SEAL Project Status

LCG Applications Area Internal Review
30 March – 1 April 2005

Contents

- Project Overview
  - Internal review recommendations
- SEAL Work Packages
  - Foundation, Framework,
    Math Libraries, Dictionaries, Scripting,
    Documentation
- Manpower, 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

AA Review Recommendations

- Foundation Libraries recommendations
  - Setup tutorials, user-guides and help developers → partially done
  - Remove unnecessary dependencies in external packages → not done
  - Convergence between the SEAL and ROOT plug-in manager → not in the Plan
- Math libraries recommendations
  - Concerns about the future of Minuit, GSL and CLHEP → ongoing work
  - Careful testing to guarantee reliable physics results → done
  - Provision of a coherent set of libraries including dictionaries → ongoing work
- Dictionary recommendations
  - Concerns about the size of dictionaries → already done
  - Encourage unifying the dictionary with ROOT/CINT → ongoing work
- Framework recommendations
  - Discussion with the experiments to evaluate interest in framework → in the Plan
- Scripting recommendations
  - Evaluate the interest in the interoperability between Boost and Swig before any work is committed → already done
  - Continue with the PyLCGDict automatic binding → done
  - Seek feedback from the experiment physics community in usability of python in interactive analysis → on going work

Foundation

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

- Was not foreseen major new developments in this work package
  - The work needed has been mainly to educate users by developing and setting up tutorials, user-guides and coaching developers
- Started maintenance phase
  - Additional functionality only on direct demand
  - Further reduction of unnecessary dependencies in external packages
    - » Meanwhile building of the externals has been automated (SPI)
    - » The code itself can be found in CVS under seal/Imports

Framework

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 Access Layer set of components

Math Libraries

Lorenzo Moneta / CERN

Math Libraries

- Following 2003 review recommendations a new program of work was defined in 2004
- 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
- Perform extensive tests and validation of mathematical library to guarantee reliable physics results
  - evaluation of GNU Scientific Library (GSL)

Math Libraries Functionality

- Selected Functionality (non-exhaustive):
  - Evaluation of mathematical functions
    - » Special functions (Bessel,etc.) and statistical functions (pdf)
  - Numerical Algorithms
    - » Integration, Differentiation, Interpolation, Minimization, etc..
  - Linear Algebra
    - » Vector and matrix classes and their operations
  - Random Number generators and distributions
- Detailed inventory available at:
  - » http://www.cern.ch/mathlib/mathTable.html
  - » Links to SEAL MathLibs, CERNLIB, GSL and ROOT
    - GSL provides almost all the needed functionality

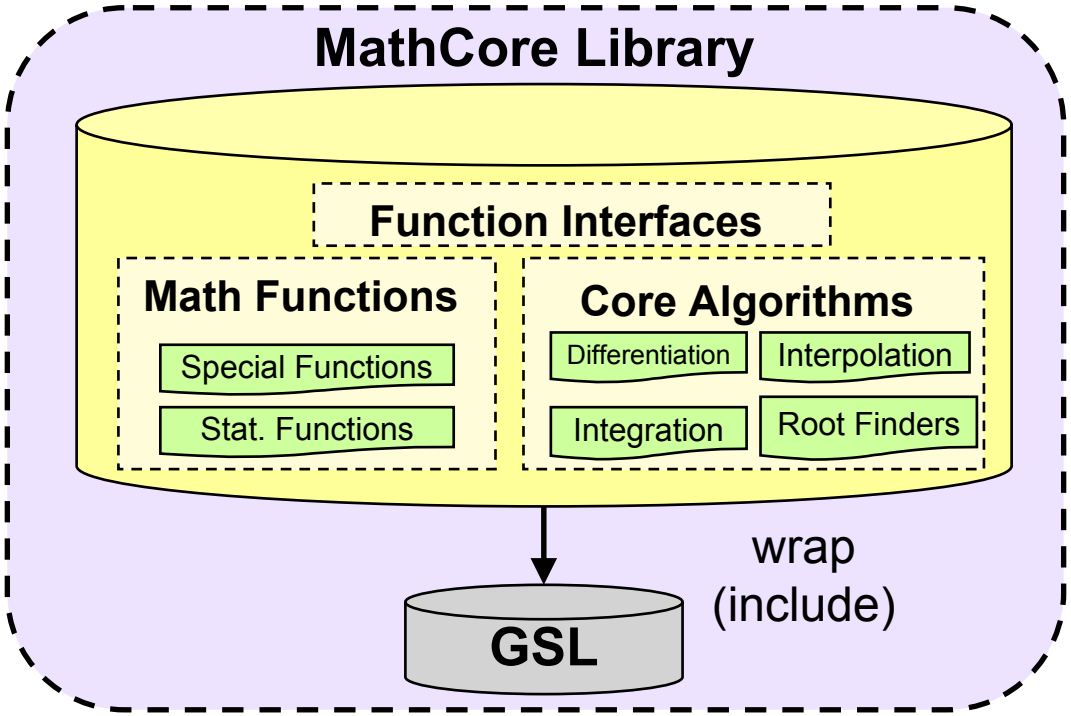
SEAL Mathematical Libraries

- Released Math Libraries (in SEAL 1.6)
  - MathCore
    - » Library with basic functionality, using GSL in implementation
  - Minuit
    - » Re-implementation of Fortran Minuit in C++
    - » Stand-alone package (no ext. dependency)
  - FML
    - » Defines generic interfaces for fitting and minimization
      - Minimizer interface implemented using C++ Minuit
  - PyFML
    - » Python interface to FML using SEAL PyLCGDict2

MathCore Library

- C++ Library containing most used functionality
  - Based on GSL (GNU Scientific Library)
  - Can be built together with the needed GSL code
    - » No external dependency
- Given to LHC experiments and ROOT
  - Detailed review done by CMS with very useful feedback

    (M. Paterno & W. Brown)



MathCore content

- Mathematical functions (free functions in namespace *mathlib*):
  - ~ 20 special functions (Bessel, Beta, Gamma, Erf, etc..)
  - ~ 50 functions used in statistics (pdf, cdf and inverse)
- Numerical algorithms
  - Integration (6 different adaptive integration algorithms)
  - Differentiation (3 algorithms)
  - Root finders (6 algorithms)
  - 1D minimization (2 algorithms)
  - Interpolation (4 algorithms)
- Generic function classes and interfaces
  - Generic and parametric function interfaces
  - Concrete function classes (polynomial function)

MatCore Documentation

- Produced Doxygen reference documentation
  - Example: Bessel functions

double mathlib::cyl_bessel_i(double *nu*,
double *x*
)

Calculates the modified Bessel function of the first kind (also called regular modified (cylindrical) Bessel function).

$$I_\nu(x) = i^{-\nu} J_\nu(ix) = \sum_{k=0}^{\infty} \frac{(\frac{1}{2}x)^{\nu+2k}}{k!\Gamma(\nu+k+1)}$$

for $x > 0, \nu > 0$. For detailed description see Mathworld. The implementation used is that of GSL.

Definition at line **201** of file **SpecFunc.cpp**.

double mathlib::cyl_bessel_j(double *nu*,
double *x*
)

Calculates the (cylindrical) Bessel functions of the first kind (also called regular (cylindrical) Bessel functions).

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (\frac{1}{2}x)^{\nu+2k}}{k!\Gamma(\nu+k+1)}$$

For detailed description see Mathworld. The implementation used is that of GSL.

Definition at line **212** of file **SpecFunc.cpp**.

MINUIT

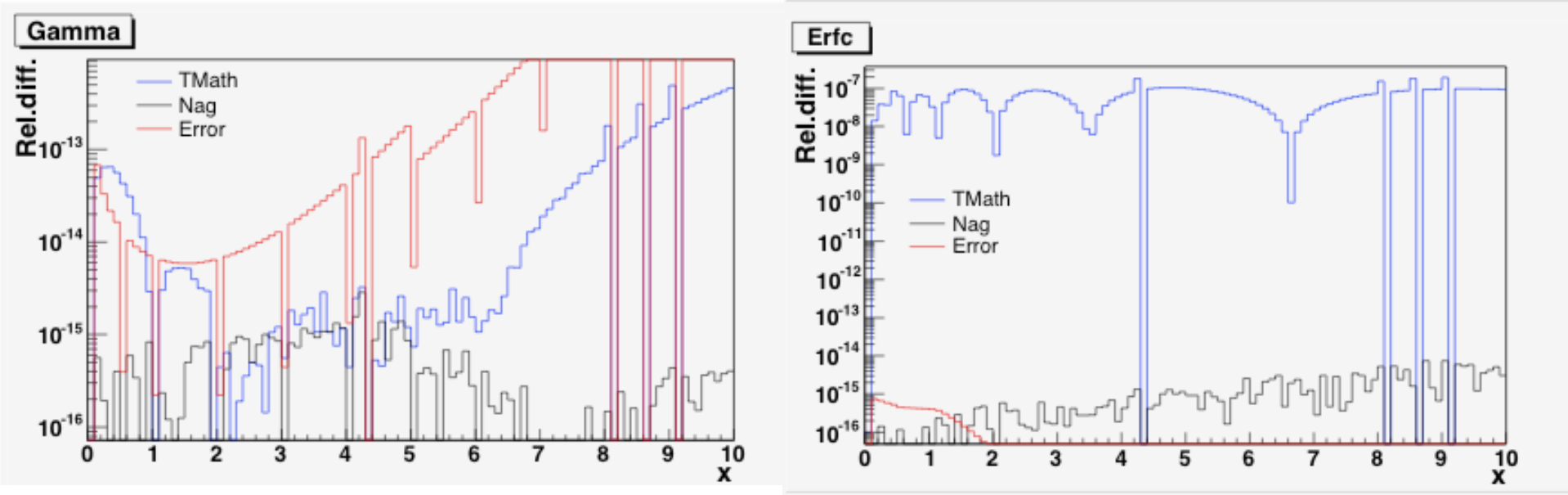
- Completed major developments of C++ Minuit
  - Reached same functionality as in Fortran version
  - Extended adding:
    - » Single side parameter bounds
    - » FUMILI algorithm
      - Specialized method for Chi2 and likelihood fitting
- Performed validation tests
    - » Same numerical accuracy and CPU performances
- MINUIT C++ is used by CMS reconstruction and end-users
  - Stand-alone package which can be built independently

FML

- Fitting and Minimization Library library
  - Provides general way of fitting data using various fitting methods and minimization engines
  - Defines generic interfaces for fitting and minimization
    - » Have implementation based on new MINUIT
      - with all Minuit minimizers (Migrad, Simplex, Fumili)
    - » For testing we have also implementations based on NagC and Fortran Minuit
  - Support for standard fitting methods (Chi2, Maximum Likelihood,..)
  - Very efficient in term of performances
- Have also Python interface for interactive users (PyFML)

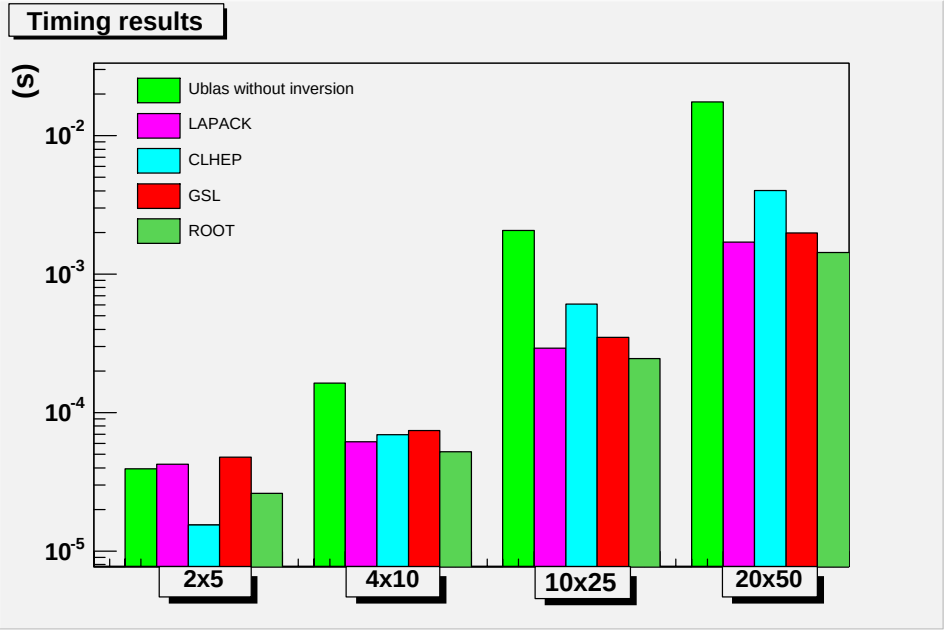
Tests and Validation studies

- Performed extensive evaluation of GSL
  - Test numerical quality and performance of mathematical functions
    - » Comparison with ROOT and NagC and also Mathematica
    - » Good results obtained by GSL for most used functions



Test and Validation Studies (2)

- Random number generator tests
  - New tests designed with the help of F. James to detect correlated effects (frequency test) or defects in the random sequence (orbit test)
    - » All major generator of GSL (Mersenne Twister, RanLux, etc..) passed the tests
- Linear Algebra performance tests
  - Comparison of various packages in operations used in track fitting
    - » CLHEP, GSL, BLAS/LAPACK, ROOT and Boost UBLAS



Dictionaries

Stefan Roiser / CERN

Dictionary

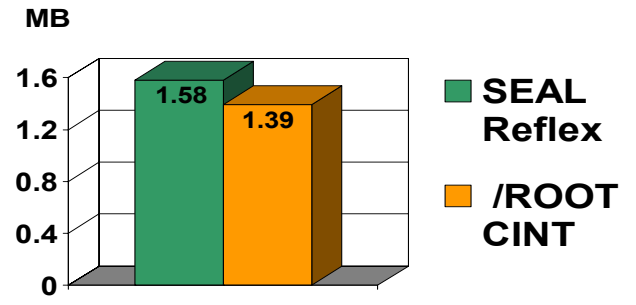
- C++ Reflection
  - Reflection is the ability of a language to introspect it's own structure at runtime and interact with it in a generic way
  - A dictionary provides reflection information about types of a certain language to the user
- Current version - Reflection(Builder) - in production
  - POOL & SEAL PyLCGDict
- New version - Reflex – released
  - API developed 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

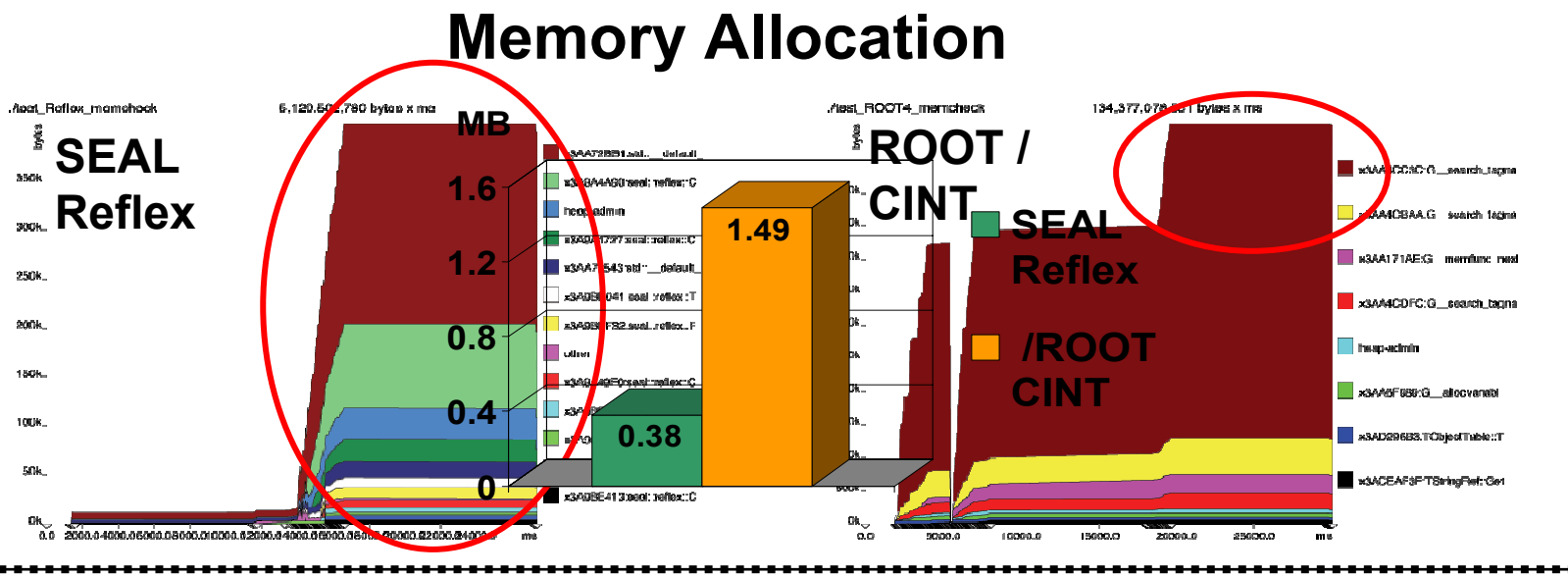
Review Report '03

- […] SEAL dictionary is the component that appears to have achieved the widest adoption and it is used in production by a wide community […]
- Remarks
  - Concern about sizes of dictionaries
  - Encourage convergence with ROOT/CINT

Dictionary Sizes

- Comparison with ROOT/CINT
  - Dictionary for 131 CLHEP classes





Convergence with ROOT/CINT

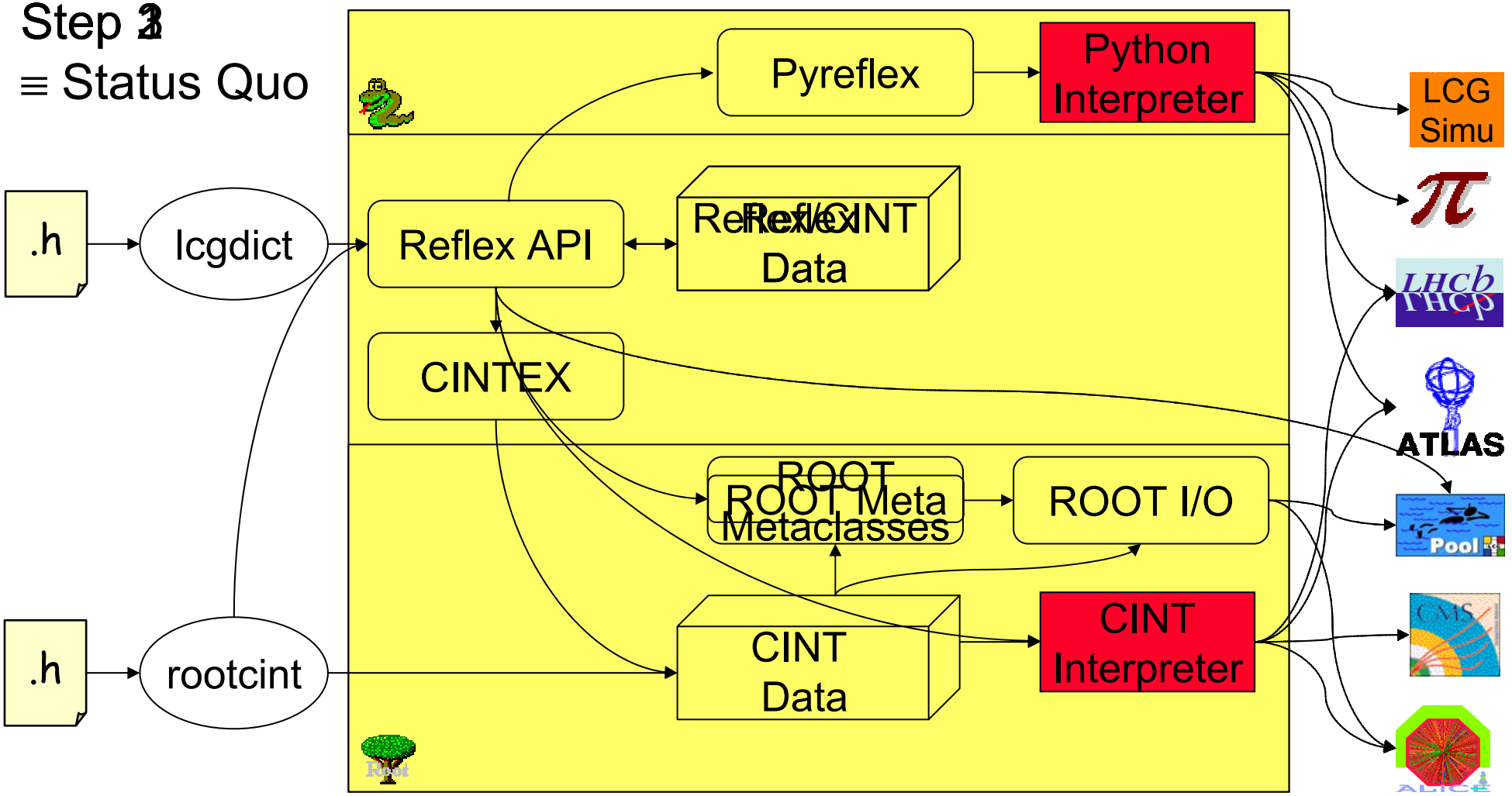
Proposed in 3 steps:

1. Fill CINT data structures (data and methods) from Reflex on demand.
2. Re-implement the ROOT metaclasses (TClass, TMethod, etc.) on top of Reflex
3. 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.

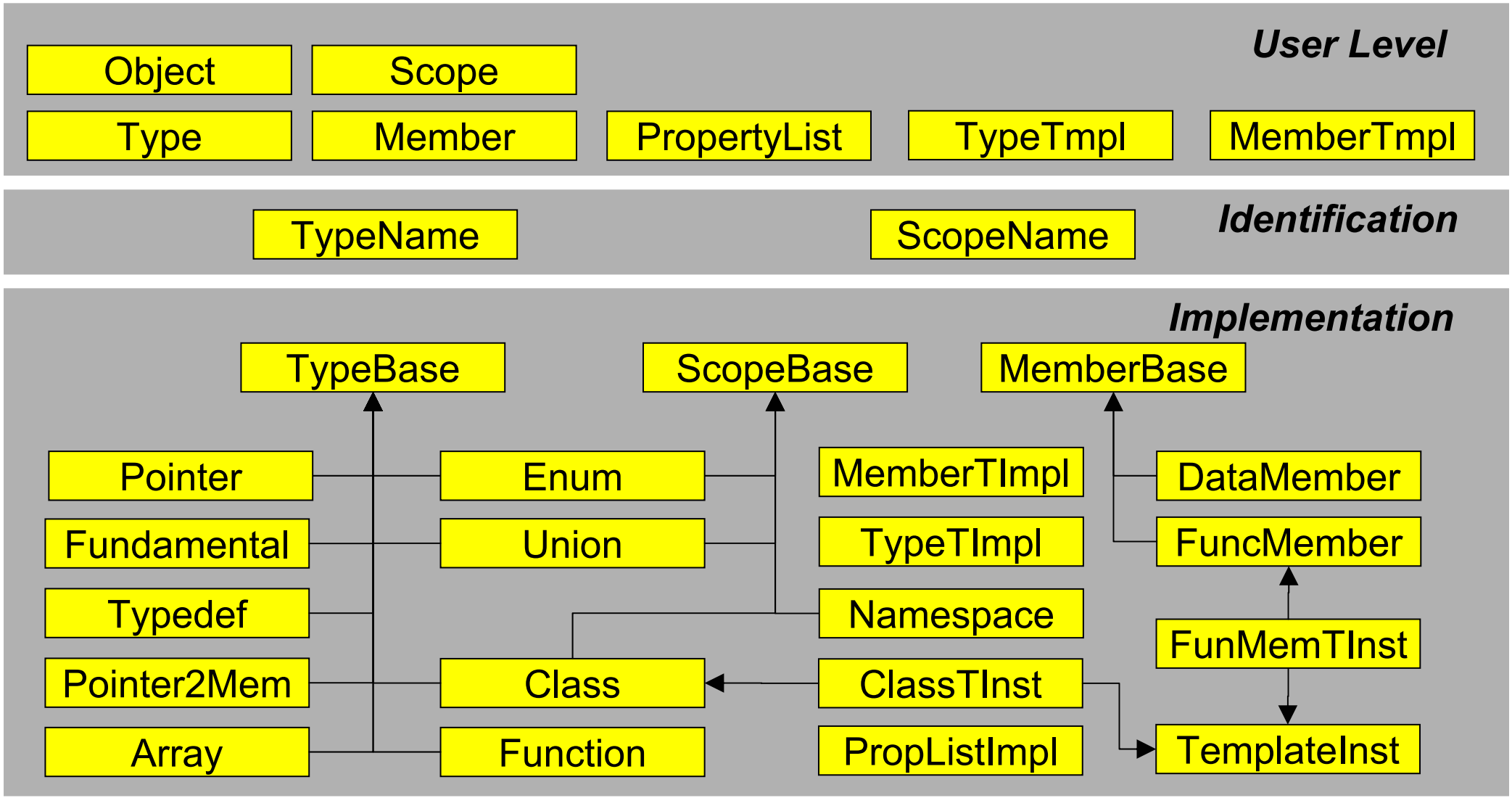
SEAL 1.6.0

Convergence (ctd.)

Step 3
≡ Status Quo



Class Diagram



Dictionary in SEAL 1.6.2

- Reflex
  - Ready, currently being integrated by others (e.g. POOL)
- Cintex
  - Released, ready for integration
- lcgdict
  - Adapted to generation of Reflex code
- Other Dictionaries
  - Two dictionaries Reflection/Reflex generated
    - » e.g. SealROOTDict, SealCLHEPDict

Scripting

Massimo Marino / LBNL

Major guidelines

- AA Review recommendations - 11/2003
  - Resolution on the results from the 2003 SEAL report on Binding Technologies
  - Continue with the PyLCGDict automatic binding approach and move to a Python implementation, aka, pylcgdict
  - Seek experiment feedback for the use of Python in interactive analysis
- SEAL Project Workplan addendum - 03/2004
  - Continue - and possibly expand - courses on Python programming
  - Complete pylcgdict
  - Remove external dependencies from PyROOT, make it part of ROOT distribution

Binding technologies

- Following AA Review 2003 recommendation
  - No work committed to further investigation and/or development based on
    - » Boost.Python
    - » SWIG
    - » interoperability issues among them
      - significant workload
      - no interest from or manifested by LHC experiment community
  - Pursued automatic binding approach initiated with PyLCGDict
  - Finished Python implementation - PyLCGDict2
  - Evolved to new Reflex API - Pyreflex

Python Bindings

- PyLCGDict, PyLCGDict2, Pyreflex
  - Provide access to C++ libraries in Python - transparently for the user
- What
  - Automatize generation of Python proxies to C++ objects
  - Map C++ constructs to Python natural ones
- How
  - Proxies are created based on information from an LCG dictionary
    - » based on Reflection (PyLCGDict, PyLCGDict2)
    - » based on Reflex (Pyreflex)
- Goal
  - Effortless creation of Python binding to C++ objects
  - Increase coverage of C++ constructs

Python bindings status

- All flavors of Python bindings are crucial components of and mostly used in:
  - ATLAS: GAUDI-Athena framework, distributed analysis (DIAL), interactive analysis (ESD/AOD)
  - LHCb: interactive GAUDI, physics analysis (Bender), event display (Panoramix)
  - PI: gluing various systems (AIDA, ROOT, HippoDraw, PyFML, etc)
  - Geant4: Geometry import/export (GDML+Python)

PyROOT

- CINT-based Python bindings developed for ROOT
  - Provides access to ROOT functionality in Python
  - Python from ROOT (Python commands in CINT)
    - » builtins and ROOT objects can cross interpreters
  - Part of ROOT distribution
- What
  - Access to all ROOT classes, as well as end-user CINT macro's
    - » Histogram, ntuples, files, graphics
  - Pythonization of ROOT classes, hooks for memory management.
- How
  - Proxies are created based on information from TClass and CINT dictionary
- Status
  - Performance has recently improved a lot
    - » benchmarks.C/benchmarks.py within 10% either way
    - » extreme worst case 3.5, but within 50% with psyco
  - Core has been refactored for easier maintenance

Python Courses

- Evangelize and teach the use of Python
- Format
  - 3 day course: Hands-on Introduction to Python Programming
  - (was!) Available through CERN Technical Training Programme
  - Excellent reviews
- How
  - Supporting material online: http://jacek.home.cern.ch/jacek/python-course/
- 11 courses given since last AA Internal Review

Documentation

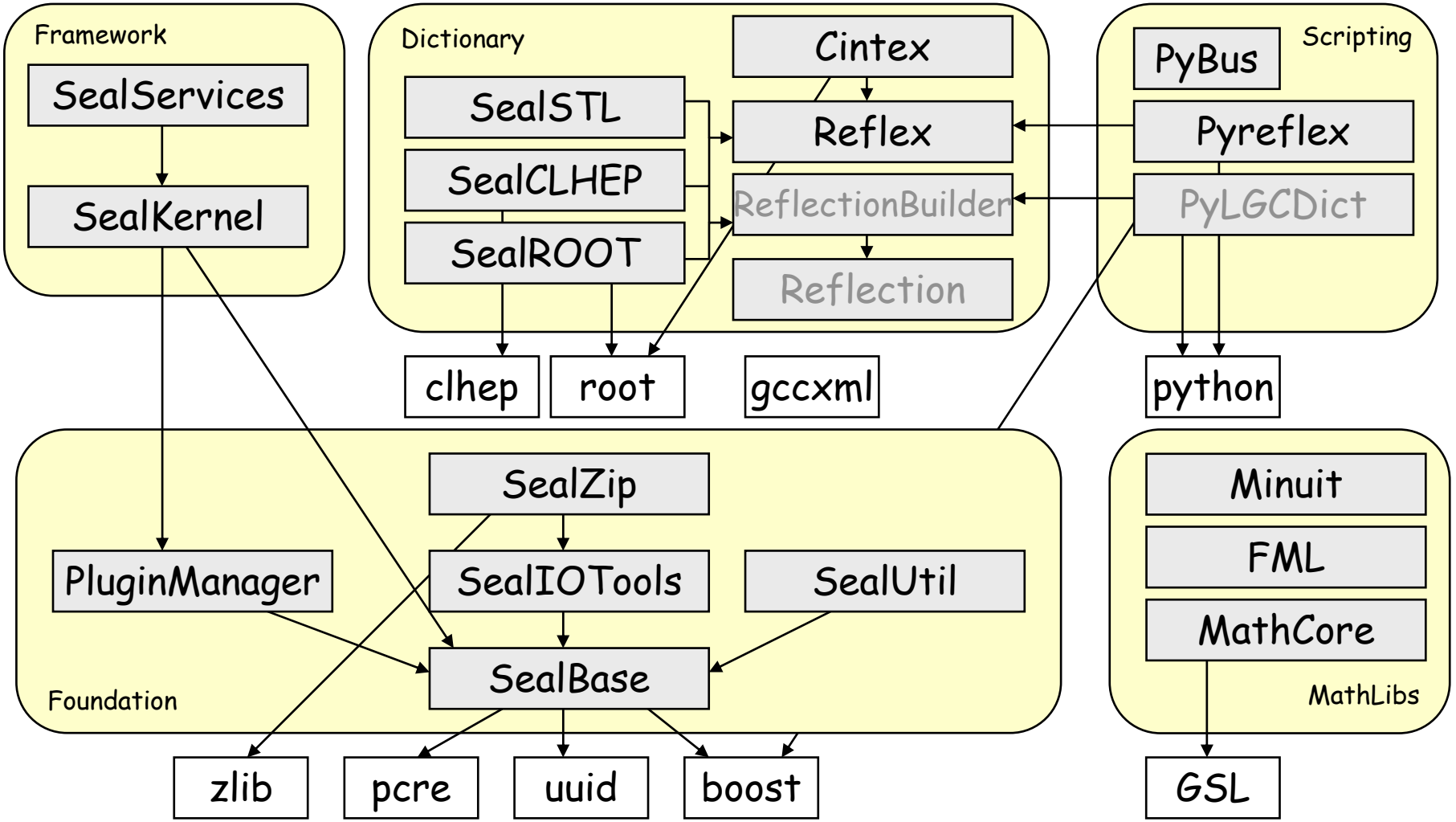
Documentation

- SEAL Workbook
  - Collection of web pages with practical How-To's for the various SEAL components
  - Available since June 2004
  - http://seal.web.cern.ch/seal/snapshot/workbook
- Reference Documentation
  - Doxygen and LXR generated

Software Process Issues

- SEAL is a collection of fairly independent packages but there are still difficulties to release the "parts" independently
  - The current version of build and configuration tools does not facilitate the task
  - Ongoing work in collaboration with SPI to solve the problem
  - Some packages (Minuit, MathCore, Reflex) has developed configure/make to build independently
- Fairly compete unit test suit
  - Most SEAL tests are unit tests (based on CppUnit and PyUnit)
  - Regularly run under the QmTest test driver on all supported platforms
- Rotating "Release Manager" role

SEAL packages and dependencies (1.6.0)



Release Road Map

Release	Date	Description (goals)
V 0.1.0	14/02/03	◆Establish dependency between POOL and SEAL ◆Dictionary generation from header files
V 0.2.0	31/03/03	◆Essential functionality sufficient for the other existing LCG projects (POOL) ◆Foundation library, system abstraction, etc. ◆Plugin management
V 1.0.0	18/07/03	◆Essential functionality sufficient to be adopted by experiments ◆Collection of basic framework services ◆Scripting support
V 1.1.0	05/09/03	◆Corrections and improvements of Framework
V 1.2.0	16/10/03	◆Support for ICC and VC++ compilers
V 1.3.0	25/11/03	◆Improvements in Plugin Manager ◆Consolidation Dictionary and Minuit
V 1.3.4	29/03/04	◆Bug fixes
V 1.4.0	23/06/04	◆Simpler plug-in manager, consolidation component model ◆New packages PyLCGdict2, FML, Workbook
V 1.4.2	02/11/04	◆Bug fixes ◆New package Reflex
V 1.6.0	23/02/05	◆New packages: Pyreflex, Cintex, MathCore

SEAL Products and their Usage

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

In use directly In use indirectly Planned use

Milestones

- 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	Done v=120	First version of the C++ mathlib package

- SEAL activities will be done within new SEAL+ROOT project
- No SEAL Milestones defined for 2005

Manpower Situation

- 2004 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%)

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
- 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)