Status of SEAL

LCG Applications Area Meeting

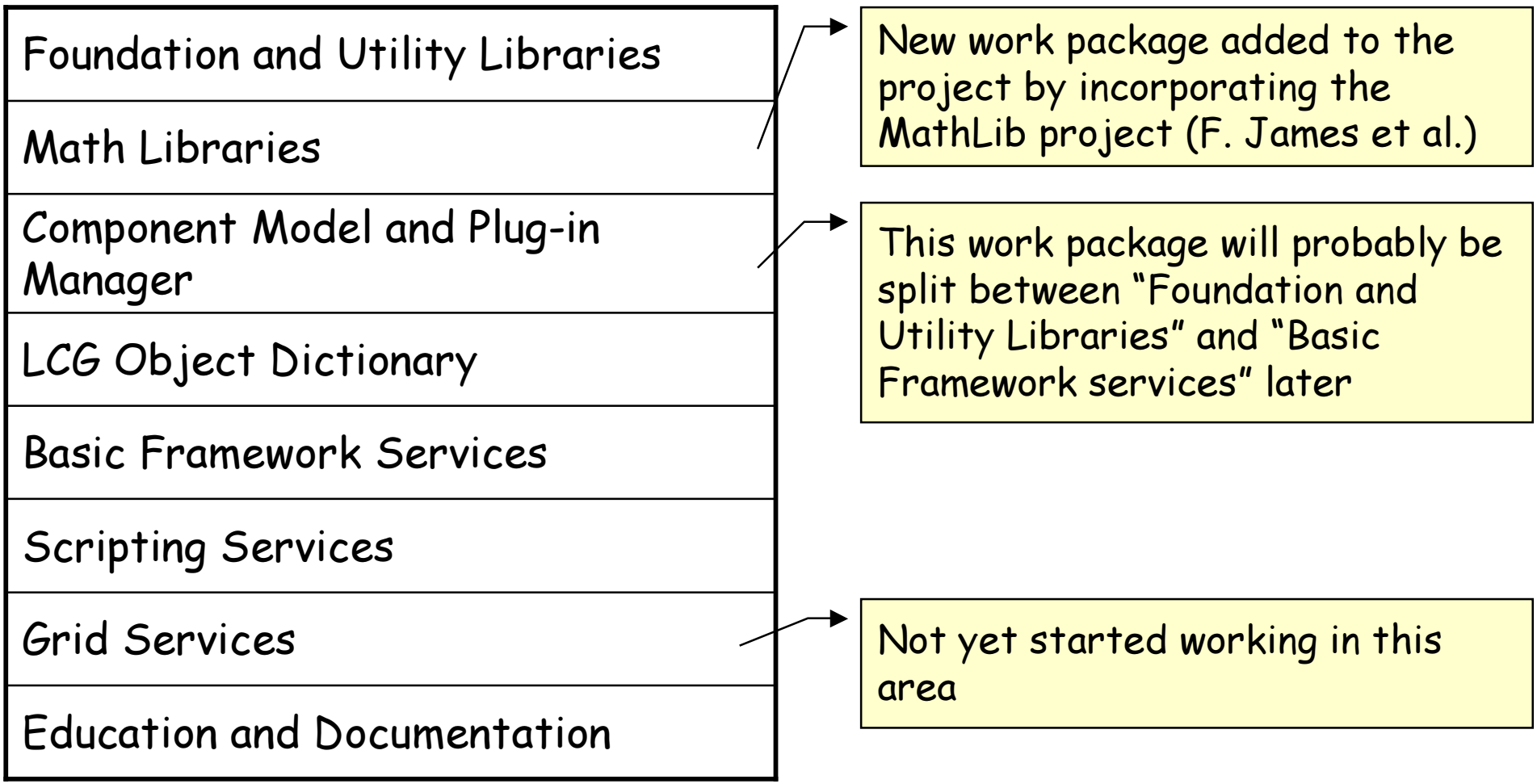
30 April 2003

P. Mato / CERN

Contents

- Overview
- Work Packages Status
- Summary

Project Work Packages



SEAL Versions Road Map

Release	Date	Status	Description (goals)
V 0.1.0	14/02/02	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 0.3.0	16/05/03	internal	◆Improve functionality required by POOL ◆Basic framework base classes
V 1.0.0	30/06/03	public	◆Essential functionality sufficient to be adopted by experiments ◆Collection of basic framework services ◆Scripting support

Released 14&26/02/03

Released 04/04/03

SEAL/POOL Releases Synchronization

SEAL			**POOL**		
this week	0.2.1	Technical release with new SCRAM			
			end-April	1.0.0	Functional complete LCG-1 set
mid-May	0.3.0	Functionality sufficient for POOL			
	0.3.X	Consolidation releases (bug fixes, etc)			
			mid-June	1.1.0	First performance & bug fix release. Consolidation.
end-June	1.0.0	Essential functionality sufficient to be adopted by experiments with tutorial and documentation			

Work Packages

1. Foundation and Utility libraries
2. Math Libraries
3. Component Model and Plug-in Manager
4. LCG Object Dictionary
5. Basic Framework Services
6. Scripting Services
7. Grid Services
8. Education and Documentation

1. Foundation and Utility Libraries

- Inventory of existing utility classes
  - Ongoing activity. Results in the SEAL Web.
- Support for Boost library
  - Installation, …
- Participation to CLHEP project
  - CLHEP workshop Jan 27-31 at CERN
  - Proposal from LCG/SEAL+SPI to help in the maintenance and infrastructure
  - LCG participation
    - » Savannah project portal (operational)
    - » CVS repository (ready to import FNAL work items)
    - » Working on template version of the Geometry classes and new version of Evaluator package

1. Foundation and Utility libraries (2)

- Imported *classlib* into Imports/classlib
  - A set of foundation and utility classes mainly used in Iguana (CMS)
  - Not yet fully incorporated as a proper SEAL library
    - » Include files redirection in V0.2.0
    - » Main classes should be done for V0.3.0
- Development of SealUtil and SealKernel packages
  - The idea is to develop SEAL utility and system library complementary to Boost and STL from existing code in classLib, Gaudi, HepUtilities, etc.
  - Prioritized by needs of POOL and other SEAL services
  - Classes already available: Exception, Status, Service, Timer, SimpleTime, FileName, SharedLibrary, Callback, ...
- Guidelines for selecting external libraries
  - Milestone delayed after June release

2. Math Libraries

- GSL evaluation by CAT, Indore
  - Project is now very advanced
  - Studied needed functionality by LHC experiments by looking what NAG routines are being used today
  - CAT team has made some implementations of functionality that could be found already in GSL
  - Next steps:
    - » Contact GSL team to incorporate new developed functionality to GSL and find out their current and future plans
    - » Automatic testing of GSL (with cppUnit) as part of every SEAL release
- Provide to experiments with math and statistics libraries to be used in analysis, reconstruction, simulation.
  - GSL support, …

2. Math Libraries (2)

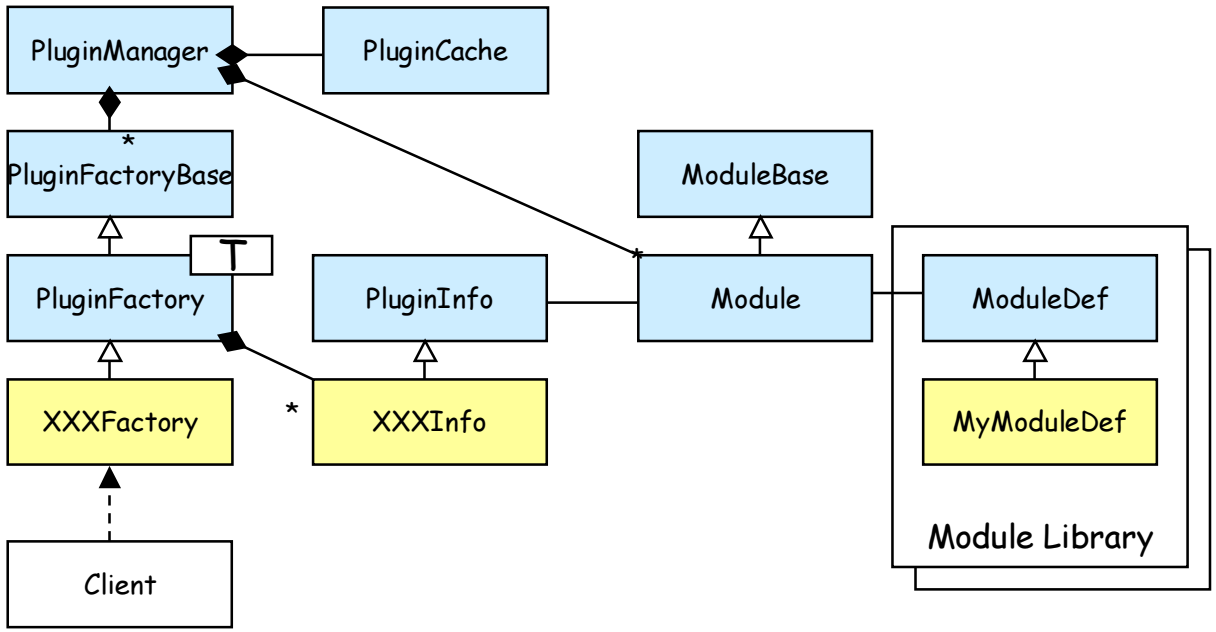
- Minuit *(see Matthias presentation 9th April)*
  - Prototype available in current SEAL release (0.2.0)
  - Migrad and Minos available (MathLib/Minuit)
    - » The numerical results of the two prototypes compared to the Fortran version. Compatible within the errors.
    - » No systematic testing done, neither for numerical accuracy nor performance.
    - » Aim of these prototypes is to get an overall view of the required functionality, as well as feedback on the C++ API.
- Other libraries
  - MathLibs/GSLAlgebra. Prototype wrapper around GSL. Not complete but restricted to Minuit needs.

3. Component Model and Plug-in Manager

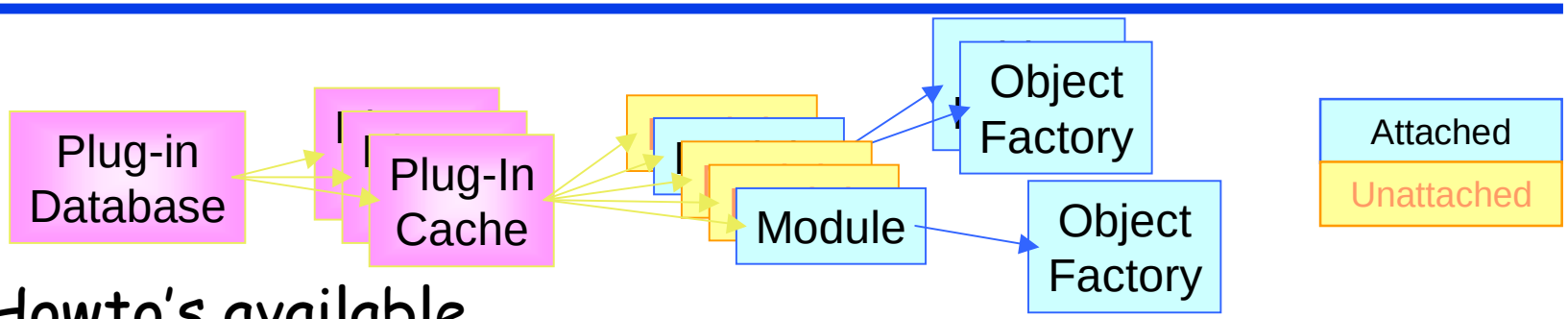
- Plugin Management
  - Service in charge of managing, querying, [un]loading plug-ins
- Implementation in two levels
  - Lower level (no framework constraints)
    - » Ability to create "module" libraries that contain "plugins"
    - » Ability to instantiate concrete implementations knowing the plugin category and the concrete type
    - » Caching information about what modules contain what plugins and their categories
  - Plugin instances management (assuming a given framework model)
    - » Locate other plugins and manage their lifetime
    - » Framework base classes to obey plugin/framework model

3. Component Model and Plug-in Manager (2)

- Available in current release (0.2.0)
  - Low level plugin management (Foundation/PluginManager)
  - Originated from CMS/Iguana



3. Component Model and Plug-in Manager(3)



- Howto's available
  - Host modules in your project
  - Create a new plugin factory
  - Create a new plugin module library

```
// Module Definition
DEFINE_SEAL_MODULE();
DEFINE_SEAL_PLUGIN(SomeFactory, MyObject);
```

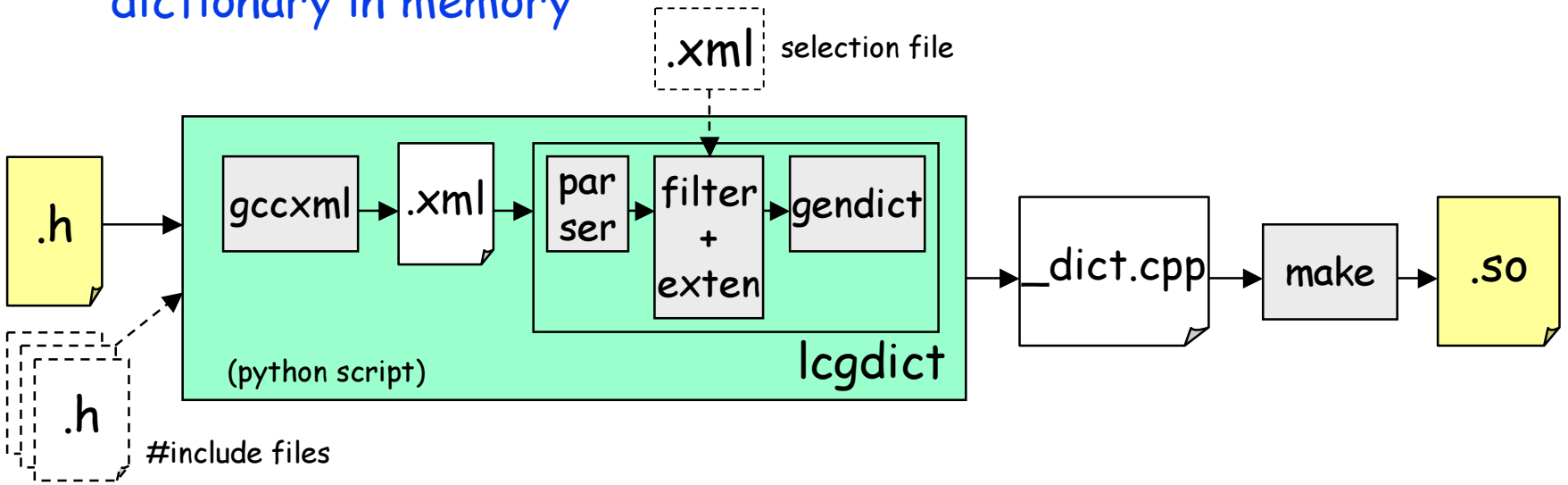
```
// Usage by a client
BaseObject* p;
p = SomeFactory::get()->create("MyObject");
```

4. Object Dictionary

- Reflection packages
  - The Dictionary/Reflection package provides the capability to introspect and interact with any C++ object at run-time
  - The Dictionary/ReflectionBuilder package is the "write" interface of the reflection information. Typically generated code uses this interface to built the reflection information at run-time
- Dictionary generation
  - Main goal: Full support of C++ without any class instrumentation
  - DictionaryGenerator package provides the command to do it
- Experiment examples
  - CMSExamples to test the CMS event model (PSimHit)
  - ATLAS is testing their event model (GuineaEvent)
  - No problem found so far

4. Object Dictionary (2)

- Dictionary generation from header files
  - Uses gccxml (0.4.0) to parse header-files (extension to gcc-compiler), generates intermediate XML file with dictionary information
  - XML file parsed by a python script which generates dictionary building C++ code (for "selected classes")
  - Compiled to shared library and loaded at run-time to create dictionary in memory



4. Object Dictionary (3)

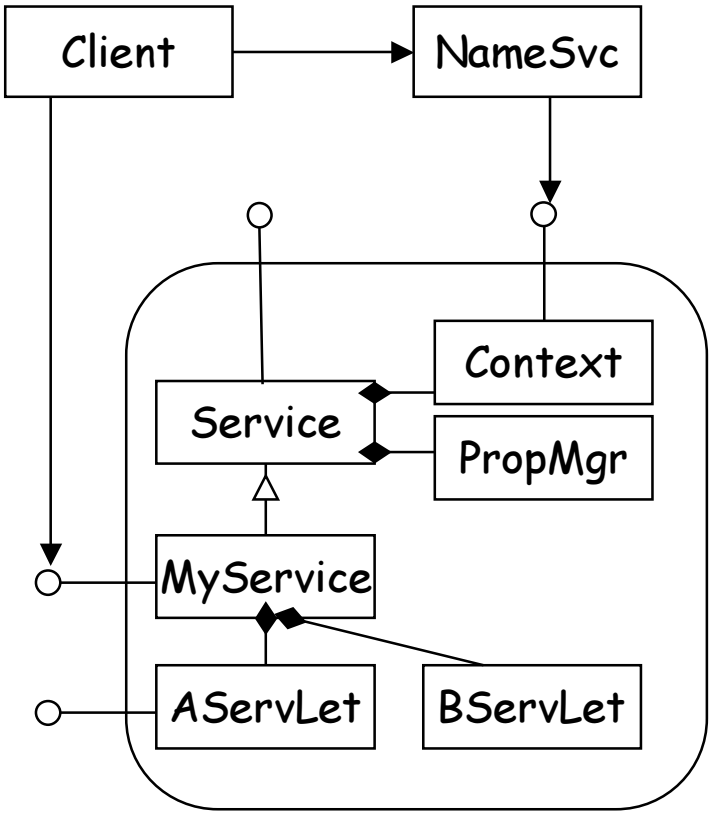
- Object Dictionary being used by POOL
  - RootStorageSvc from POOL uses the LCG Object Dictionary to populate ROOT/CINT dictionary
  - Examples to write/read complex C++ object models involving many levels of STL containers exist using the DictionaryGenerator from header files
- Common dictionaries
  - Started to produce the dictionary for CLHEP
    - » Useful to share between experiments events models

5. Basic Framework Services

- Developing the basic framework
  - Component model definition
    - » identification, lifetime strategy, interfaces, etc.
  - Started to review existing designs
- Developing basic services
  - Started design and implementation for the message reporting service
  - Other services like component configuration, "event" management, object white board, etc. will continue soon after
- Nothing released yet
  - Expected rapid progress
  - Basic framework and few basic services by release 0.3.0 (mid-May)

5. Basic Framework Services (2)

- The current ideas
  - Give to the "users" (framework developers) a number pre-defined service base classes (from simple to more complex)
  - Developers should "model" their services using one of the proposed base classes
  - Standard functionality (instantiation, configuration, lifetime management, etc.) should be provided by the bases classes



6. Scripting Services

- Define guidelines for developing Python bindings
  - Evaluate existing options: SWIG, Boost.Python, SIP, and raw Python C-API
  - Study how each technology handles a number of predefined cases
    - » Examples: Method overloading, inheritance across boundaries, templated classes and methods, natural mapping of Python constructs, etc.
  - Study interoperability between binding technologies
  - Produce an evaluation report. Expected by mid-May
- Failed to develop GSL bindings using raw Python C-API
  - Selected as an example for automating the generation of Python bindings
  - To achieve a "natural" Python binding requires manual intervention

6. Scripting Services (2)

- PyROOT: Python bindings for ROOT (former RootPython)
  - PyROOT is a Python extension module that allows the user to interact with any ROOT class
    - » Done generically using the ROOT dictionary. No need to generate any Python wrapper code to include new ROOT classes
  - Upgrading to version 2 of Boost.Python required a complete re-write (~ 6 classes)
  - Many added features with respect previous version
  - Released in SEAL 0.2.0
- Started to work on the LCG Dictionary bindings
  - Basically repeating the work of PyROOT using the LCG dictionary instead

8. Education/Documentation

- Main activities and tasks
  - Produce documentation
  - Produce training material (tutorials)
  - Help incorporating SEAL components into LCG projects and experiment frameworks
- Existing documentation
  - Very limited for the time being
  - Produced a number of HowTo pages for the released elements
    - » Plugin Manager, Dictionary generation, PyROOT, etc.
- Helping POOL to incorporate released components
  - We have a team member working for both POOL and SEAL
- Will be preparing tutorials for June release
  - Time of possible adoption by experiments

Software Process

- SPI provided infrastructure
  - Savannah Web portal, CVS repository, development tools (SCRAM, Doxygen, CppUnit, etc.) and services (external tool repository, etc.)
  - Slowly getting the good dynamics and practices
- Efforts to "standardize" tool usage and conventions
  - Avoiding divergences between LCG projects
  - Adaptation of tools to "LCG proper" working models
- Release procedures
  - Developing SEAL proper procedures
  - Tagging, building, testing, releasing, documentation generation, announcing, etc.
  - Automation of the process is essential (rotating release manager role)
- Platforms
  - Single release platform for the moment: Linux RedHat 7.3/gcc-3.2

Summary

- SEAL has had three releases: 0.1.0, 0.1.1, 0.2.0 since the beginning of the year at the scheduled times
  - The main emphasis has been to support POOL (Dictionary, Plugin Management, Foundation classes, etc.)
- The functionality available is not yet enormous but its development has help us
  - To get experience on new tools and procedures
  - Build the development team
- No technical problem encountered so far
- Next scheduled releases
  - Technical release using new SCRAM version these days
  - 0.3.0 by mid-May, 1.0.0 by end-June