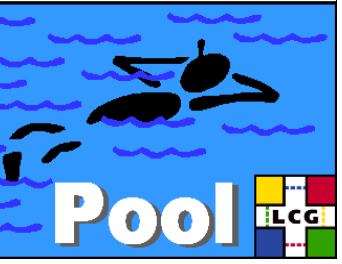


POOL Project Review
Introduction & Overview

Dirk Düllmann,
IT-DB & LCG-POOL

LCG Application Area Internal Review

20-22 October 2003

What is POOL?

- Project Goal: Develop a common Persistency Framework for physics applications at the LHC
  - Pool Of persistent Objects for LHC
- Part of the LHC Computing Grid (LCG)
  - One of the first Application Area Projects
- Common effort between the LHC experiments and the CERN IT-DB group
  - for defining its scope and architecture
  - for the development of its components

POOL Objectives

- To allow the multi-PB of experiment data and associated meta data to be stored in a distributed and Grid enabled fashion
  - various types of data of different volumes (event data, physics and detector simulation, detector data and bookkeeping data)
- Hybrid technology approach, combining
  - C++ object streaming technology, such as Root I/O, for the bulk data
  - transactional safe Relational Database (RDBMS) services, such as MySQL, for catalogs, collections and meta data
- In particular, it provides
  - Persistency for C++ transient objects
  - Transparent navigation from one object across file and technology boundaries
    - Integrated with a external File Catalog to keep track of the file physical location, allowing files to be moved or replicated

POOL Timeline and Statistics

- POOL project started April 2002
  - Ramping up from 1.6 to ~10 FTE
  - Persistency Workshop in June 2002
  - First internal release POOL V0.1 in October 2002

- In one year of active development since then
  - 10 public releases
    - POOL V1.3.1 last Friday
  - Some 50 internal releases
    - Often picked up by experiments to confirm fixes/new functionality
    - Very useful to insure releases meet experiment expectations beforehand
  - Handled some 150 bug reports
    - Savannah web portal proven helpful

- POOL followed from the beginning a rather aggressive schedule to meet the first production needs of the experiments.

POOL Milestones

- First "Public" Release - V0.3 December '02
  - Navigation between files supported, catalog components integrated
  - LCG Dictionary moved to SEAL – and picked up from there
    - Basic dictionary integration for elementary types
- First "Functionally Complete" Release - V1.0 June '03
  - LCG dictionary integration for most requested language features including STL containers
  - Consistent meta data support for file catalog and event collections (aka tag collections)
  - Integration with EDG-RLS pre-production service (rlstest.cern.ch)
- First "Production Release" - V1.1 July `03
  - Added bare C++ pointer support, transient data members, update of streaming layer data, simplified (user) transaction model
  - Due to the large number of requests from integration activities still rather a functionality release than the planned consolidation release.
  - EDG-RLS production service (one catalog server per experiment)
- Project stayed close to release data estimates
  - Maximum variance 2 weeks
  - Usually release within a few days around the predicted target date

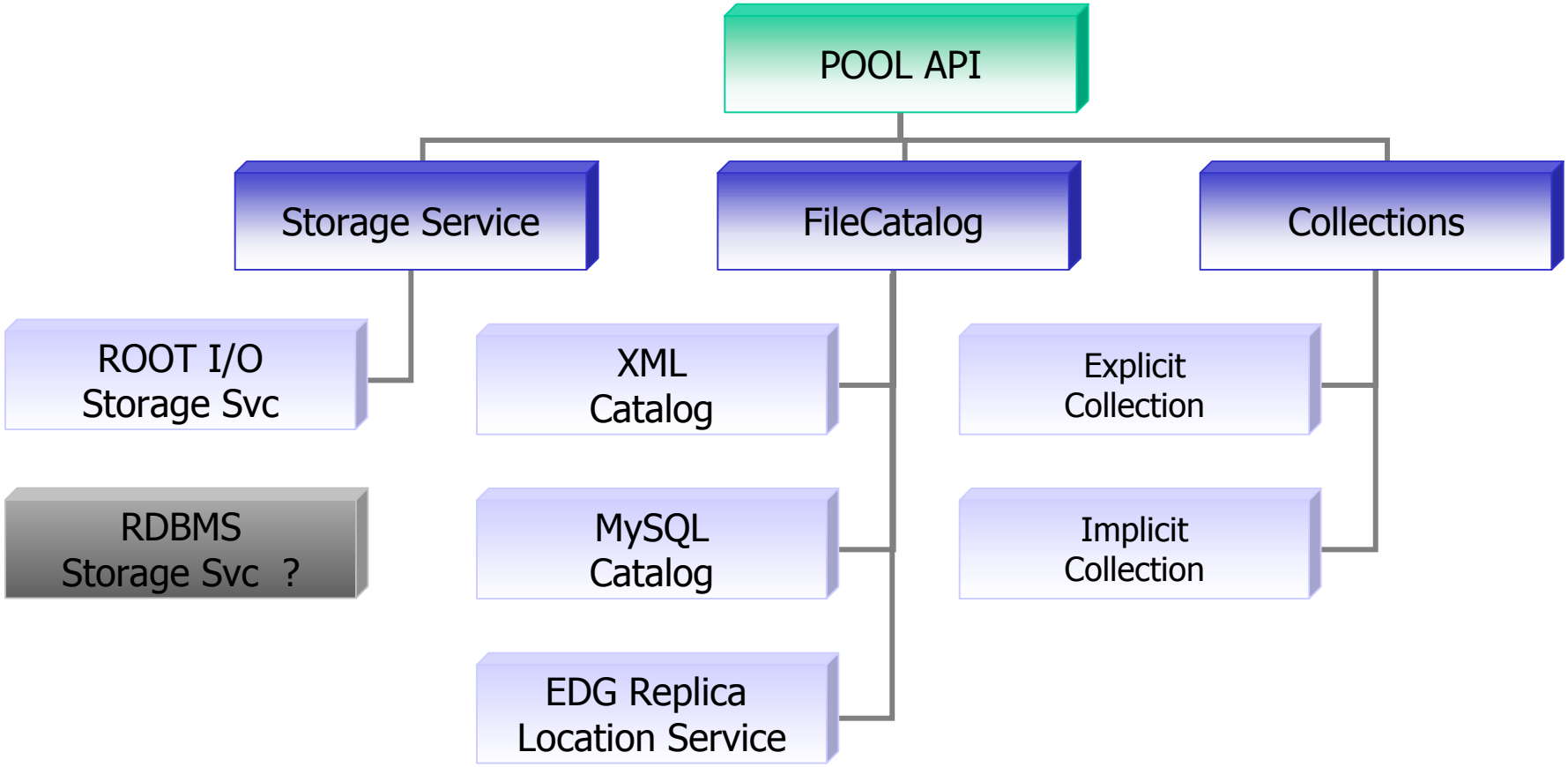
POOL - Known Issues

- No adequate end-user documentation yet
  - Preparing a user guide collecting the experience gained during the framework integration
  - Expanding the set of example programs and prepare a hands-on tutorial
- Testing is not perfect..
  - .. but about 60 functional and integration tests are executed in an automated way each release cycle
  - Feature requests now often come as a complete test case from the experiment. Thanks!
  - Number of high priority feature requests have not always left the time to turn each reported bug into a new test
- Performance optimisation not yet fully addressed
  - Performance tests now exist for all components (addressed in June release)
  - …but not enough time for in depth performance optimisation in all areas - focus stayed on functionality longer than anticipated.
  - External design and code reviews setup for use of ROOT I/O and for Object cache
- Schema Evolution and Stability of File Format
  - Current strategy relies fully on ROOT I/O facilities
    - The use of ROOT I/O as black box makes more generic schema evolution support non trivial
  - POOL does not fully control the file format, but can help to detect unwanted format changes during regression testing

Component Architecture

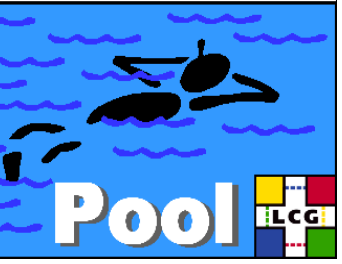
- POOL is a component based system
  - follows the LCG Architecture Blueprint
- Provides a technology neutral API
  - Abstract component C++ interfaces
  - Insulates the experiment framework user code from several concrete implementation details and technologies used today
- POOL user code is not dependent on implementation libraries
  - No link time dependency on implementation packages (e.g. MySQL, Root, Xerces-C..)
  - Backend component implementations are loaded at runtime via the SEAL plug-in infrastructure
- Three major domains, weakly coupled, interacting via abstract interfaces

POOL Component Breakdown



Work Package Breakdown

- Storage Manager
  - Streams transient C++ objects into/from disk storage
  - Resolves a logical object reference into a physical object
  - Uses Root I/O. A proof of concept with a RDBMS storage manager prototype underway
- File Catalog
  - Maintains consistent lists of accessible files (physical and logical names) together with their unique identifiers (FileID), which appear in the object representation in the persistent space
  - Resolves a logical file reference (FileID) into a physical file
- Collections
  - Provides the tools to manage potentially (large) ensembles of objects stored via POOL persistence services
    - Explicit: server-side selection of object from queryable collections
    - Implicit: defined by physical containment of the objects



POOL Work Package Presentations

- Storage Manager
- File Catalog
- Collections
- Integration & Testing