Seal Foundation

Lassi A. Tuura, Northeastern University, CMS
Applications Area Internal Review
20 October 2003

SEAL Foundation Overview

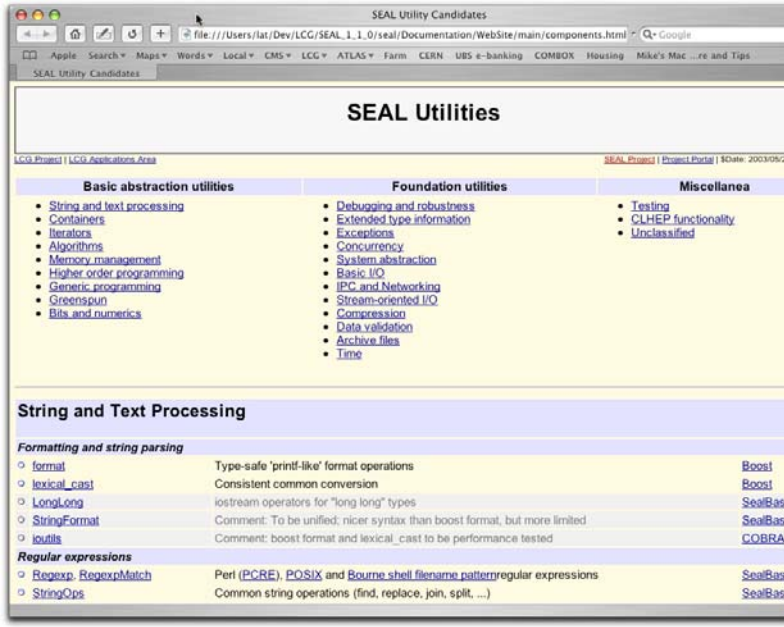
- Low-level, fairly independent class libraries complementary to existing high-quality standard and de-facto standard libraries
  - Broad range of unrelated functionality which makes sense to re-use across the LCG projects and the experiments
  - Pure libraries, no high-level application or framework behaviour
  - Leverage existing high-quality libraries but avoid excessive dependencies
  - Developed or adapted as needs arise
- Basic plug-in manager
- Largely originated from CMS

Foundation Components

- Basic foundation and system abstraction
  - Boost, SealPlatform, SealBase
- Stream-oriented I/O, compression and decompression, digest and archive utilities
  - SealIOTools, SealZip
- Plug-in manager and tools
  - PluginManager, PluginDumper, PluginChecker

Basic Utility and Abstraction Layer

- Inventory of existing libraries
  - Recommends classes by purpose
  - Grouping by most likely interest
  - http://seal.cern.ch/components.html
- Main external library: **Boost**
  - Open source utility library
  - SEAL in contact with developers
  - Portions being included in the next C+
  - Supported installation and working on release tests
- Auxiliary libraries: **zlib**, **bz2lib**, **pcre** (perl regexps), **uuid** (aka e2fsprogs), **rx** (posix regexps; only on windows)



Basic Utility And Abstraction Layer…

- Focus on high-quality, efficient, well tested, portable library that allows developers to program meaningful tasks in easy and understandable ways
  - Encode best practises and techniques
  - Not just a least common denominator
  - Example: creating an anonymous memory map region

```
FileMapping null(SystemInfo::pagesize() * 10);
IOBuffer   sect = nullmap(FileMapping::ProtRead
                        | FileMapping::ProtWrite,
                                    FileMapping::MapPrivate);
char       *data = static_cast<char *>(sect.data());

for(IOSize i = 0; i < sect.size(); ++i)
   data[i] = i % 10;
```

Basic Utility And Abstraction Layer…

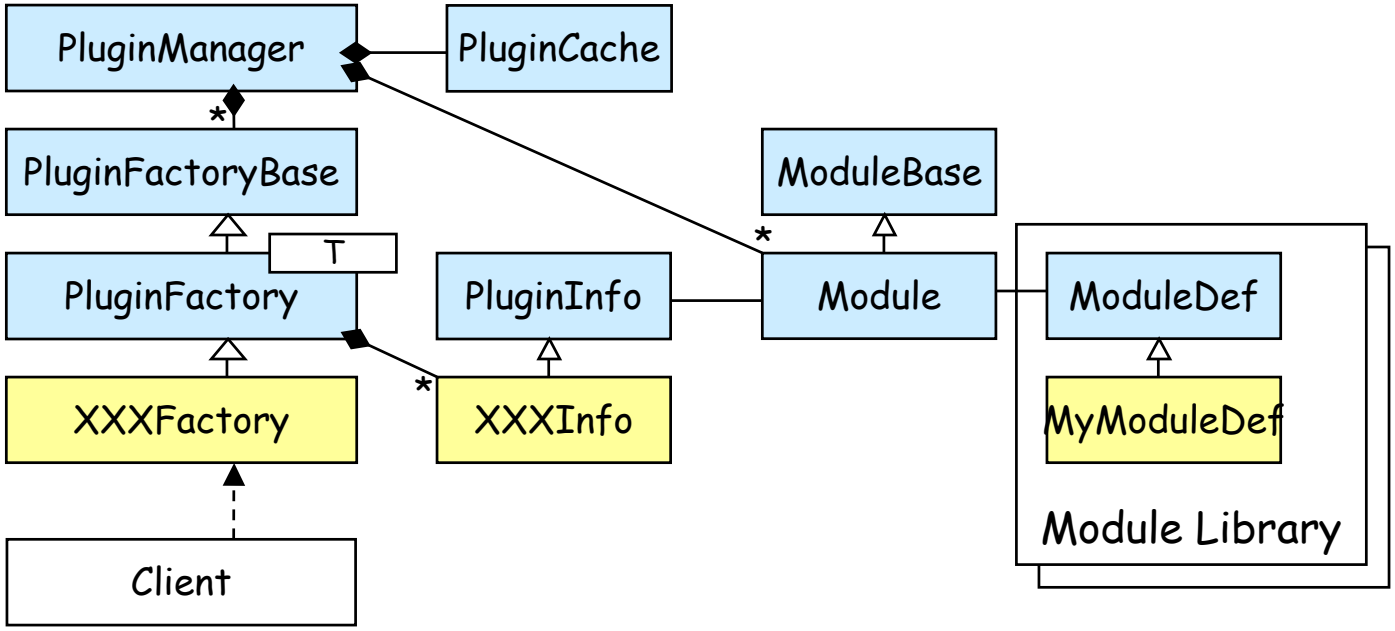
- Aiming at high quality in…
  - **Class selection**: recommending best practice to LHC developers
  - **Documentation**: all classes have documentation, some of them extensively so; also the "jump table" by purpose
  - **Testing**: ~200 test programs, at least one for every class, some classes have even more than one test per method (corner cases)
- But…
  - Why have a portability layer when project depends on ROOT?
  - Many good libraries to consider — sensitivity to dependencies
  - Automated test infrastructure crucial, only recently available
  - Windows port stellarly poorly tested due to lack of development machines — I have no Windows box, will try terminal server

Basic Utility And Abstraction Layer…

- Status of the base utilities in Foundation
  - Following up on standard library and Boost developments
  - Little development currently planned in SEAL (hash maps?)
  - Waiting for feedback and future guidance from real use experience in the LHC experiments and LCG projects
  - Feedback so far mixed
    - » Shouts of joy from a small number of individuals
    - » Apparently little use outside CMS — even within SEAL
  - Few bugs found so far
    - » Mostly by myself before anyone else noticed
    - » Probably a reflection of limited usage, not of testing quality

Plug-in Manager

- Low level plug-in management
- Originated from CMS/IGUANA



Plug-in Manager…

- Basic concept: advanced object factory
  - Two simple interfaces: object instantiation, plug-in provider
- Dynamic loading completely orthogonal — and optional!
  - Self-discovering regarding to module libraries
  - Dynamic loading can be used where desirable and practical
  - Class packaging can be changed according to other needs
  - Physical packaging changes require no code or configuration changes

Plug-in Manager…

- Lot of positive feedback, seems to be liked a lot
  - People who like good modular design seem to like it a lot
  - POOL "jumped on it" the moment it was released
- Lot of negative feedback as well
  - Lot of code out in our community does not conform to language standards or platform ABI and breaks when used as a plug-in
  - Lot of broken configurations (e.g. CERN Linux GCC 3.2.x builds)
  - Much not-so-modular code that causes problems
  - Problems are frequently complex and difficult to debug
  - Module validation tools help, but still facing problems
  - Not sure if this technology is mature — nor us mature for it

Plug-in Manager…

- Bugs: a few reported, urgent ones fixed, small ones remain
- Status
  - Used by POOL, PI
  - Used by CMS: IGUANA; considering in COBRA, probably will prototype first to see how well it will work in CMS-wide scale
  - Tried in ATLAS, don't know if it ever succeeded
    - » Problems with ABI violations
    - » Suggested two or three alternatives
    - » Conversation seems to have died, didn't push, no bugs reported
  - Don't know about LHCb
- Future plans
  - A few small things to implement, a day or two of work

Summary

- A substantial amount of work has been done
  - A good team effort: Lorenzo, Massimo, Jacek and me
  - Kudos to Lorenzo for timing services and major work on testing
- Waiting for feedback on experiment integration
  - A lot of tools and services waiting for deployment
  - Gladly welcoming feedback on documentation and usability
  - SEAL team would be enthusiastic to help experiments use the classes or provide education (Boost or SEAL itself)
- Doubts
  - Plug-in manager seems to generate many problems
  - Project dependency structure puts a lot into question