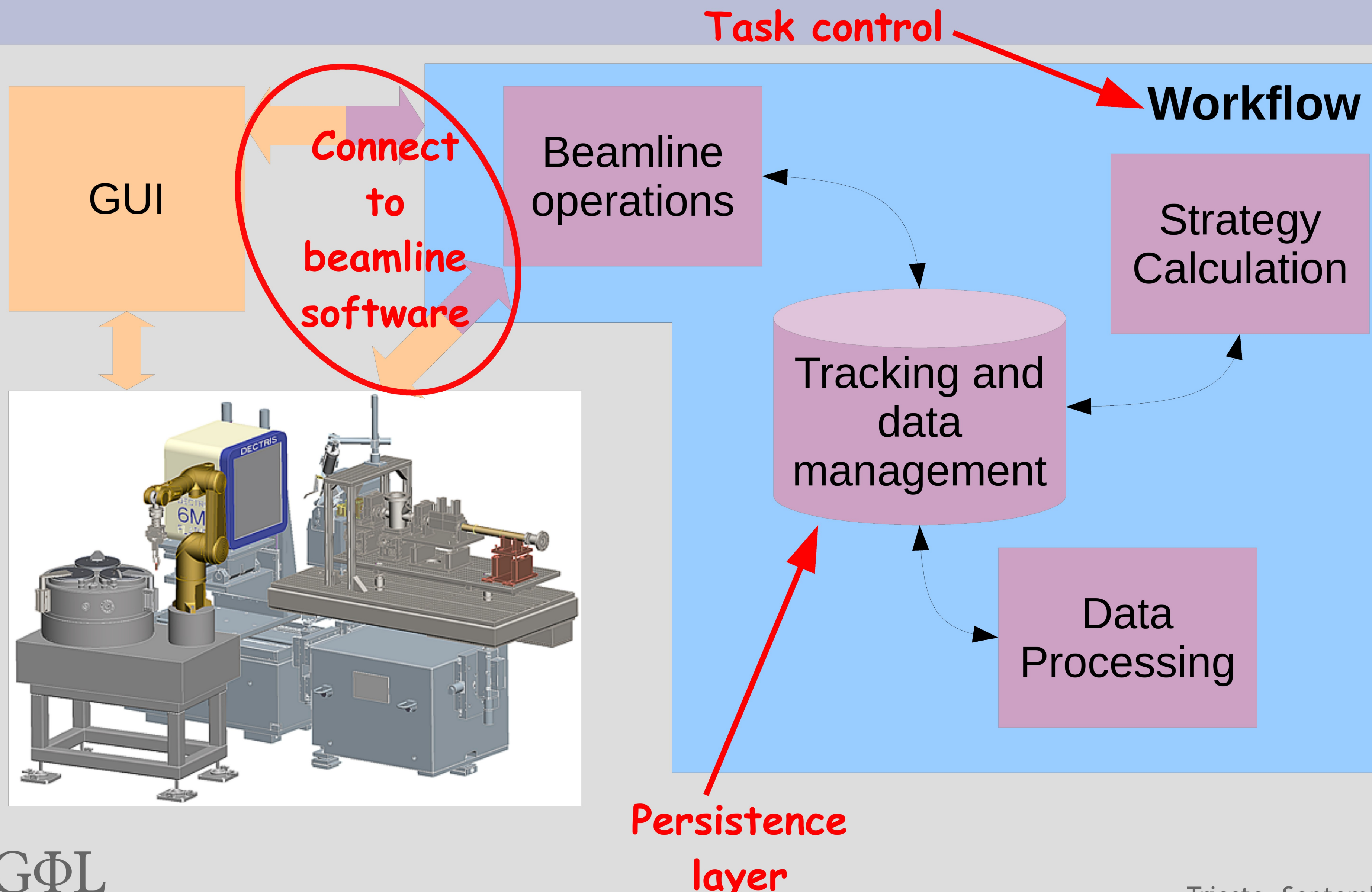
How much does this matter?

- The data collection can make a huge difference to the interpretability of the electron density maps
  - Both for experimental phasing and native data
  - A high-resolution reference structure refined against “Club class” data can significantly improve other structures of the same protein, even when the data collections for those other structures have followed simple protocols.

GΦL's take on data collection protocols

- Transferable
  - Cannot justify synchrotron-specific solutions
- Use a useful abstraction of instrumentation
  - Software access to individual devices/motors is inappropriate
  - Level of abstraction corresponds to that used for strategy calculation and image processing
- Data processing and collection are linked
  - Data processing should:
    - have full information about the strategy and collection
    - not use the collected data to “calibrate” instrumentation

Key software technologies

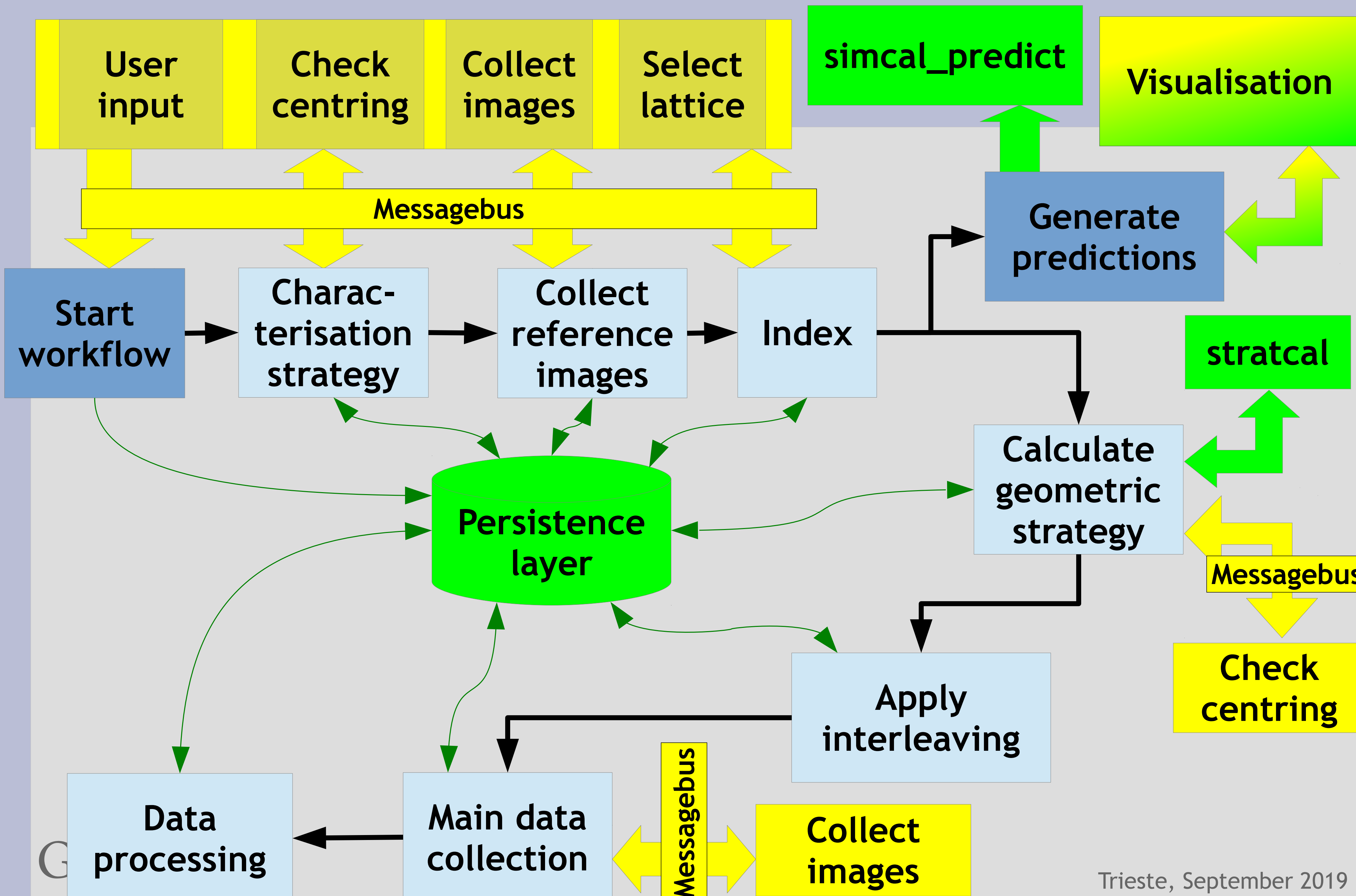


Connection to beamline control software

- Different synchrotrons use different beamline control systems
  - GDA (Generic Data Acquisition): Diamond
  - MXCuBE: 6 European Synchrotrons
  - Blu-Ice: SSRL, GM/CA-CAT
  - …
- Each one needs a different approach to connect with external software
  - We are using Py4J for MXCuBE

Abstract Beamline Interface

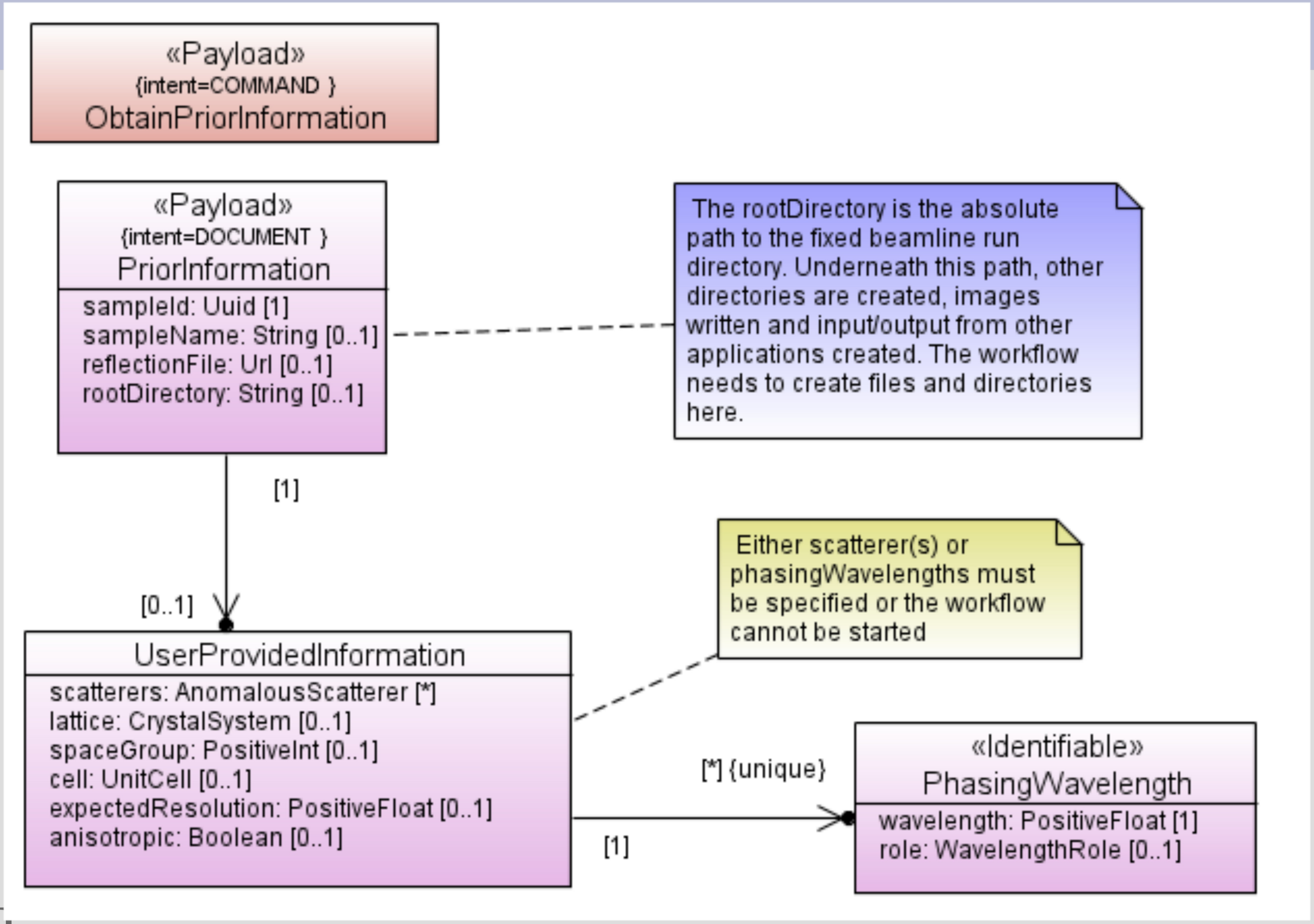
MX Experimental Workflow



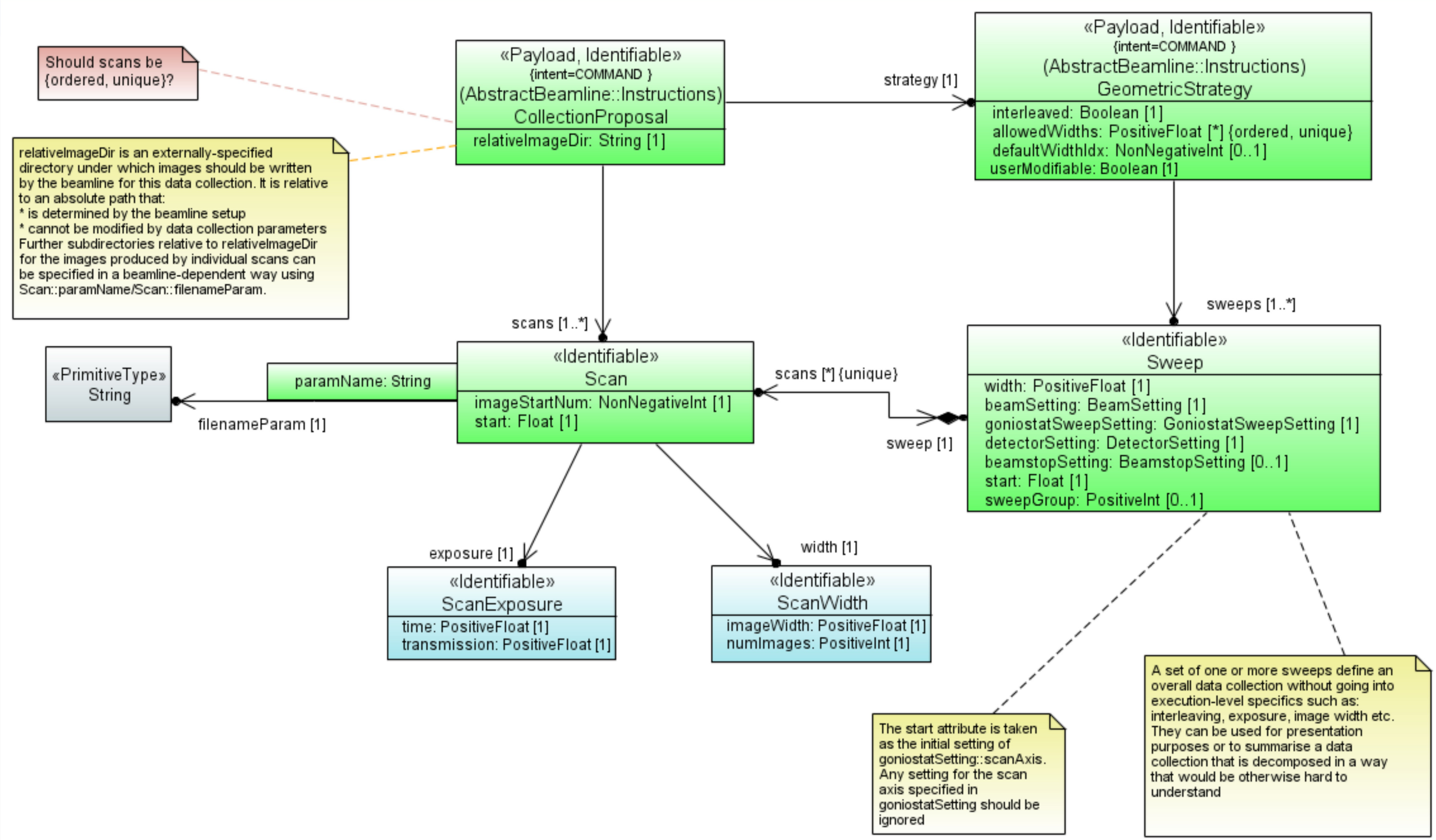
Abstract Beamline Interface

- Defines the data that are exchanged between the workflow and the beamline control system.
  - Strategies and data collection
  - Centring requests and results
  - User-entered information
  - Indexing results
- Uses a scientific level of abstraction
  - Appropriate to strategy calculation and data processing
  - Omits lower-level details (e.g. relevant to instrument control)
- Search for “abstract beamline interface” on https://github.com/
  - Click on the “Wiki” link for more discussion

Abstract Beamline Interface



Abstract Beamline Interface

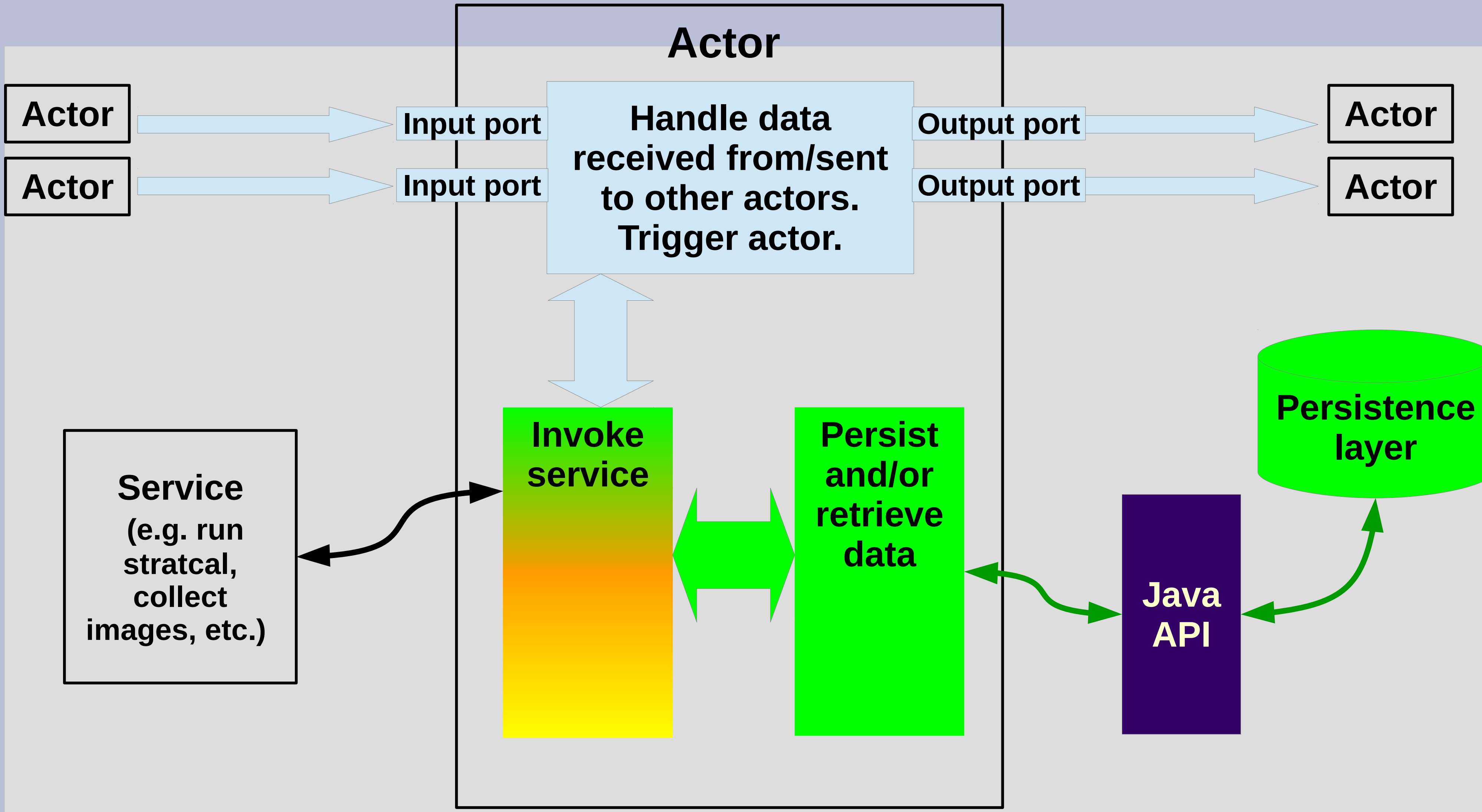


Abstract Beamline in use

- Successfully used with test samples on:
  - DLS-I23: specialised beamline with unusual detector geometry
  - DLS-I04: in 2016 when it still used a mini-Kappa
  - ESRF-ID30B: using an old Qt3-based version of MXCuBE
- Next steps:
  - Real industrial project data on ESRF-ID30B
  - ALBA-BL13 and SOLEIL-PX1/2: testing deployment with a Qt4 version of MXCuBE

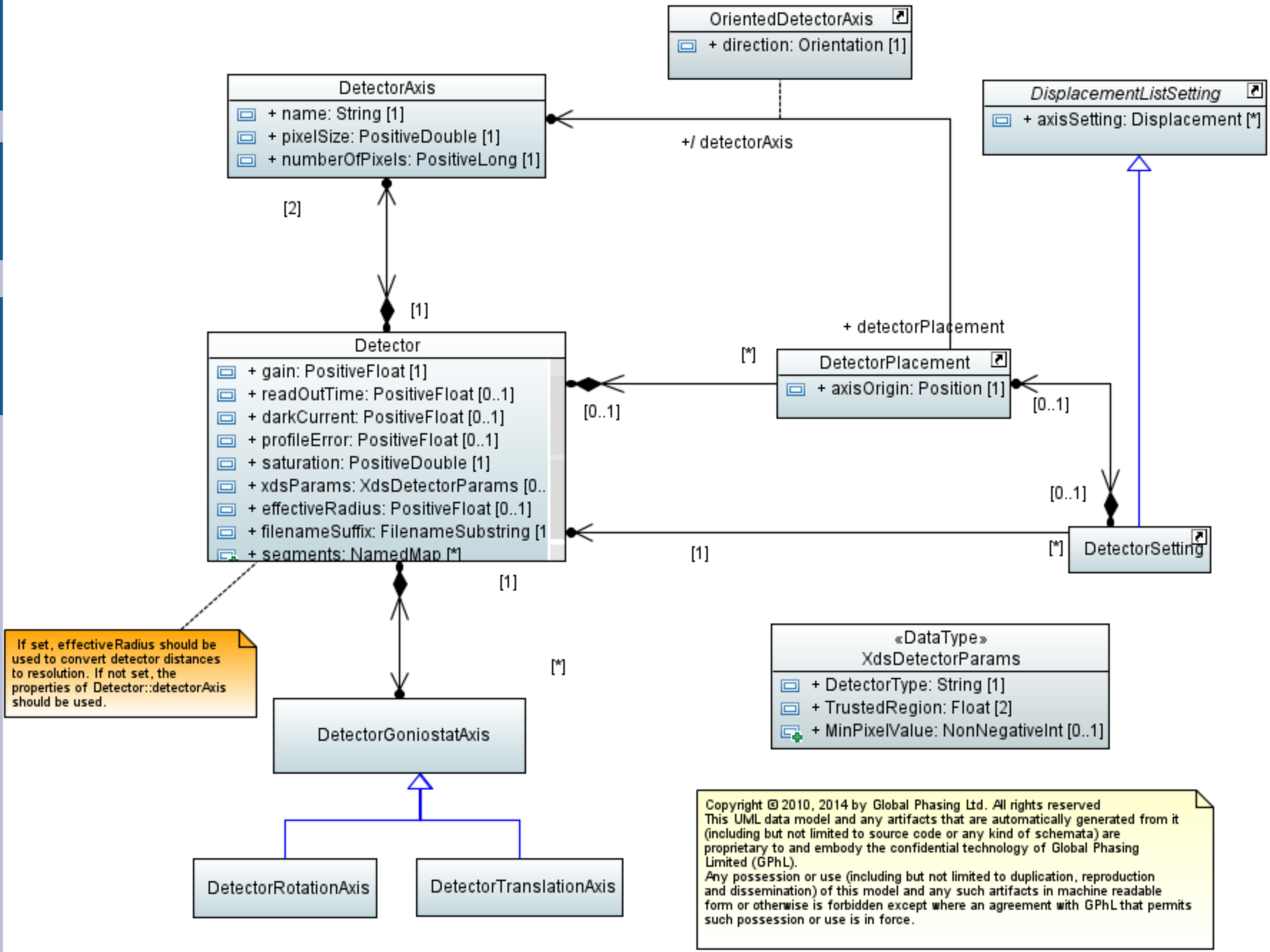
Persistence layer

Anatomy of a workflow actor

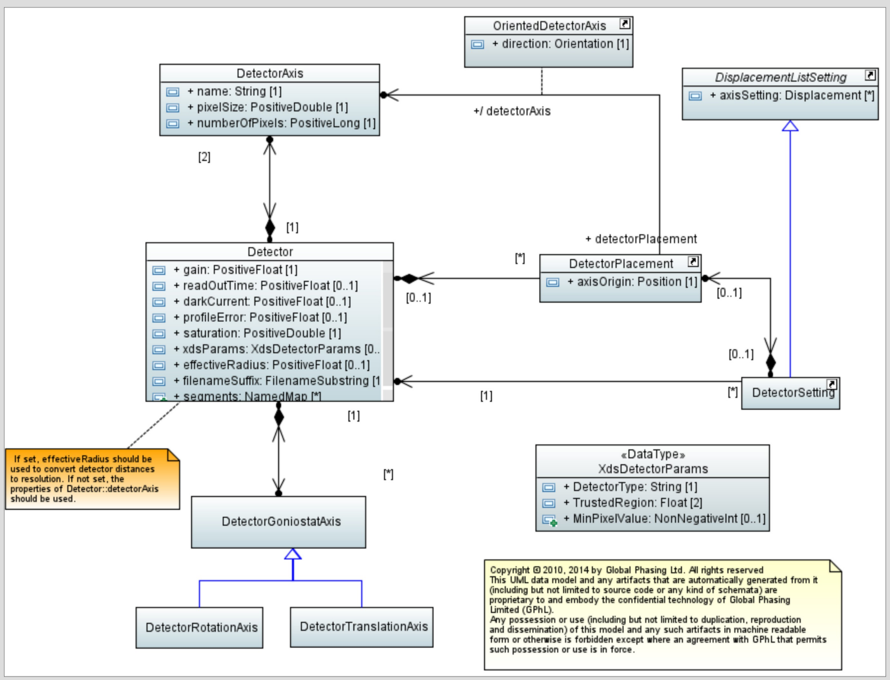


The persistence layer

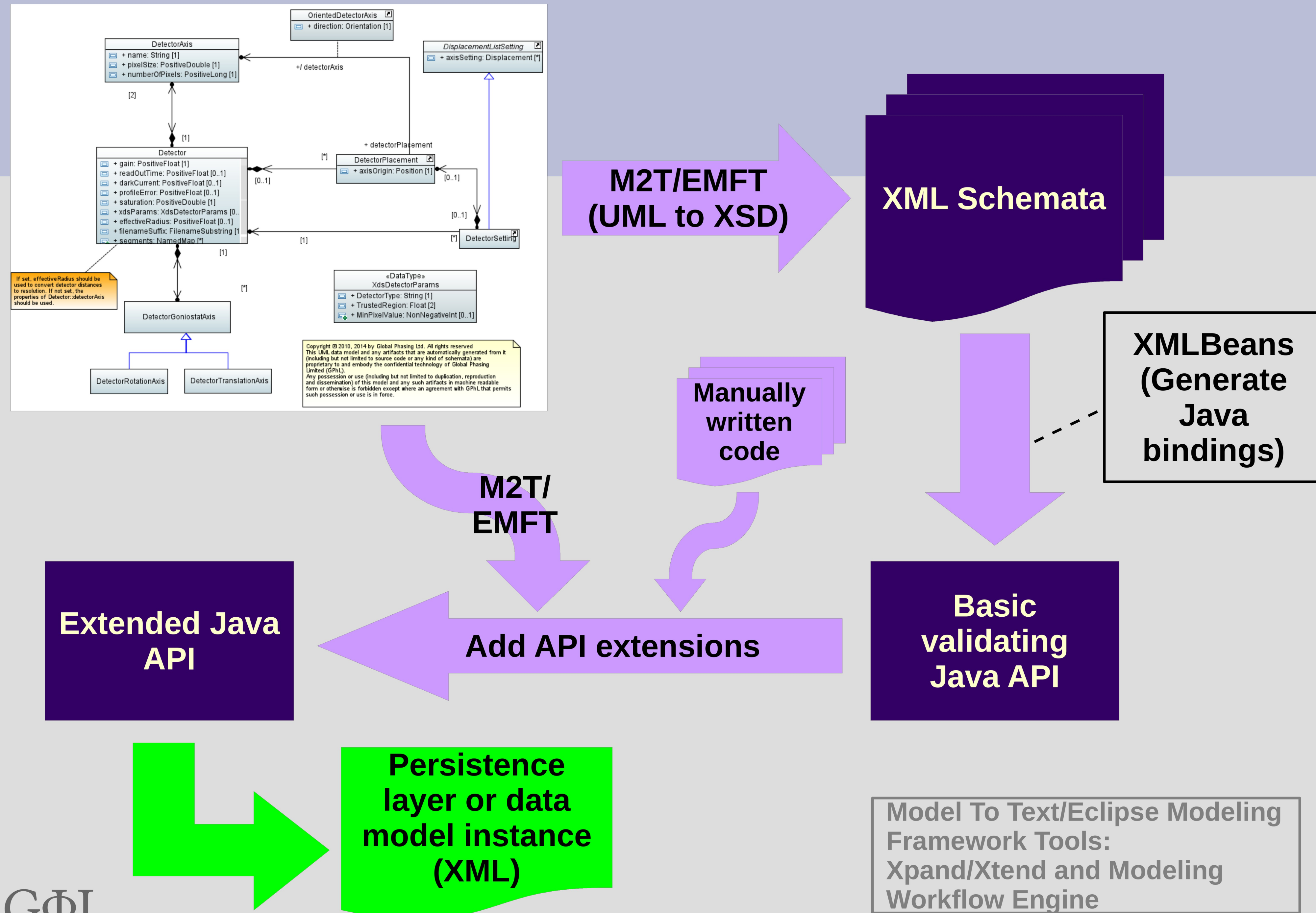
- Structured store for application data
  - We do not rely on passing applications' output files downstream to other applications
- Defined using a subset of the UML (Universal Modeling Language)
  - Working code is generated from the data model by a process developed by Global Phasing Ltd.
- Allows enforcement of constraints on, and validation of, the persisted data


OrientedDetectorAxis
+ direction: Orientation [1]
DetectorAxis
+ name: String [1]
+ pixelSize: PositiveDouble [1]
+ numberOfPixels: PositiveLong [1]
+/ detectorAxis
DisplacementListSetting
+ axisSetting: Displacement [*]
[2]
[1]
+ detectorPlacement
[*]
Detector
+ gain: PositiveFloat [1]
+ readOutTime: PositiveFloat [0..1]
+ darkCurrent: PositiveFloat [0..1]
+ profileError: PositiveFloat [0..1]
+ saturation: PositiveDouble [1]
+ xdsParams: XdsDetectorParams [0..
+ effectiveRadius: PositiveFloat [0..1]
+ filenameSuffix: FilenameSubstring [1
+ segments: NamedMap [*]
DetectorPlacement
+ axisOrigin: Position [1]
[0..1]
[0..1]
[0..1]
[1]
[*]
DetectorSetting
[1]
If set, effectiveRadius should be used to convert detector distances to resolution. If not set, the properties of Detector::detectorAxis should be used.
[*]
«DataType»
XdsDetectorParams
+ DetectorType: String [1]
+ TrustedRegion: Float [2]
+ MinPixelValue: NonNegativeInt [0..1]
DetectorGoniostatAxis
DetectorRotationAxis
DetectorTranslationAxis
Copyright © 2010, 2014 by Global Phasing Ltd. All rights reserved
This UML data model and any artifacts that are automatically generated from it (including but not limited to source code or any kind of schemata) are proprietary to and embody the confidential technology of Global Phasing Limited (GPhL).
Any possession or use (including but not limited to duplication, reproduction and dissemination) of this model and any such artifacts in machine readable form or otherwise is forbidden except where an agreement with GPhL that permits such possession or use is in force.

Size of the data model

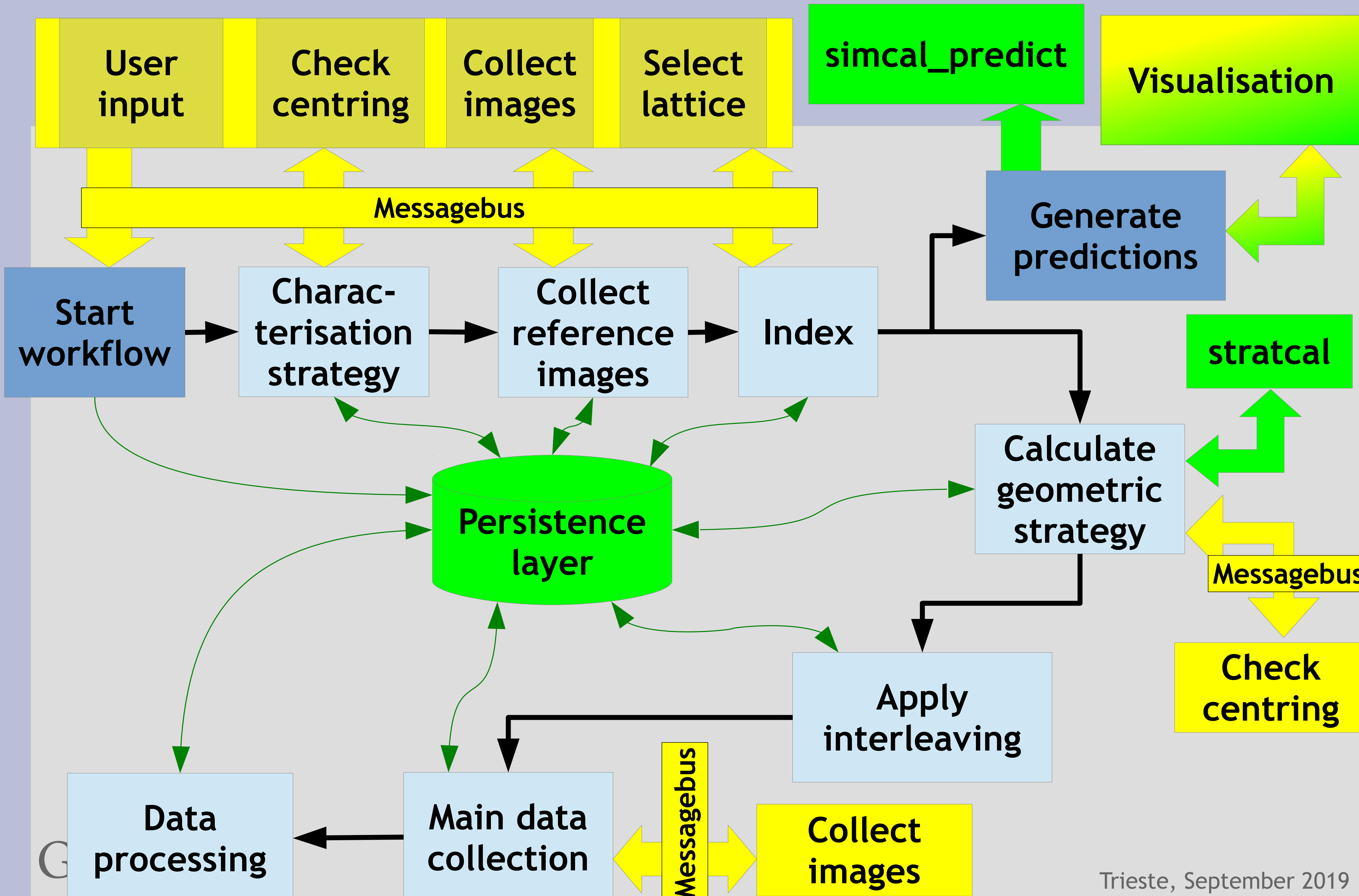


- About 120 Classifiers
  - Classes, Enumerations, DataTypes, PrimitiveTypes
- About 50 Associations



OrientedDetectorAxis
+ direction: Orientation [1]
DetectorAxis
+ name: String [1]
+ pixelSize: PositiveDouble [1]
+ numberOfPixels: PositiveLong [1]
+/ detectorAxis
DisplacementListSetting
+ axisSetting: Displacement [*]
Detector
+ gain: PositiveFloat [1]
+ readOutTime: PositiveFloat [0..1]
+ darkCurrent: PositiveFloat [0..1]
+ profileError: PositiveFloat [0..1]
+ saturation: PositiveDouble [1]
+ effectiveRadius: PositiveFloat [0..1]
+ detectorPlacement
DetectorPlacement
+ axisOrigin: Position [1]
DetectorSetting
«DataType»
XdsDetectorParams
+ DetectorType: String [1]
+ TrustedRegion: Float [2]
+ MinPixelValue: NonNegativeInt [0..1]
DetectorGoniostatAxis
DetectorRotationAxis
DetectorTranslationAxis
M2T/EMFT
(UML to XSD)
XML Schemata
XMLBeans
(Generate
Java
bindings)
Manually
written
code
M2T/
EMFT
Extended Java
API
Add API extensions
Basic
validating
Java API
Persistence
layer or data
model instance
(XML)
Model To Text/Eclipse Modeling
Framework Tools:
Xpand/Xtend and Modeling
Workflow Engine
GΦL

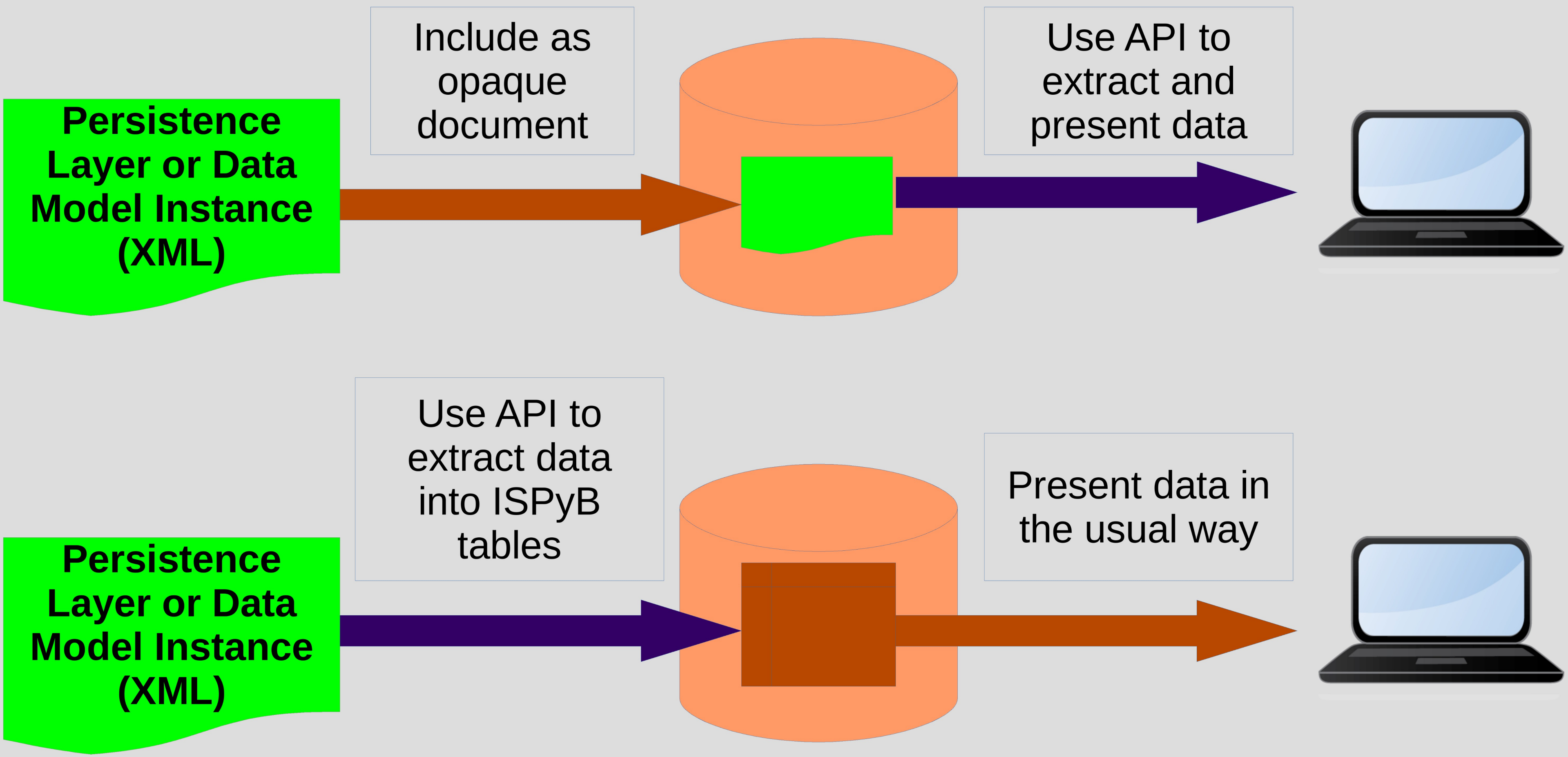
MX Experimental Workflow



Purposes of the Persistence Layer

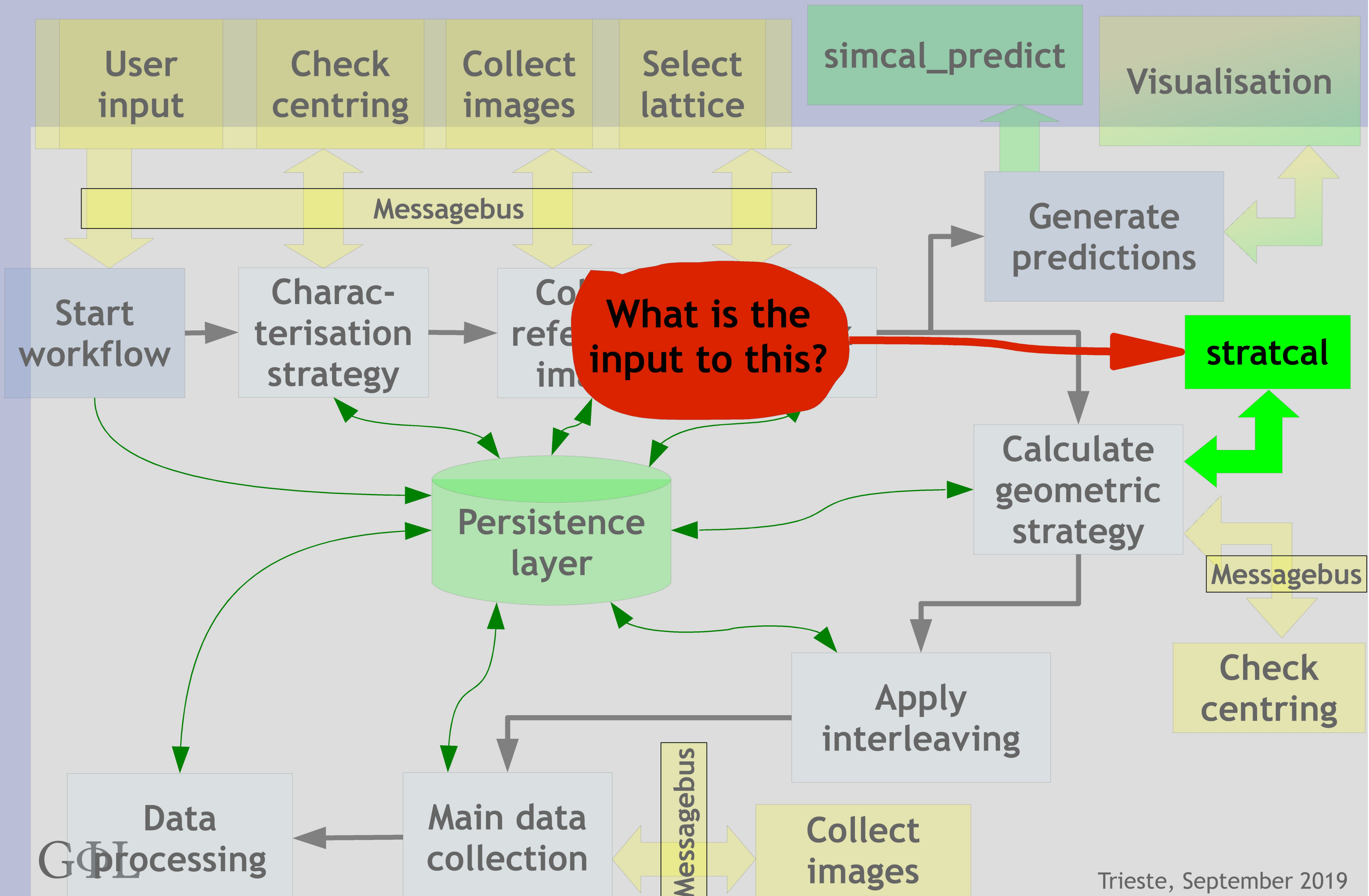
- Support the execution of workflows
  - Downstream steps have access to validated data generated by upstream steps, through a structured API
- Provide complete information for data processing
  - Final data processing is currently launched by the workflow, but it would be better to do this through existing auto-processing facilities
- Enhance archival of experimental information

Future link-up with ISPyB?



Calibration

Experiment with real beamline



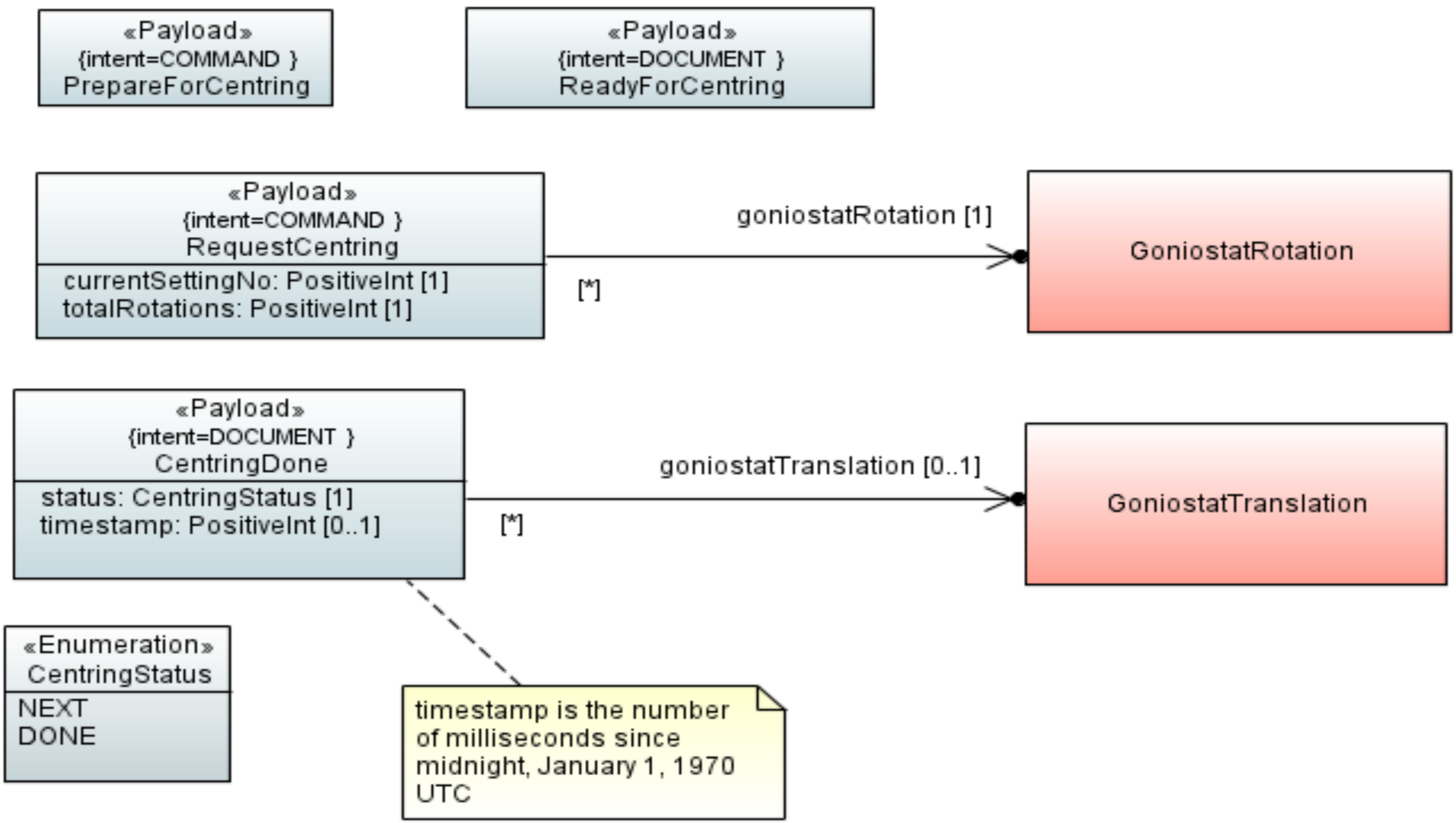
Input to stratcal

- Sample information:
  - Orientation, cell, symmetry
- Static instrumentation data:
  - Goniostat rotation/centring axes, detector geometry, beam direction
- Required for calculating viable high-quality strategies:
  - Accessible alignments, multiple orientations
  - Maintaining sample centring
  - Anticipating shadowing
  - Avoiding collisions
- Nominal data values are not good enough!

The requirement for calibration

- Available instrumentation data can be inaccurate or incomplete:
  - not always required for current standard procedures
- GΦL has developed two calibration workflows:
  - TransCal: sample centring axes
  - DiffractCal: detector geometry, goniostat rotation axes and beam direction
- These workflows are for beamline staff
  - to enable local maintenance of the instrumentation data that stratcal/MX Experimental Workflow requires
- Recent Abstract Beamline Interface developments explicitly cater for calibration

Abstract Beamline Interface: Translational Calibration



GΦL People

- Gérard Bricogne
- Rasmus Fogh
  - MXCuBE
- Claus Flensburg, Wlodek Paciorek
  - stratcal, simcal, calibration applications
- Clemens Vonrhein
  - autoPROC, connecting calibration with workflow