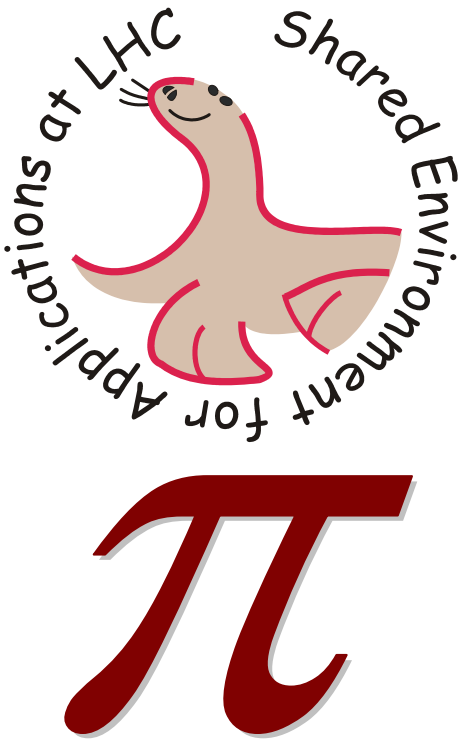
SEAL and PI Project Status



LHCC Comprehensive Review of LCG

22-23 November 2004

P. Mato / CERN

Contents

- Project Overview
- SEAL Work Packages
  - Foundation, Framework, Math Libraries, Dictionaries, Scripting, Documentation
- PI Status
- Manpower Situation
- Status and Milestones
- Summary

SEAL Overview

- Provide the software infrastructure, basic frameworks, libraries and tools that are common among the LHC experiments
  - Select, integrate, develop and support foundation and utility class libraries
  - Develop a coherent set of basic framework services to facilitate the integration of LCG and non - LCG software
- Scope
  - Foundation Class Libraries
    - » Basic types (STL, Boost, CLHEP, …), utility libraries, system isolation libraries, domain specific foundation libraries
  - Mathematical Libraries
    - » Common set of mathematical libraries
  - Basic Framework Services
    - » Component model, reflection, plugin management, incident (event) management, scripting

SEAL Project Work Packages

Foundation	Foundation and Utility Libraries and Plug-in Manager
MathLibs	Math Libraries Support and Coordination
Dictionary	LCG Object Dictionary
Framework	Component Model and Basic Framework services
Scripting	Scripting Services
Documentation	Education and Documentation

Foundation

- Provide support for foundation, utility and operating system isolation libraries
- Inventory of existing libraries
  - http://seal.cern.ch/components.html
- Main external library: Boost
  - Being adopted by experiments and LCG AA projects
- Developed SEAL utility and system libraries complementary to Boost and STL from existing code
  - SealBase - library containing a large variety of utility classes
  - SealIOTools - library containing utility classes for stream oriented I/O
  - SealZip - library for compression I/O and producing archive files
- Plugin Manager
  - Basic concept: advanced object factory, dynamic loading of plugins
  - Two simple interfaces: object instantiation, plug-in provider

Foundation Plans

- Not foreseen major new developments in this work package
  - The work needed is mainly to educate users by developing and setting up tutorials, user-guides and coaching developers
- Entering maintenance phase
  - Additional functionality only on direct demand
  - Further reduction of unnecessary dependencies in external packages

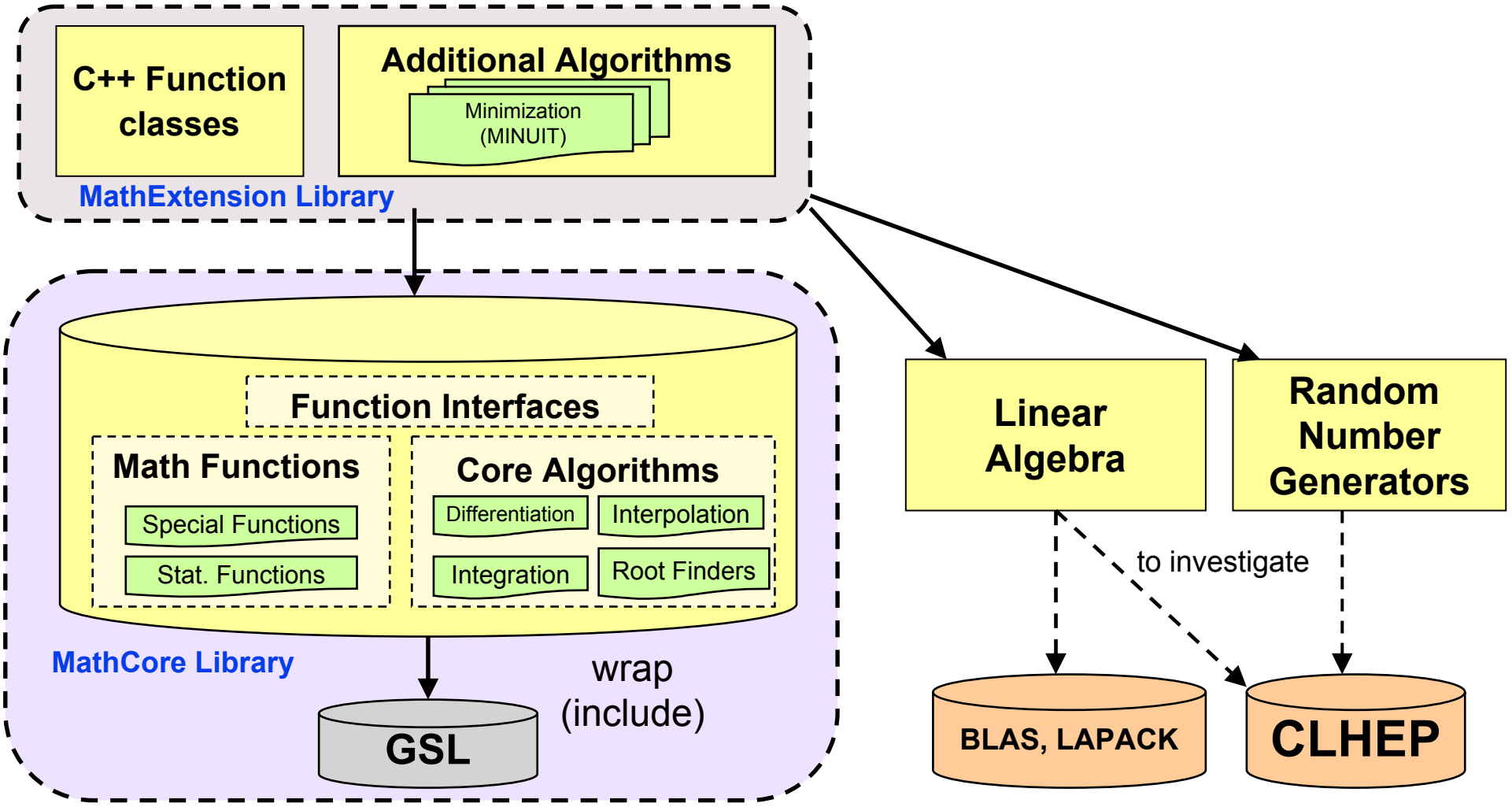
Math Libraries

- Aim is to provide a coherent set of Mathematical Libraries to end-users and developers of LHC experiments
- Requirement to use the same core library in all environments
  - simulation, reconstruction and analysis
  - from C++ and interactively (Python, CINT) via C++ dictionary
- Avoid duplication and maintenance burden of similar libraries
- Collaboration with experiments, ROOT and Geant4

Math Libraries Contents

- Selected Functionality (non-exhaustive):
  - Evaluation of Special and Statistical functions (Pdf)
  - Numerical Algorithms (Integration, Differentiation, Interpolation, Minimization, etc..)
  - Linear Algebra
  - Random Number generators and distributions
- Avoid duplicating functionality already present in STL (complex, search and sorting algorithms) and non-mathematical functionality
- Detailed inventory available at:
  - » http://www.cern.ch/mathlib/mathTable.html

C++ Mathematical Libraries



Mathematical Libraries Organization

- A Core Library (MathCore) with most used functionality:
  - Small library (order of 1MB size) containing
    - » Special and Statistical functions
    - » Core algorithms
    - » Basic function interfaces for algorithms
- Linear Algebra library with vector and matrix classes and their operations
- Random number generator and distributions
- Extension Library (one or maybe more lib's)
  - Sophisticated numerical algorithms, like Minimization
    - » Dependency on Linear Algebra or Random Numbers
  - C++ classes for math functions and their operations
    - » Arithmetic (+,-,*,/), composition and convolution
- And Dictionary libraries for interactivity and persistency

Math Libraries Current Status

- Developed a C++ MathCore library prototype
  - Based on GSL for majority of functionality
  - Can be built together with the needed GSL source code
    - » have a library without external dependency (ROOT requirement)
  - Library include:
    - » ~ 20 Special functions (Bessel, Erf, etc…) implemented following C++ Standard proposal
    - » Most used probability distributions (Chi2, Landau, etc..)
    - » Numerical integration, differentiation and root finders
    - » Generic functions interfaces
- Next step is to give to users for feedback and to test the integration with ROOT

Tests and Validation studies

- Performed extensive evaluation of GSL
  - Test numerical quality and performance of special functions
    - » Comparison with ROOT and NagC
    - » Good results obtained by GSL for most used functions
- Random number generator tests
  - New tests designed for correlations and to detect non-random sequences
- Linear Algebra performance tests
  - Comparison of various packages (CLHEP, GSL, BLAS/LAPACK, ROOT, UBLAS) in matrix operations used in track fitting

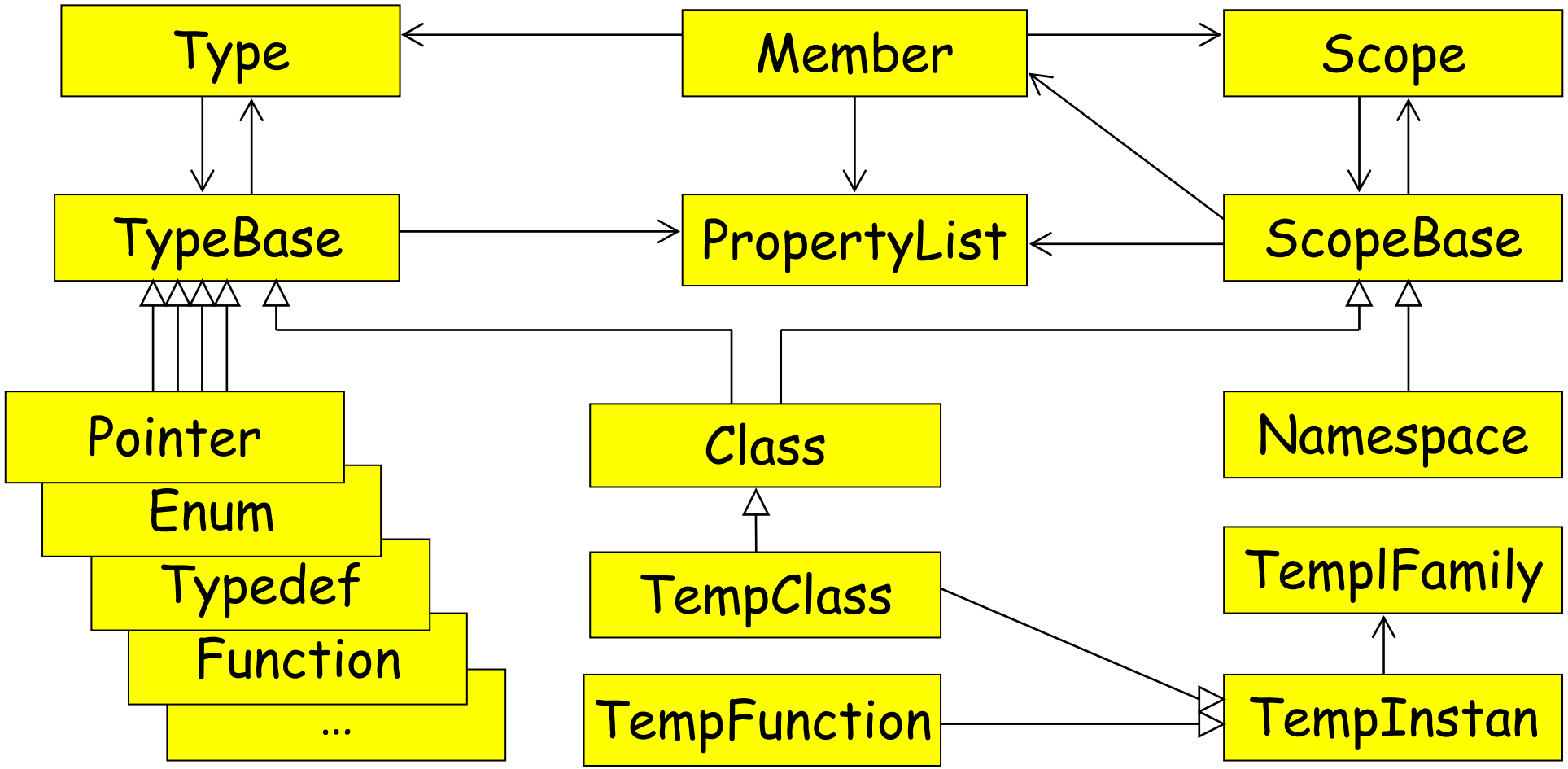
Fitting and Minimization

- Completed major developments of C++ Minuit
  - Reached same functionality as in Fortran version
  - Performed validation tests
    - » Same numerical accuracy and CPU performances
  - Working on adding a new minimizer algorithm (FUMILI)
- MINUIT C++ is used by CMS reconstruction and end-users
  - Stand-alone package which can be built independently
- Provided a Fitting library (FML) for standard fitting problems based on MINUIT
  - Have also Python interface for interactive users

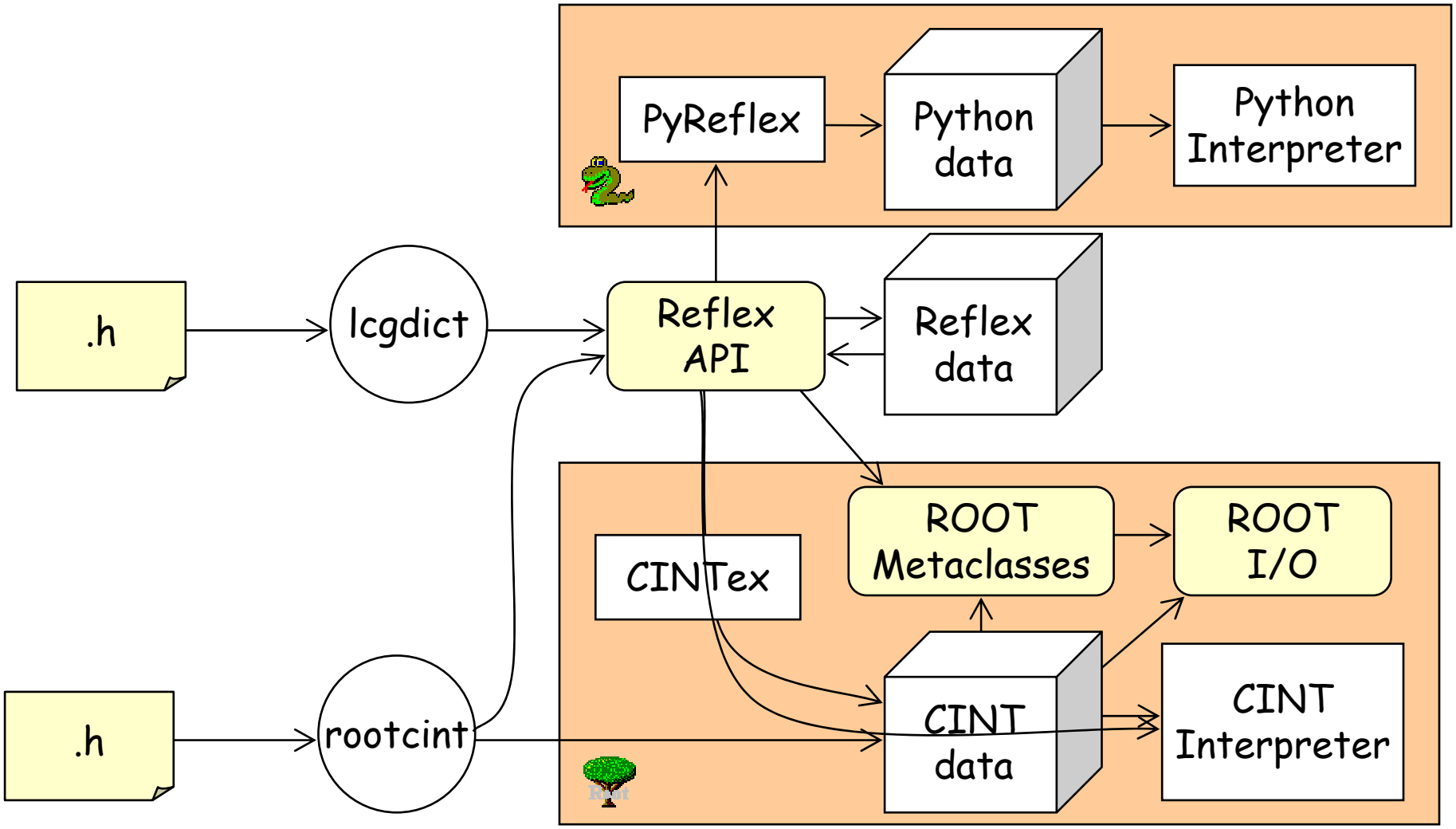
Dictionary

- C++ Reflection
  - Reflection is the ability of a language to introspect it's own structure at runtime and interact with it in a generic way
  - A dictionary provides reflection information about types of a certain language to the user
- Current version - Reflection(Builder) - in production
  - POOL & SEAL PyLCGDict
- New version - Reflex – released
  - API done in collaboration with the ROOT team
  - The idea is a common dictionary between SEAL & ROOT
  - Several enhancements (e.g. references, enums)
  - Closer to the C++ ISO standard

Reflection Model



Reflex and ROOT



Reflex/ROOT Next Steps

- Fill CINT data structures (data and methods) from Reflex on demand.
  - This is needed to allow interactive work using CINT (ROOT) for classes for which only the Reflex dictionary exists. The code for this already exists in POOL for the LCG/CINT dictionary gateway.
  - Estimated to 3-4 months
- Re-implement the ROOT metaclasses (TClass, TMethod, etc.) on top of Reflex
  - Estimated to 2 months
- Adaptation of the CINT interpreter to run on top of Reflex directly is foreseen in principle but detailed planning will only be done after tasks 1 and 2 are completed.

Framework

- Component Model
  - Hierarchy of bases classes to support the component model
  - User classes inherit from *Component* or *Service*
  - Plug-in functionality for free

- The first set of Basic Services came with the new Component Model
  - Application (Defines the top level Context)
  - Message Service (Message composition, filtering and reporting)
  - Configuration Service (Management of Component properties and loading configurations)

Framework Status

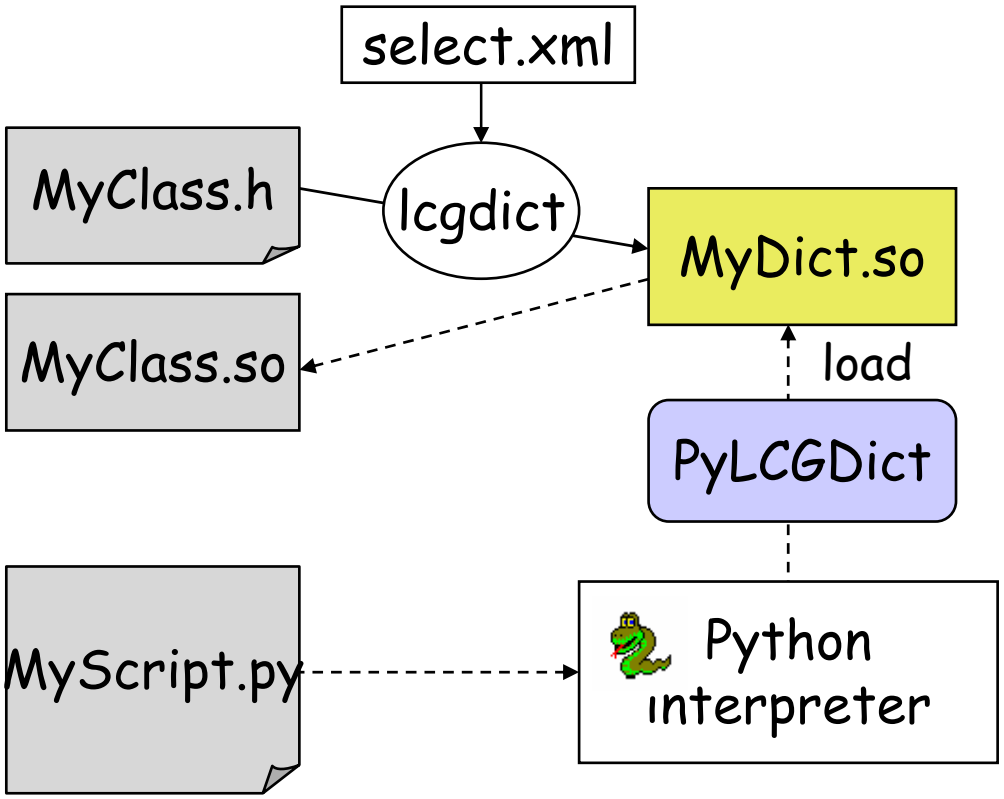
- The objective has been to integrate SEAL component model into the existing Gaudi/Athena framework and evaluate its costs and benefits
  - Not yet done due to lack of manpower (LHCb & ATLAS)
  - The development of new services was put on hold until there is a firm commitment from at least 2 experiments
- The new developments in POOL are exploiting the use of SEAL component model to implement true components and services together with the component loader
  - POOL Relational Abstraction Layer set of components

Scripting: Python

- Recommendations for developing Python bindings
  - Architects Forum selected to use Boost.Pyhton
  - PyLCGDict provides an alternative approach
- PyROOT
  - Provides access to ROOT functionality from Python
  - Uses ROOT/CINT dictionary
  - Removed any external dependency and performance improvements
  - Integrated and distributed with ROOT
- PyLCGDict
  - Access to C++ libraries from Python
  - Uses LCG dictionary. Automatically generates Python proxies for C++ objects dynamically.
  - Mapping C++ constructs to Python natural constructs

PyLCGDict : Mode d'emploi

- A "dictionary" library is produced from class definitions (.h files)
  - Description of the class
  - "stub" functions to class methods
- Absolutely non-intrusive
- The PyLCGDict module does the adaptation between Python objects and C++ objects in a generic way
  - It works for any dictionary



PyLCGDict Status

- Used mainly by LHCb, ATLAS, PI, etc.
  - ATLAS: Athena application configuration, simulation configuration, distributed analysis (DIAL), interactive analysis (AOD), etc.
  - LHCb: interactive Gaudi, physics data analysis (Bender), event display (Panoramix)
  - PI: gluing various systems (ROOT, HippoDraw, etc.)
  - Geant4: Geometry import/export (GDML+Python)
- Adapting to new Reflex API
  - Ongoing work adapting to the new API (PyReflex)
  - Better C++ coverage

Documentation

- SEAL Workbook
  - Collection of very practical web pages to teach how to use the various SEAL components
  - Initial version available since June
  - http://seal.web.cern.ch/seal/snapshot/workbook/
- Reference Documentation
  - Doxygen and LXR generated
- Python Courses
  - Provide assistance in the use of Python
  - 3 day course: Hands-on Introduction to Python Programming
  - Available through CERN Technical Training programme

Software Process Issues

- SEAL is a collection of fairly independent packages but there are difficulties to release the "parts" independently
  - The current build and configuration tools does not facilitate the task
  - Ongoing work in collaboration with SPI to solve the problem
- Fairly compete unit test suit
  - 270 independent test programs
  - Regularly run under the QmTest test driver on all supported platforms

Current Status of PI

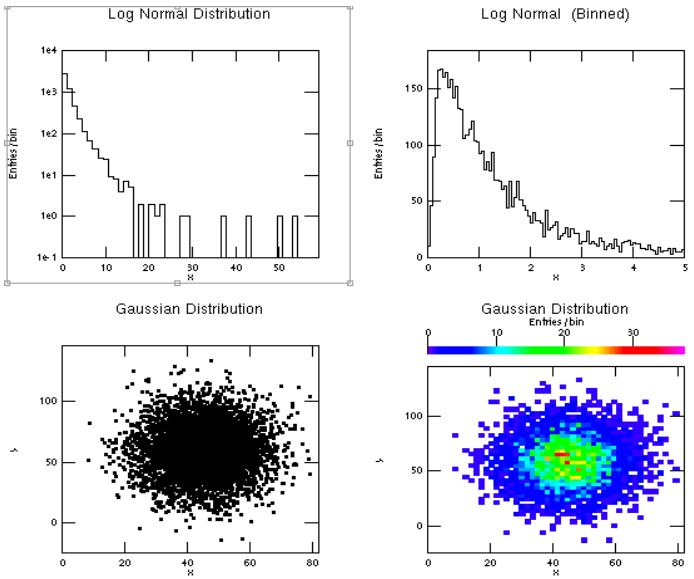
- Developments of the Analysis Services component is completed
  - Set of component libraries implementing AIDA interfaces
  - Flexible: choose implementation at runtime using plug-in manager system from SEAL
    - » ROOT and Native implementations for histograms
    - » Storage (I/O) for Histograms and Tuples in ROOT, HBook and XML compressed format
- Provide easy conversion between all formats
  - Well tested, large number of unit tests
    - » Failures due to differences in implementations
  - Provided Python bindings to AIDA interfaces using LCG Dictionary and PyLCGDict from SEAL

Current Status of PI (2)

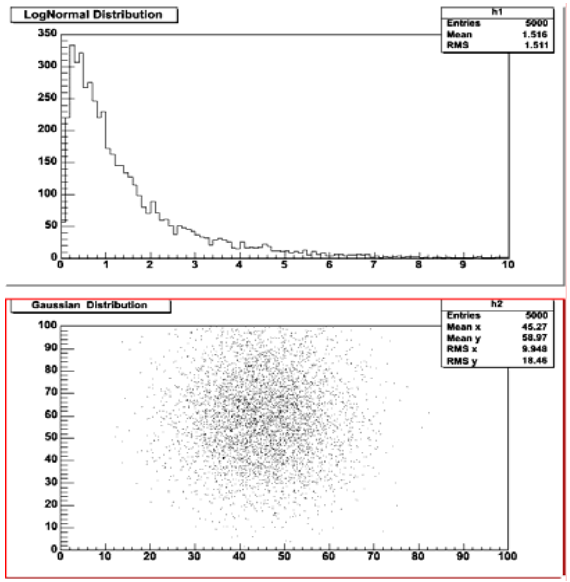
- Ongoing maintenance
  - Low effort, mainly bug fixes
  - Release new versions following SEAL, AIDA, etc. releases
- Users of PI :
  - Histogram libraries are used by Gaudi (LHCb + ATLAS) and CMS
    - » LHCb is evaluating to use also AIDA Tuple libraries
  - Used by Geant4 in advanced examples for storing histograms and tuples
  - Python layer is used by some physicists for analysis (fitting, etc…)
- Need to provide customized download and installation for external users

Status of PI: interoperability

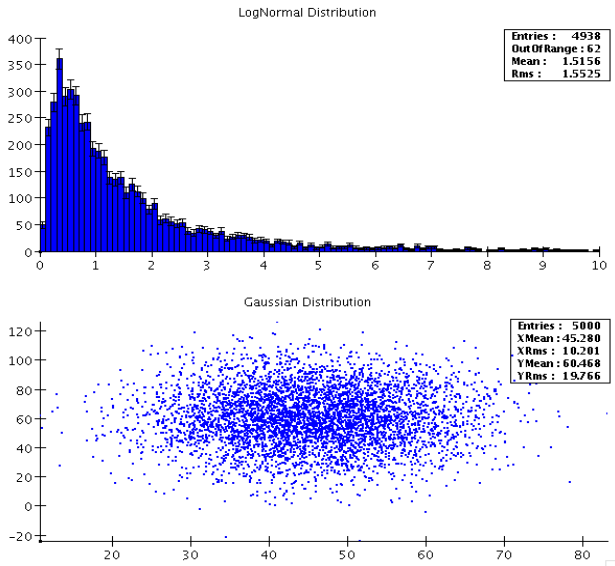
- Easy interoperability with external tools
  - Can display histogram in Python using ROOT, JAS and HippoDraw



HippoDraw

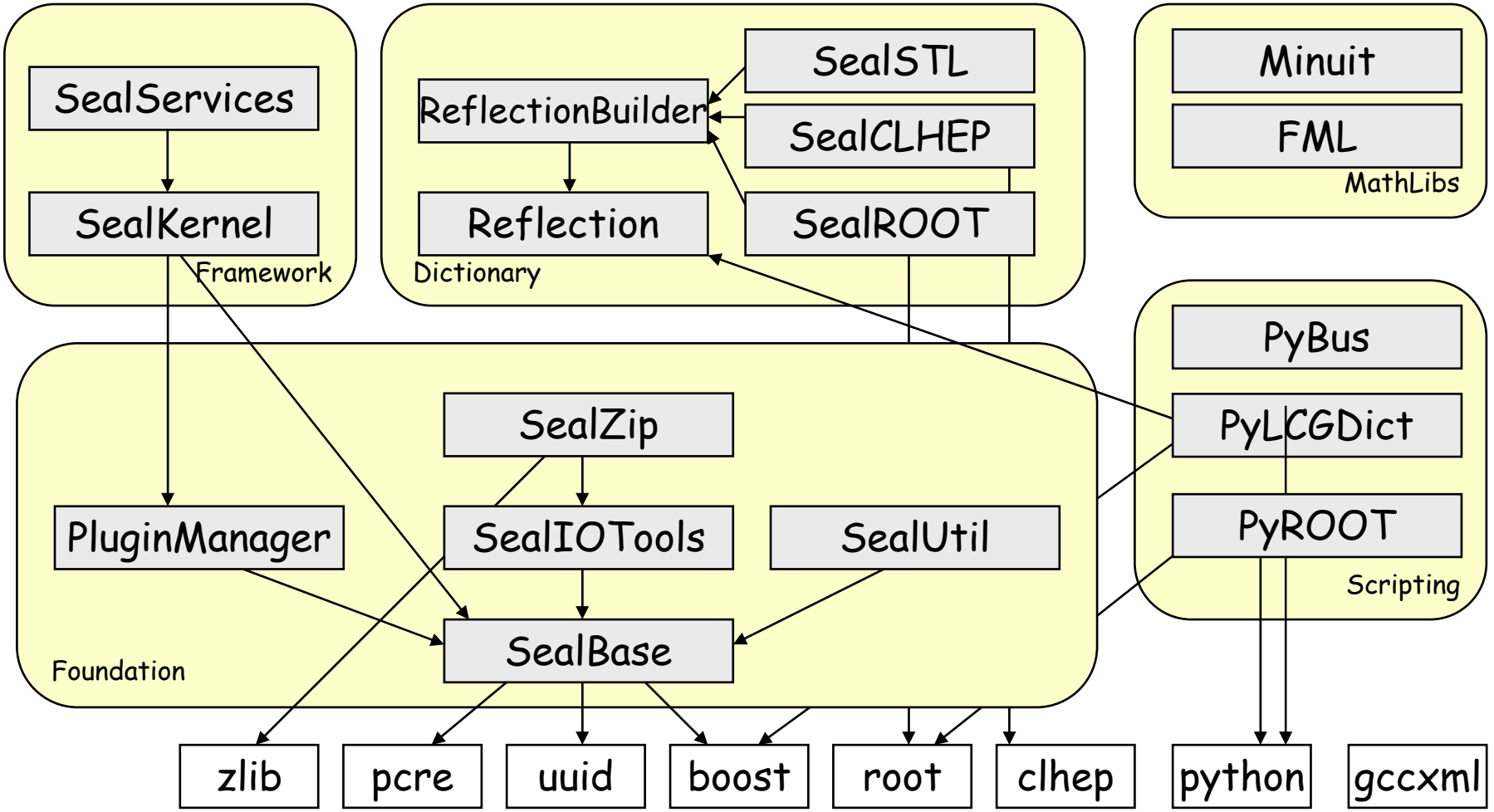


ROOT



JAIDA (JAS)

Packages and Dependencies (v1.4.2)



Release Road Map

Release	Date	Status	Description (goals)
V 0.1.0	14/02/03	internal	◆Establish dependency between POOL and SEAL ◆Dictionary generation from header files
V 0.2.0	31/03/03	public	◆Essential functionality sufficient for the other existing LCG projects (POOL) ◆Foundation library, system abstraction, etc. ◆Plugin management
V 1.0.0	18/07/03	public	◆Essential functionality sufficient to be adopted by experiments ◆Collection of basic framework services ◆Scripting support
V 1.1.0	05/09/03	public	◆Corrections and improvements of Framework
V 1.2.0	16/10/03	public	◆Support for ICC and VC++ compilers
V 1.3.0	25/11/03	public	◆Improvements in Plugin Manager ◆Consolidation Dictionary and Minuit
V 1.3.4	29/03/04	public	◆Bug fixes
V 1.4.0	23/06/04	public	◆Simpler plug-in manager, consolidation component model ◆New packages PyLCGdict2, FML, Workbook
V 1.4.2	02/11/04	public	◆Bug fixes ◆New package Reflex

SEAL Products and their Usage

		ATLAS	Alice	CMS	LHCb	Other
Foundation	SealBase					
	SealZip					
	SealIOTools					
	PluginManager					
Framework	Component Model					
	Basic Services					
Dictionary	Reflection					
	Lcgdict tool					
	Specific Dictionaries					
Scripting	PyROOT					
	PyLCGDict					
	PyBus					
MathLibs	Minuit					

In use directly In use indirectly Planned use

Milestones 2004

2004/6/1	Late	SEAL icc-64 test build support
2004/6/15	Done v=15	Workbook for SEAL
2004/6/30	Done v=65	New Dictionary API and reference implementation
2004/7/15	Done v=0	Mathlib project web
2004/10/1	Late	First version of the C++ mathlib package

Manpower Situation

- Current SEAL manpower: ~5 FTE

Foundation	Lassi Tuura (5%)
MathLibs	Lorenzo Moneta (90%), András Zsenei (100%)
Dictionary	Stefan Roiser (90%)
Framework	Radovan Chytracek (20%), Lassi Tuura (5%)
Scripting	Jacek Generowicz (80%), Massimo Marino (50%)
Documentation	Jacek Generowicz (20%)

- Current PI manpower: ~0.2 FTE
- Missing by end-2005: 1.5-2 FTEs for remaining development, experiment integration, support, ROOT convergence (Dictionary, MathLibs)

Summary

- SEAL has delivered a number of components that constitutes the basic foundation and utility libraries and object dictionary
  - Most of the delivered components are already in use or being tested and planned to be in use by LHC experiments
- Ongoing development in two areas mainly:
  - New C++ Reflection system with the goal to achieve a single dictionary between ROOT and LCG applications
  - Coherent set of common Mathematical Libraries
- PI development is basically finished, entering maintenance phase
- Advocating scripting based on Python
  - Peaceful coexistence between CINT and Python
  - Powerful tools have been developed (PyROOT, PyLCGDict)
  - Feedback from early adopters is encouraging (LHCb, ATLAS)