SEAL Project Status and Plans

Shared Environment for Applications at LHC

LHCC Comprehensive Review of LCG-AA
24-25 November 2003
P. Mato / CERN

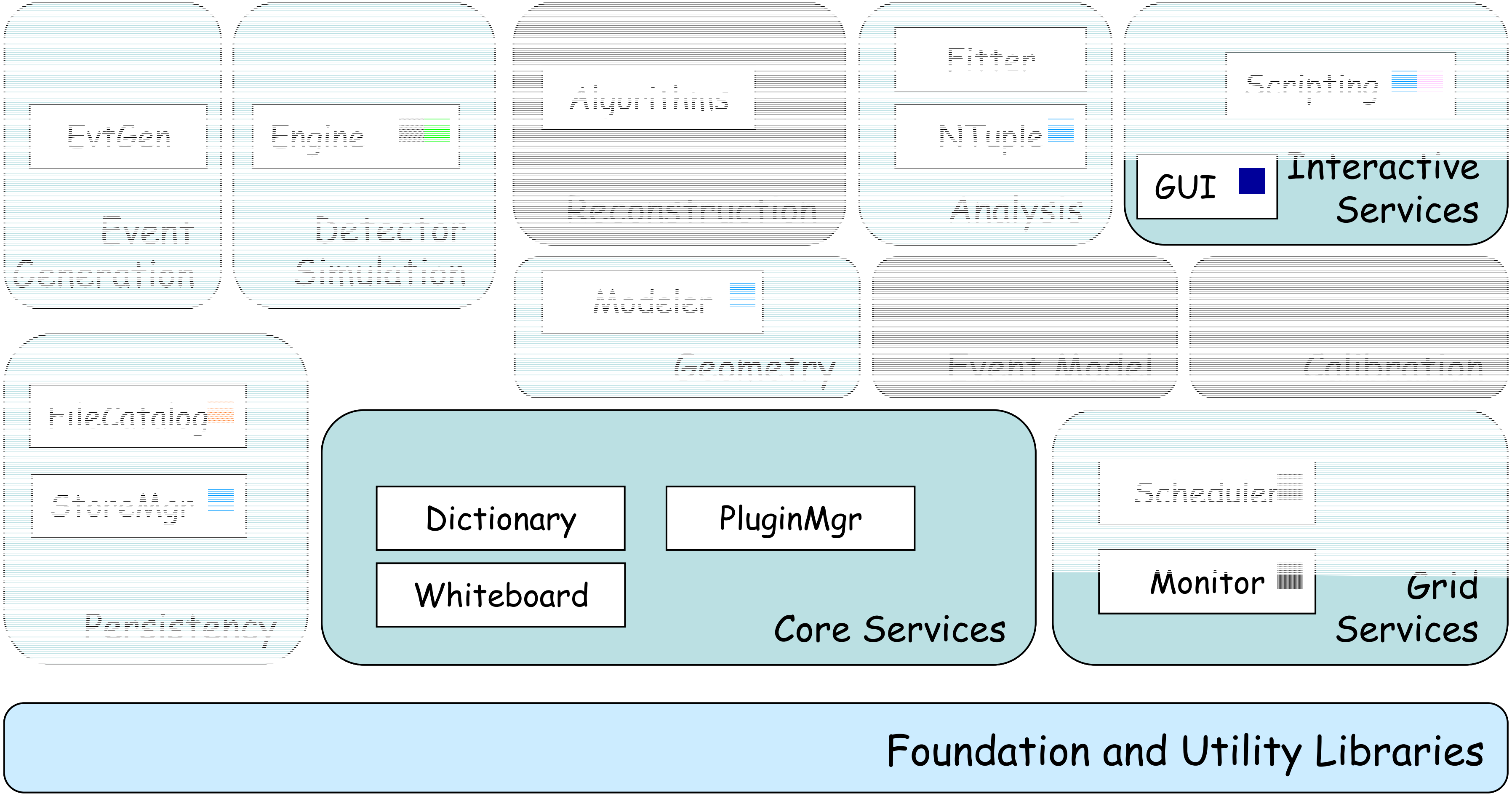
Contents

- Project Overview
- Work Packages
- Status and Milestones
- Summary

SEAL Overview

- SEAL aims to
  - 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
  - Basic Framework Services
    - Component model, reflection, plugin management, incident (event) management, distributed computing, grid services, scripting

Domain Coverage



Event Generation
EvtGen

Detector Simulation
Engine

Reconstruction
Algorithms

Analysis
Fitter
NTuple

Interactive Services
Scripting
GUI

Geometry
Modeler

Event Model

Calibration

Persistency
FileCatalog
StoreMgr

Core Services
Dictionary
PluginMgr
Whiteboard

Grid Services
Scheduler
Monitor

Foundation and Utility Libraries

🟦 ROOT 🟩 GEANT4 ▨ FLUKA 🟧 MySQL ▨ DataGrid 🟪 Python ■ Qt . . .

Assumptions, constraints, risks

- Do not re-invent the wheel
  - Most of the core software to be delivered by SEAL exists - more or less - in experiments' core software
    - » We will re-use as much as possible existing software
    - » Most of the work will be in re- packaging existing pieces of software
- If wheel squeaks…
  - Develop / adapt / generalize in order to achieve the necessary level of coherency and conformance to the architectural vision already established
- Adopt a Seal
  - In order to use SEAL, projects will need to replace their own software elements with SEAL functionally equivalent ones. This will certainly imply some period of instability for the experiment applications

Customers

- Other software LCG application area projects
  - Persistency (POOL)
  - Physicist Interface (PI)
  - Simulation (ROSE,…)
- LHC Experiment Frameworks and Applications
  - ATHENA/GAUDI (ATLAS)
  - COBRA (CMS)
  - GAUDI (LHCb)
- Other HEP projects
  - GEANT4 ?, …

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
Grid	Grid Services (not yet active)
Documentation	Education and Documentation

People

Foundation	**Lassi Tuura**, Lorenzo Moneta, Massimo Marino, Radovan Chytracek
MathLibs	**Fred James**, Matthias Winkler
Dictionary	**Stefan Roiser**, Christian Arnault, RD Schaffer, Zhen Xie, Pere Mato
Framework	**Radovan Chytracek**, Lassi Tuura, Pere Mato, Massimo Marino, Lorenzo Moneta
Scripting	**Jacek Generowicz**, Pere Mato, Wim Lavrijsen, Massimo Marino
Grid	
Documentation	Jacek Generowicz

~ 5 FTE

Foundation

- Inventory of existing libraries (http://seal.cern.ch/components.html)
  - Recommends classes by purpose
  - Grouping by most likely interest
- Main external library: Boost
  - Open source utility library (SEAL in contact with developers)
  - Portions being included in the next C++ standard library
- Auxiliary libraries: zlib, bz2lib, pcre (perl regexps), uuid (aka e2fsprogs), rx
- SealBase, SealUtil, SealIOTools, SealZip
  - Originated mainly from ClassLib (CMS)
- Plugin Manager
  - Basic concept: advanced object factory
  - Two simple interfaces: object instantiation, plug-in provider
  - Dynamic loading completely orthogonal — and optional!
- Next steps
  - Utility libraries development
    - » Hash maps (… others on demand)
  - Plugin Manager
    - » Work on the negative feed-back
    - » Development of utilities to diagnose problems
    - » Interfacing to dictionaries libraries
  - Education
    - » Teach how to use SEAL itself and Boost

MathLibs

- Support for GSL (Gnu Scientific Library)
  - Evaluation. How it compares with NagC.
  - Installation, validation, user consultancy, communication with GSL developers, extensions
- Re-implementation of MINUIT in C++
  - Prototype already available (Migrad and Minos). The numerical results of the two prototypes compared to the Fortran version. Compatible within the errors.
- Other studies
  - Comparison of various linear algebra packages
- Next Steps
  - Work plan in preparation (to be presented at SC2 meeting in December)
    - Ongoing discussions with Rene Brun to achieve a coherent program of work (LCG+ROOT)
  - Support for GSL
    - Recommendation to use GSL
    - Consultancy (contact with GSL developers)
  - Support for CLHEP
    - Active participation in maintenance. Consultancy
  - New Minuit
    - Evolve prototype to a finish product
    - Integration into analysis tools (ROOT, HippoDraw, …)

Dictionary

- Dictionary packages
  - *Reflection* (user API) and *ReflectionBuilder* (loading interface)
  - *DictionaryGenerator* for producing dictionary sources from C++ header files
    - Based on gcc_xml
- Standard Dictionaries
  - CLHEP: Random, Vector
  - STL: Vector, List, String
  - Dictionary: Reflection
- Dictionaries are being used
  - POOL (DataService, StorageService)
  - SEAL (PyLCGDict)
- Next Steps
  - Implementation of new reflection model (overcome some existing limitations)
  - Extending and creating dictionaries of popular packages on demand
  - Optimizations in size and speed
  - Common dictionary between CINT(ROOT) and LCG

Framework

- Component Model defined
  - Hierarchy of bases classes to support the component model
    - » A *Component* lives in a *Context,* forming a hierarchy.
    - » A *Service* provides its own local *Context*
  - 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)
- Next Steps
  - New Services
    - » Whiteboard service (object repository)
    - » Dictionary service (loading of dictionary libraries on-demand)
  - New implementations
    - » More Configuration service back-ends
    - » Corrections and re-designs are foreseen and possible
  - Integration in POOL and experiment frameworks (GAUDI/ATHENA)

Scripting

- Investigate ways in which Python bindings could be created
  - Make recommendations of best practice
  - Boost.Python and SWIG are the clear favourites
  - No convincing technical argument for choosing one over the other
    - AF selected to use Boost.Pyhton
  - PyLCGDict provides an alternative approach
- PyROOT
  - Provides access to ROOT functionality from Python
  - Uses ROOT/CINT dictionary with Boost.Python
  - Avoids binding individual ROOT classes
- PyLCGDict
  - Provides access to C++ libraries from Python
  - Uses LCG dictionary. Automatically generates Python proxies for C++ objects
  - Namespaces and Templates look natural in Python
- Next Steps
  - Python Bindings: Training and consultancy
  - PYLCGDict
    - Migrate much of functional core from C++ to Python exploiting Python's metaclasses.
    - Support more natural Python features (eg iterator protocol)
  - PyROOT: Undergoing performance improvements

Documentation

- Code Reference
  - Generated with Doxygen
- HowTo's
  - A set of HowTo's pages to teach specific aspects of SEAL
  - Being incorporated into the SEAL Workbook
- Release Notes
  - Detailed release notes for each release
- Design documents
  - Partial design documents exists in SEAL web
- 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

- Design
  - Team design sessions (sometimes very lengthy discussions)
- Python prototypes
  - To illustrate use cases and functionality
  - To test design choices
- Configuration and Build system
  - SCRAM is used to configure and build the software (CMT used to build the Win32 binaries)

Quality Assurance

- Code Review
  - No formal code reviews
  - Coding done very often in pairs (XP style)
  - More than one developer knowledgeable for each package
- Testing
  - Most of the SEAL tests are unit tests based on *CppUnit*
  - 217 tests driven by *QmTest* (small tunings still needed)
- Bug reporting and tracking
  - Savannah Portal
  - Internal SEAL "problems" also reported as bugs

SEAL Versions Road Map

Release	Plan	Date	Status	Description (goals)
V 0.1.0	14/02/03	14/02/03	internal	◆Establish dependency between POOL and SEAL ◆Dictionary generation from header files
V 0.2.0	31/03/03	04/04/03	public	◆Essential functionality sufficient for the other existing LCG projects (POOL) ◆Foundation library, system abstraction, etc. ◆Plugin management
V 0.3.0	16/05/03	23/05/03	internal	◆Improve functionality required by POOL ◆Basic framework base classes
V 1.0.0	30/06/03	18/07/03	public	◆Essential functionality sufficient to be adopted by experiments ◆Collection of basic framework services ◆Scripting support
V 1.1.0		03/09/03	public	◆Corrections and improvements of Framework
V 1.2.0		16/10/03	public	◆Support for ICC and VC++ compilers
V 1.3.0		24/11/03	public	◆Bug fixes

Milestones

2002/10/30	Done	Establish core libraries and services (SEAL) project
2002/11/30	Done	Define the V1 SEAL software suite
2002/12/1	Done v=17	Prototype object dictionary service released
2003/1/10	Done v=0	Present the initial SEAL work plan to SC2
2003/3/31	Done v=7	SEAL V1 essentials in alpha (V0.2)
2003/5/16	Done v=8	SEAL V0.3 internal release
2003/5/30	Done	Delivery of first round of GSL enhancements
2003/6/30	Done v=10	Nightly builds deployed in SEAL
2003/6/30	Done v=18	SEAL V1 release
2003/7/31	Late	Math library workplan in place
2003/8/30	Done v=44	SEAL icc test build support
2003/9/15	Late	SEAL ecc test build support
2003/9/15	Done v=24	SEAL support for Windows binaries
2003/9/30	Late	Statement on GSL and NAG usage for math library

Summary

- SEAL has delivered a number components that constitutes the basic foundation and utility libraries and object dictionary
  - The main "client" has been POOL
  - Currently being integrated into experiments' frameworks
- The first version of the Component Model and Framework services available
  - Must engage experiments to seek feedback before developing more services
- Scripting based on Python
  - Boost.Python and PyLCGDict recommended to provide Python bindings
  - Need to help POOL and experiments to provide Python bindings
  - Identifying early adopters to provide feedback
- The SEAL Workplan for 2004 is currently being defined
  - Including MathLibs
  - To be presented in the SC2 meeting in December