

# Status of the SNS/DANSE Reduction Effort

Peter Peterson  
petersonpf@ornl.gov

# Development Models



# Who was there?

2006-11-28 at SNS

- Mark Hagen (SNS)
- Jiao Lin (CalTech)
- Mike McKerns (CalTech)
- Steve Miller (SNS)
- Peter Peterson (SNS)
- Michael Reuter (SNS)
- Tom Swain (SQRL at UT)



# What was discussed?

---

- Projects Scope
- Frameworks
- Data Objects
- Function Types
- Error Reporting/Logging
- Software Hosting



# Projects Scope

- SNS Scientific Computing Group
  - Data curation
  - Data **reduction**
  - First look analysis
  - Application hosting
- DANSE (**analysis** and simulation)
  - Diffraction
  - Engineering Diffraction
  - Small Angle Scattering
  - Reflectometry
  - Inelastic Scattering



# What is reduction?



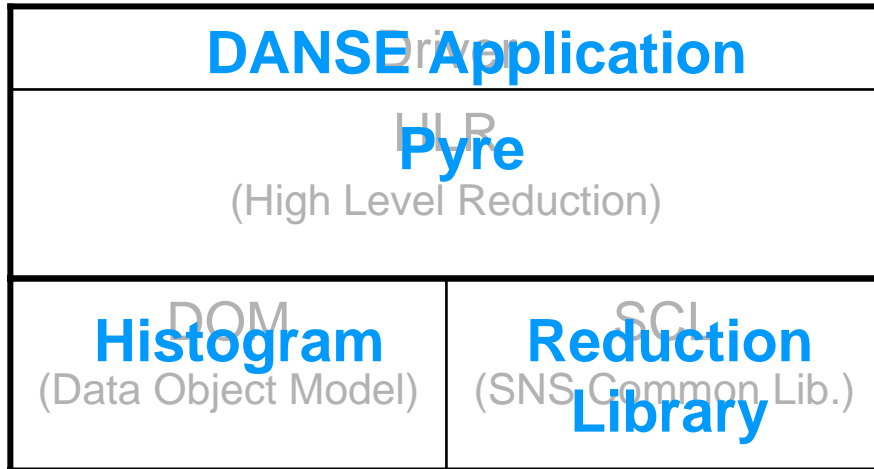
SNS 107030214-T00 0 01-R01

## Data Reduction Library Software Requirements and Specifications



July 2005

- Consult the instrument scientists for how they want reduction done
- Commit consensus to paper detailing atomic functions required
- The near-current set of requests is kept in the document shown here
- The near-current version can be found at [http://sns.ornl.gov/asg/projects/SCL/reqspec/DR\\_Lib\\_RS.doc](http://sns.ornl.gov/asg/projects/SCL/reqspec/DR_Lib_RS.doc)



## Three levels of reduction at SNS

- Level 1: **Driver** is the overall mechanism that runs the data reduction process. It based on the requirements that are given by the instrument scientists.
- Level 2: **HLR** is the representation of functions given by the requirements. It unifies calls to retrieve data and to call low level functions.
- Level 3:
  - **DOM** provides abstract layer for data manipulation.
  - **SCL** is a toolbox of reusable primitive functions necessary for data reduction process.

- SCL is a collection of functions derived from the Data Reduction requirements document.
- Functions have well defined input, action and outputs
- The functions are designed to use simple data structures
  - `Nessi::Vector` (PFL)
  - `<Type>NessiVector` (PBL)
  - `NessiList` (PAL)
- `Nessi::Vector` is currently a wrapper for `std::vector`
  - Allow for another container to be substituted without lots of code changes

## **Python Abstraction Layer (PAL):**

represents the Python code that lays over the top of the Swig generated Python binding functions and hides the details of those binding functions from the user.

## **Python Binding Layer (PBL):**

deals with the Swig generated Python – C++ binding layer that allows the Fundamental C++ functions to be used from Python

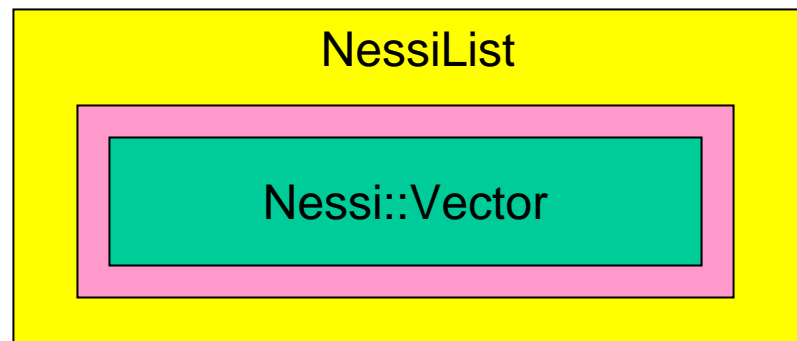
## **Primitive Functions Layer (PFL):**

includes all of the C++ code in the common libraries. The code is divided into 2 main parts:

- C++ code dealing with the fundamental library functions
- C++ code dealing with extra compiler tests

# “Framework independence”

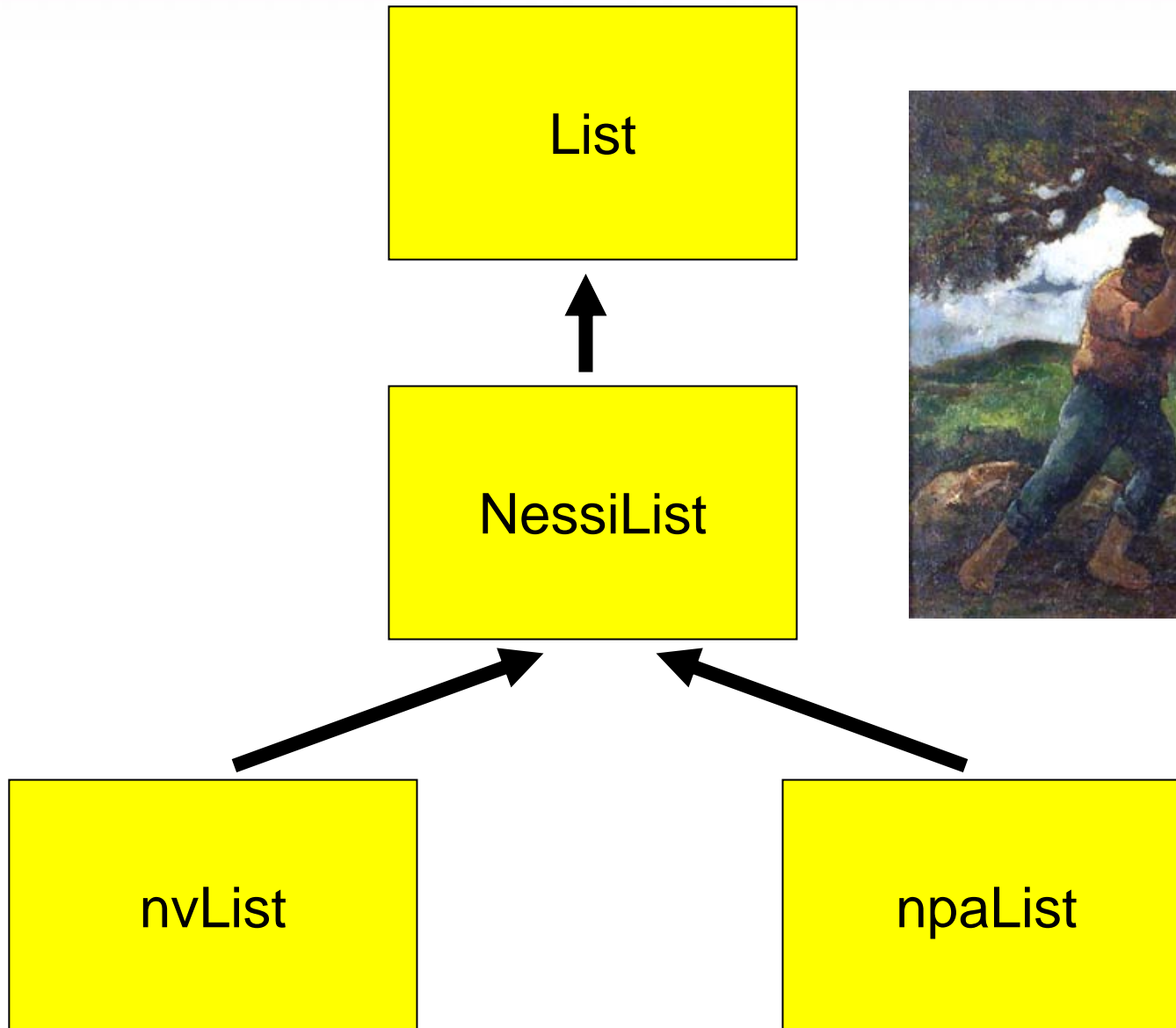
- “Common” data object
  - Scalars and arrays



Default constructor  
Copy constructor  
Sized constructor  
Forward iterator  
Random iterator  
Size operator

List

# “Low” level data object



# Library Tenets – The Seven Virtues



The C++ functions will follow a set of rules that will provide the application programmer with confidence that the library is well-formed, does not affect outside systems and does not maintain state. The rules are:

1. There will be no global/package variables.
2. There will be no side effects.
3. They will be thread safe.
4. They will neither allocate nor de-allocate any persistent resources. This allows for allocating (and de-allocating) temporary memory for calculations.
5. There will be at least one smoke test for every function.
6. Functions will create no heap memory. The functions will need to be passed the arguments, space for the results, and space for any temporary storage needed. To notify the caller of how much memory is needed for temporary space, any function that uses temporary space may have a related utility function that returns information on the amount of temporary storage needed. If the user passes in null pointers for the temporary storage the function is intended to allocate and de-allocate what is needed.
7. The functions will generate a warning when non-fatal errors are encountered. (Fatal errors throw exceptions)

# SCL Functions I



Section	Function	BSS	REF
3.1	Add Scalar to Array	N	Y
3.2	Subtract Scalar from Array	Y	N
3.5	Divide Array by Scalar	Y	N
3.6	Add Two Arrays	Y	Y
3.8	Multiply Two Arrays	N	Y
3.9	Divide Two Arrays	Y	Y
3.10	Add Arrays Weighted by Uncertainties	Y	Y
3.11	Reverse Array	Y	Y
3.12	Rebin 1D Data	Y	Y
3.13	Rebin 2D Data	Y	N
3.15	TOF to Wavelength	Y	Y
3.22	Wavelength to Energy	Y	N
3.24	Wavelength to Scalar Wavevector	Y	N
3.29	Initial Wavelength for Inv Geom Spec	Y	N
3.30	Energy Transfer	Y	N
3.33	Inc. and Scatt. Wavevector to Scalar Q	Y	N
3.38	Dead-Time Correction	Y	N
3.42	Fit 2D Reflectometer Background	N	Y
3.43	Fit Linear Background	Y	N
3.45	Calibrate Reflectometer Area Detector	N	Y

# SCL Functions II



Section	Function	ARCS	CNCS	POWGEN3	EQ-SANS
3.14	Rebin 4D Data	Y	Y	N	N
3.17	TOF to Scalar Q	N	N	Y	Y
3.19	Initial Velocity DGS	Y	Y	N	N
3.21	Velocity to Energy	Y	Y	N	N
3.23	Velocity to Scalar k	Y	Y	N	N
3.25	Wavelength to d-spacing	N	N	Y	N
3.26	Time Offset DGS	Y	Y	N	N
3.27	TOF to Final Velocity DGS	Y	Y	N	N
3.31	Frequency to Angular Frequency	Y	Y	N	N
3.32	Inc. and Scatt. Wavevector to Q	Y	Y	N	N
3.36	Absorption and Multiple Scattering	N	N	Y	N
3.37	Matrix Multiplication	Y	Y	N	N
3.39	D-spacing to TOF Focused Detector	Y	Y	Y	N
3.40	Inelastic Scattering Correction	N	N	Y	N
3.41	Incoherent Scattering Correction	N	N	Y	N

# New Functionality

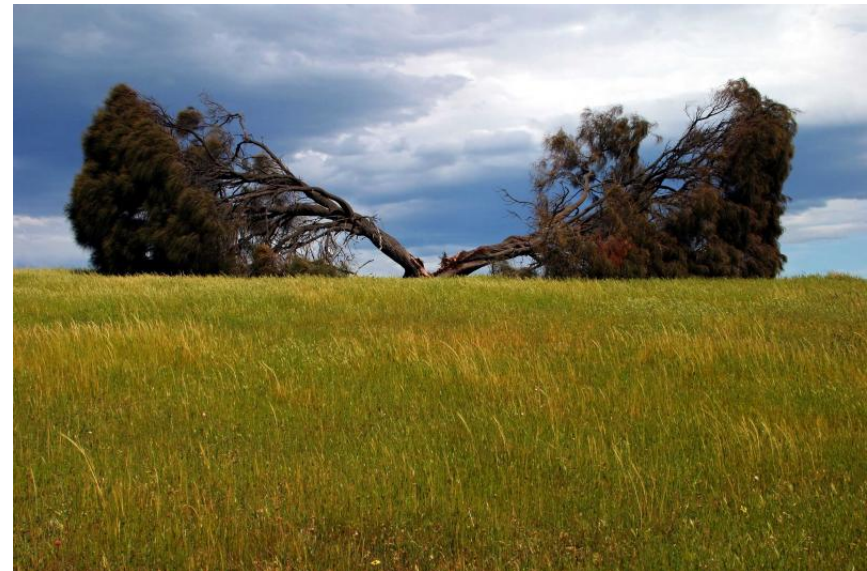
---

1. Prototype in python
2. Finalize Algorithm
3. Convert to C++ for performance
4. Integrate with existing library hosted at SNS



# Error reporting/logging

- Logging/warnings
  - In C++ an object that implements “operator<<”
  - In python has “simple” operation that collects logs
- Errors
  - Exceptions with preference on native types



# Remaining “big” issues NOT discussed

- Finalize low level object interface
- Finalize logging interface
- High level data object
- File I/O
- User Interface



# Working group

---

- Paul Kienzle
- Jiao Lin
- Dennis Mikkelson
- Peter Peterson
- Tom Swain
- unidentified ISIS person

