reduction

data treatment for ARCS

http://wiki.danse.cacr.caltech.edu/danse/index.php/Reduction
reduction

• tools to reduce raw, inelastic data from time-of-flight chopper spectrometer
• essential processing—must be done for every experiment
• validation is essential

• descended from Pharos IDL procedures
why change?

• information sources badly mixed
  – data formats, instrument specs, algorithms
  – not reusable

• monolithic
  – unit testing difficult
  – difficult/impossible to tinker with

• slow
gross testing useful only as long as it is not needed!
unit testing

• very good for you
• different methods: project must specify
• some elements of u. t.:
  – test every execution path
  – provide standard inputs, expected outputs
  – some method of comparing actual & expected outputs
why change?

• information sources badly mixed
  – data formats, instrument specs, algorithms
  – not reusable

• monolithic
  – unit testing difficult
  – difficult/impossible to tinker with

• slow
two approaches
why change?

- information sources badly mixed
  - data formats, instrument specs, algorithms
  - not reusable

- monolithic
  - unit testing difficult
  - difficult/impossible to tinker with

- slow
change to what?

• Object oriented
  – separate, equalize information sources
    • histograms, instrument data, transformations
• Finer granularity, more layered
  – simpler components, easier to test, replace
  – add scientific context to generic components
• Unit tests, integrate nexus, Pyre
example: layering

layers add context

basic layers have no scientific context: can be reused for *any* science

primary code

secondary code: tests, etc

C++

histograms

Python

std::vector

StdVector

histogram

S_QE
reduction structure

Python

C++

histograms, instruments, transformations
Histograms

Store and access n. s. data
• multidimensional data sets
• 200—1400 MB; 9000+MB?
need
• ability to associate metadata
• efficient iteration
• availability, stability

http://wiki.cacr.caltech.edu/danse/index.php/Simple_Histogram_class_idea
Histograms: C++

Storage based on STL vector class
- ANSI standard C++—universally available
- Optimized by compilers

Adaptor classes extend to
- Different storage schemes
- Different dimensionality

Low-level routines: arithmetic, average, etc.
Histograms: Python

Group C++ containers to model complex data sets
- *associate signal & error histograms, ...*

Add context, preferred behaviors
- *names, axes, units*
- *error propagation?*
Instruments

Find, store, and serve instrument data

• detector & monitor configuration, properties
  – pixel-sample distances, angles, dimensions, ...
• moderator, chopper, sample positions...

Need

• comprehensive description of instrument
• flexibility to describe different instruments

http://wiki.cacr.caltech.edu/danse/index.php/instrument_and_related_classes
for HDF/nexus I/O, see also
Instruments: C++

Abstract base classes parallel NeXus
Concrete subclasses specialize
- *To given instrument, detector type, etc.*
- *To information source (hard-coded, file, ...)*

STL-based containers implement sorting
- *e.g. pixels sorted by*
  - *pixel-sample distance*
  - *scattering angle(s)*
Instruments: Python

Interface to instrument descriptions
- Hard-coded
- NeXus or other files

Pass info to C++ layer
- Python is better suited to parsing tasks
- Also useful for instrument diagnostics and simulation
Transformations

Variable changes, reductions, slicing

- *time-of-flight to energy*
- *detector coord. to scattering angle (powder)*
- $S(Q_i, Q_j)$ for $Q_i \in \mathbb{R}(Q_x, Q_y, Q_z, E), ...$

Need

- *scientific integrity*
- *efficient (fast) operations*
- *flexibility*
Transformations: C++

Modular design

- *Building blocks perform simple tasks*
  - time to energy rebin
  - energy & phi to energy & |Q| rebin

Generic (iterator) interface

- *Independent of underlying container*
- *Independent of instrument*

Compiled, optimized for maximum speed
Transformations: Python

Driver routines

• coordinate simpler objects to perform complex tasks
  • EnergyRebinDriver, QRebinDriver
  • easily modified by user

Experiment with pipeline strategies

• e.g. reduce data one pixel at a time
  – Trivially parallel for much of process
  – Real-time refinement of ARCS data!
Transformations: integrity

test cases
• “trivial” tests: can be checked w/ pencil & paper
• application tests: produce “sensible” results from realistic, reference inputs.

numerics
• stability
• propagation of numerical errors
  – interval analysis?
  – num. errors always $\ll$ exprmntl (counting )errors?