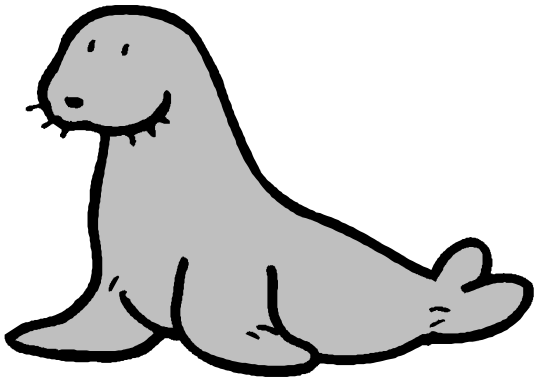
SEAL V1 Status

12 February 2003

P. Mato / CERN



Shared Environment for Applications at LHC

Release Roadmap

Release	Date	Status	Description (goals)
V 0.1	14/02/03	internal	◆Establish dependency between POOL and SEAL ◆Dictionary generation from header files
V 0.2	31/03/03	public	◆Essential functionality sufficient for the other existing LCG projects (POOL) ◆Foundation library, system abstraction, etc. ◆Plugin management
V 0.3	16/05/03	internal	
V 1.0	30/06/03	public	◆Essential functionality sufficient to be adopted by experiments ◆Collection of basic framework services ◆Scripting support

public = complete, documentation, etc.

Work Packages

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

Foundation and Utility Libraries

- Goal
  - Develop SEAL utility and system library complementary to Boost and STL from existing code in *classLib*, Gaudi, HepUtilities, etc.
- Achieved so far
  - Boost installation
  - Inventory of existing utility classes
  - *classlib* in SEAL repository
  - Few classes required by POOL moved from POOL to SEAL
- Packages
  - Foundation/SealKernel (available in pre-release V 0.1)

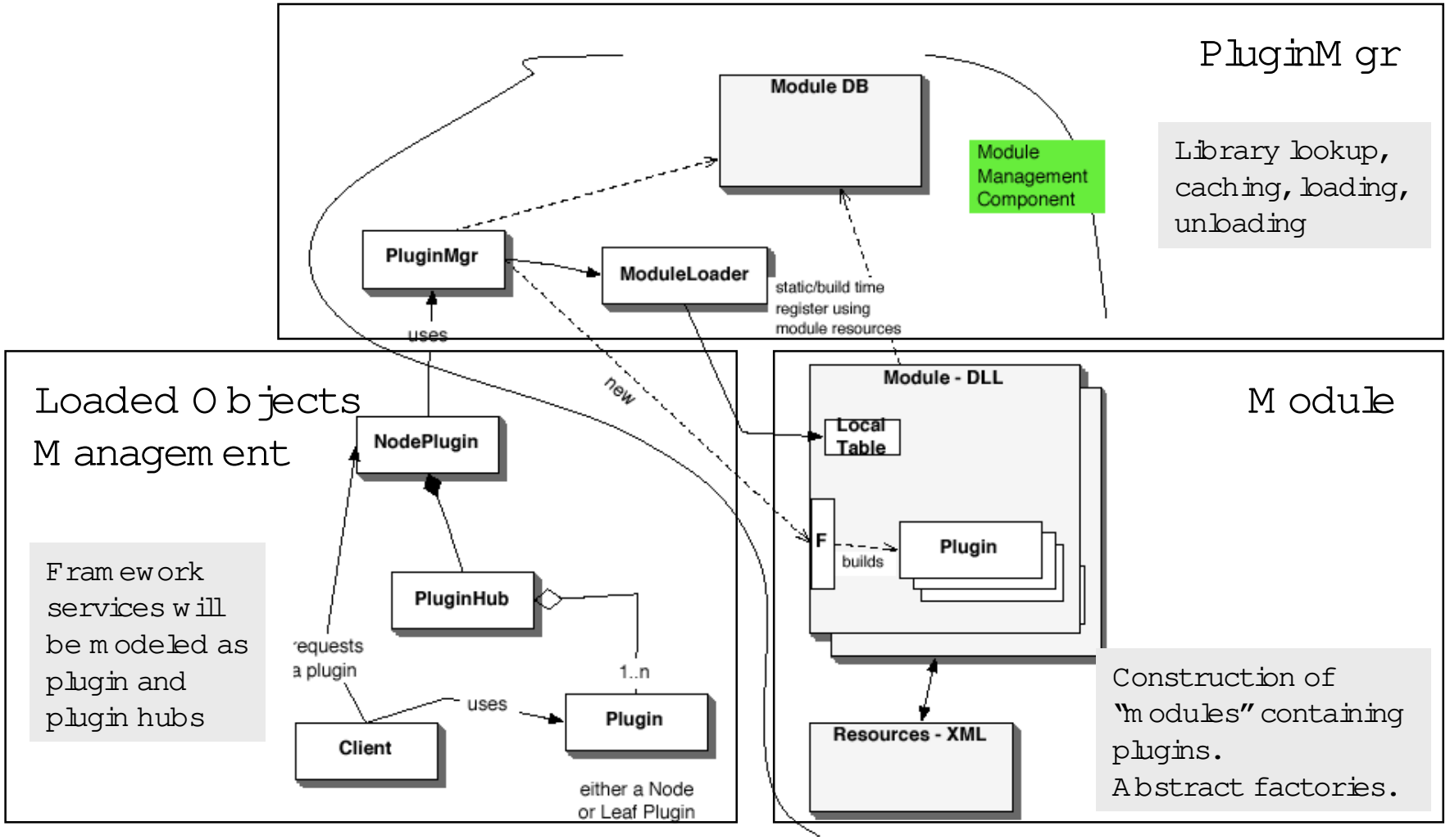
Math Libraries

- New SEAL work package
  - The Math Libraries project (F. James et al.) is becoming a work package of the SEAL project
  - Started to define the work plan
- Goals
  - Provide to experiments with math and statistics libraries to be used in analysis, reconstruction, simulation.
- Current activities
  - Evaluation of GSL (India). Finished by 15 April 2003
  - Implementation of Minuit in C++, ...
  - CLHEP participation and support
  - GSL support

Component Model and Plug-in Manager

- Goals
  - Define component and interface model following the blueprint report guidance. Develop plug-in Manager for lookup, loading plugins.
- Current activities
  - Design phase. Definition of concepts.
  - Ideas from Iguana and Gaudi.
- Deliverables
  - Prototype should be ready by first public release (end March)

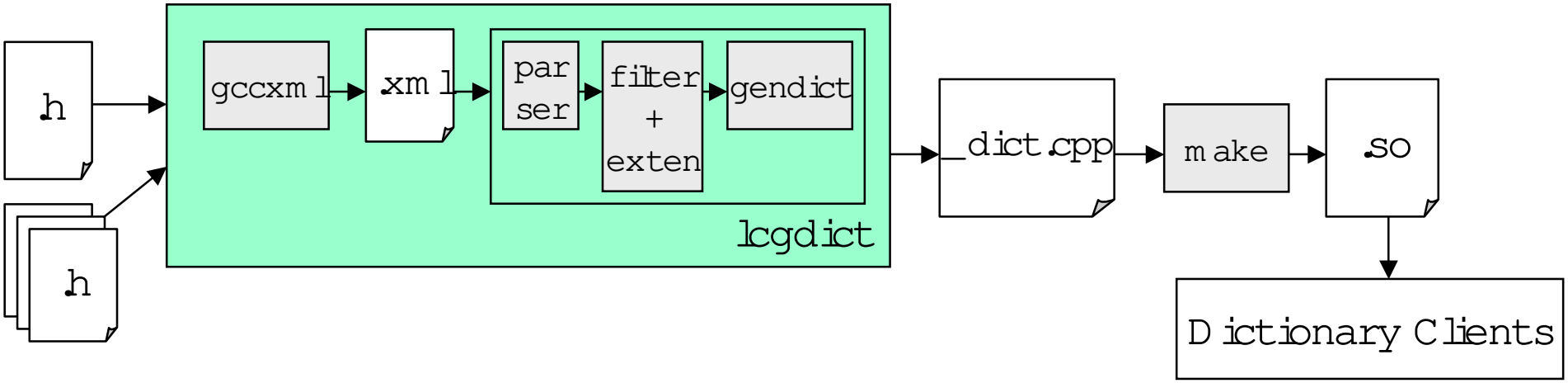
Plugin Management (current ideas)



LCG Object Dictionary

- Areas of work
  - Reflection Packages (Reflection, ReflectionBuilder)
  - Dictionary generation from header files using gcc_xml technology (DictionaryGenerator)
  - Python-binding to Reflection
  - Gateway CINT -> LCG Dictionary
  - Generation of .h-files and dictionary from some higher-level-language (e.g. XML)
- Concentrating in Reflection packages and dictionary generation from header files
  - Main goal: Full support of C++, without any class instrumentation
  - Parsing XML file from gccxml and producing "usable" dictionaries
- Packages available in pre-release V 0.1
  - Reflection, ReflectionBuilder, DictionaryGenerator, DictionaryExample

Dictionary Generation




```
>lcgdict LorentzVector.h -I{...}\CLHEP\1.8.0.0\include
Parsing file with GCC_XML OK
Generating LCG Dictionary
Generating class HepLorentzVector
Generating class Tcomponent
```

Filtering and Extending

- For any simple header file, gccxml generates the information for hundreds of classes
- Started filtering with a naïve and simple algorithm
  - The dictionary is generated for all classes defined in the input file and templated classes using these classes.
  - Explicit template instantiations will be required
- Extensions to classes are also required by some clients (e.g. POOL)
  - Default constructors, special methods to handle collections, class ID, etc.
  - Hanging based on conventions and strategies for the time being
  - Later these extensions could be provided by the client itself applying the same conventions and strategies

Dictionary ToDo

- The Dictionary supports already quite a lot of C++
  - No problem with things like vector<pair<list<A>,pair<B,C>>>
- Things to be improved in the dictionary or in the generation
  - Handling function types
  - Handling typedef
  - Proper handling of attributes (public, private, const, etc.)
- Integration in the build system (SCRAM)
  - Minimal integration will be done by V 0.1 pre-release
- Not unsolvable known problems
  - The "guinea-events" from ATLAS and CMS will be used as examples

Scripting Services

- Goals
  - Define guidelines for developing Python bindings
  - Python bindings for standard services and utility libraries developed in SEAL
  - Upgrade Python bindings for ROOT (PyROOT)
- Current activities
  - Evaluate existing options: SWIG, Boost.Python, SIP, ..., raw Python API
  - Started with PyROOT (C++) and PyGSL (C) as examples in the evaluation process. They are candidates to become "real" products.

Pre-release V 0.1 Plans

- Platform and Externals
  - Linux gcc-3.2
  - Current externalpackages: Python 2.2.2, gccxml 0.4.0
  - Soon: boost 1.29, GSL 1.3
- The repository will be tagged Friday 14th
  - Ready to be used by POOL in V0.4

Summary

- The goals of this first pre-release V 0.1 are:
  - Exercise the dependency and integration between POOL and SEAL
  - LCG Dictionary generation (at prototype level) to produce "usable" persistency applications
  - Good test for the development process and infrastructure for SEAL
- Do not expect much more in this release
- Looking forward to have more functionality by end of March (V 0.2)