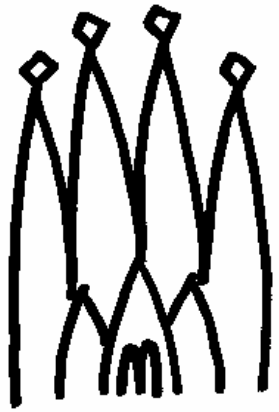
Status of the GAUDI event-processing framework

CHEP'01, September 3 - 7, 2001 Beijing, CHINA

M.Cattaneo[1], M.Frank[1], **P.Mato[1]**, S.Ponce[1], F.Ranjard[1], S. Roiser[1], I. Belyaev[1,2], C. Arnault[3], P. Calafiura[4], C.Day[4], C. Leggett[4], M. Marino[4], D. Quarrie[4], C. Tull[4]

[1] CERN, Geneva, Switzerland; [2] ITEP, Moscow, Russia;
[3] LAL, Orsay, France;[4] LBNL, Berkeley, USA



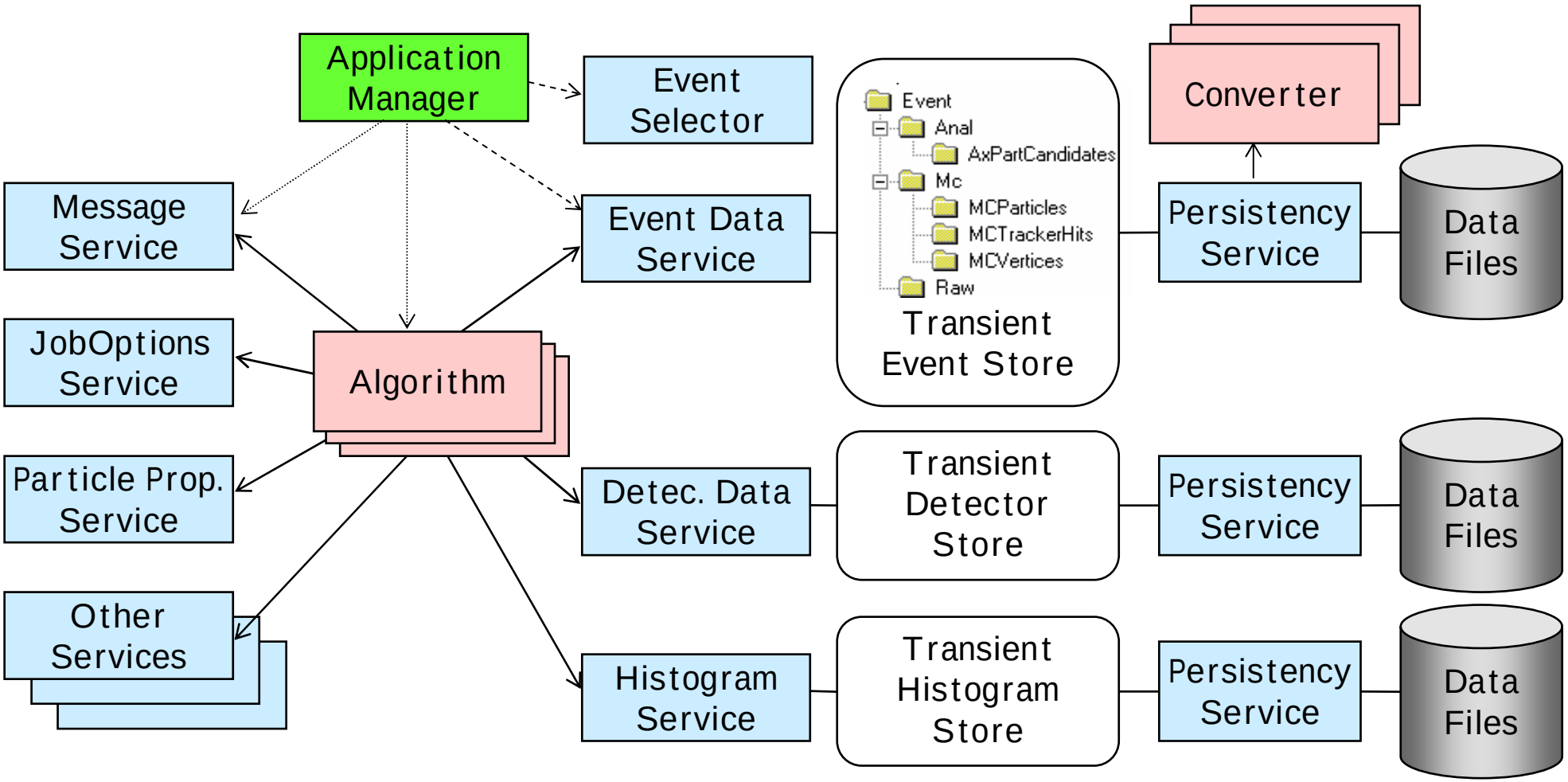
Contents

- What is GAUDI, History, …
- Extensions since CHEP'00
- Setting-up a Collaboration
- Difficulties
- Conclusions

GAUDI Architecture

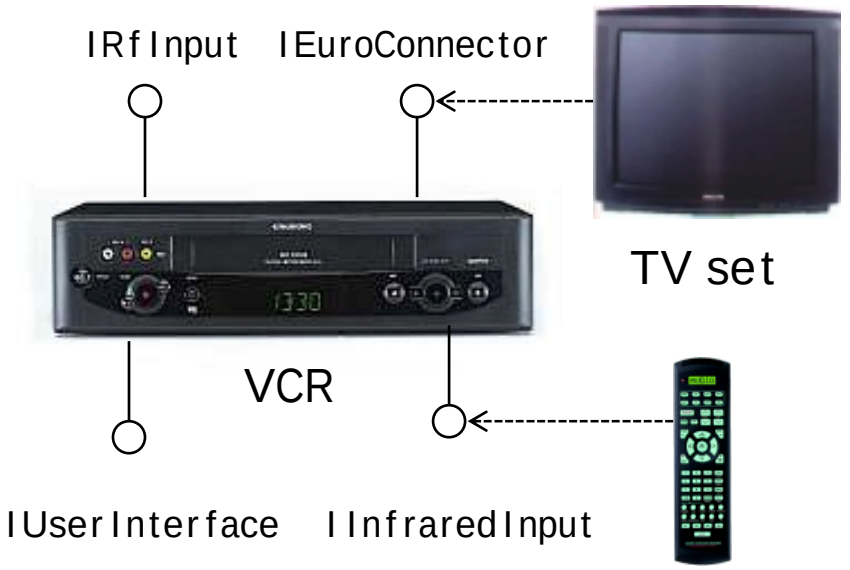
- GAUDI is an **architecture** and **framework** for event-processing applications (simulation, reconstruction, analysis, etc.)
- Principal design choices
  - Separation between “data” and “algorithms”
  - Three basic categories of “data”: event data, detector data, statistical data
  - Separation between “transient” and “persistent” representations of data
  - Data store-centered (“blackboard”) architectural style
  - “User code” encapsulated in few specific places
  - Well defined component “interfaces”

Gaudi Architecture: Object Diagram



Interfaces

- Components (services, algorithms, etc.) with abstract interfaces
  - C++ pure abstract classes
- Alows:
  - Evolution/replacement of the implementation
  - Runtime selection among several existing implementations



Development Process

- Followed more or less the Unified Software Development Process (USDP)
- Architecture centric
  - GAUDI architecture design document
- Use-case driven
  - Started with a set of architecturally relevant use-cases
  - Feedback and priorities provided by users
- Iterative and incremental
  - Initial implementations very simplistic. Refinements came later.
  - 3 releases a year with increased functionality. Replacement of implementations.

History

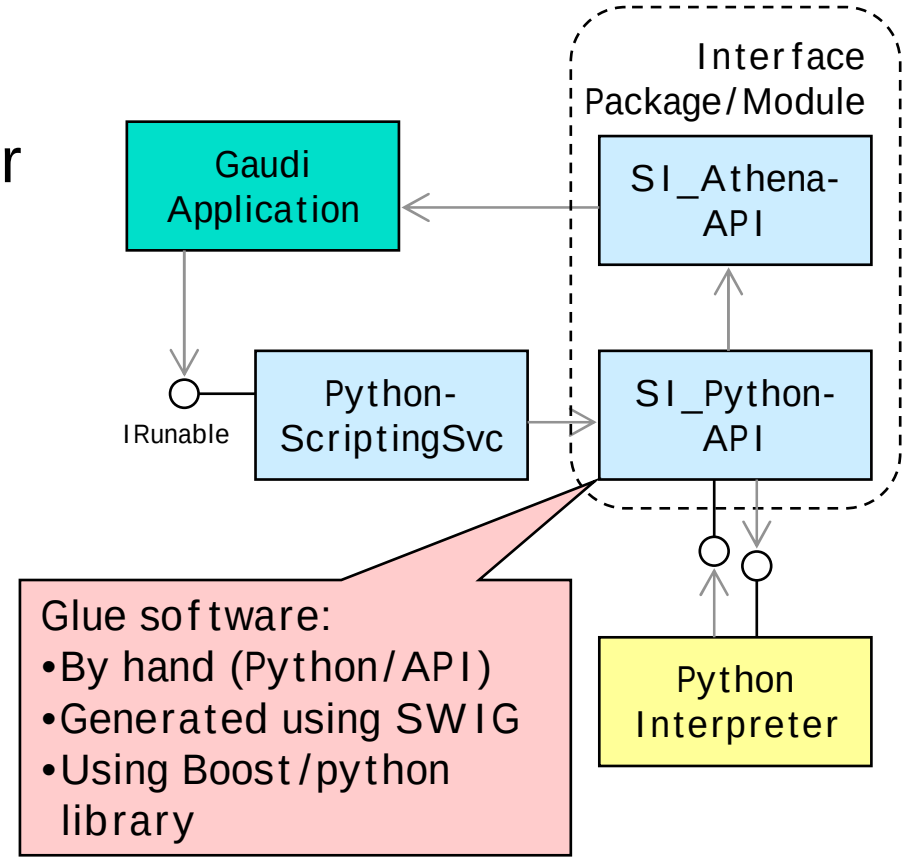
- Sep '98 Started development in the context of the LHCb experiment
- Feb '99 First GAUDI release
- Nov '99 Third release. Functionally complete version. Start deployment in LHCb
- Started being used by HARP, GLAST, ATLAS
- Nov '00 Version v6 released
- Mar' 01 Split into a common part and a LHCb specific part. Version v7 released
- Jul '01 Version v8 released

Recent Framework Enhancements

- Interfacing Gaudi to a scripting language
- Complex control sequences
- Extensions to Algorithm and Service *properties*
  - Validity range checking, read and write callbacks, remote properties, etc.
- Add a number of new general purpose services
  - Resource monitoring (Auditor service), Histo/Ntuple persistency service for ROOT and HBOOK, etc.
- Improved support for dynamic-loading of shared libraries: *component libraries*
- Improved documentation and Tutorials

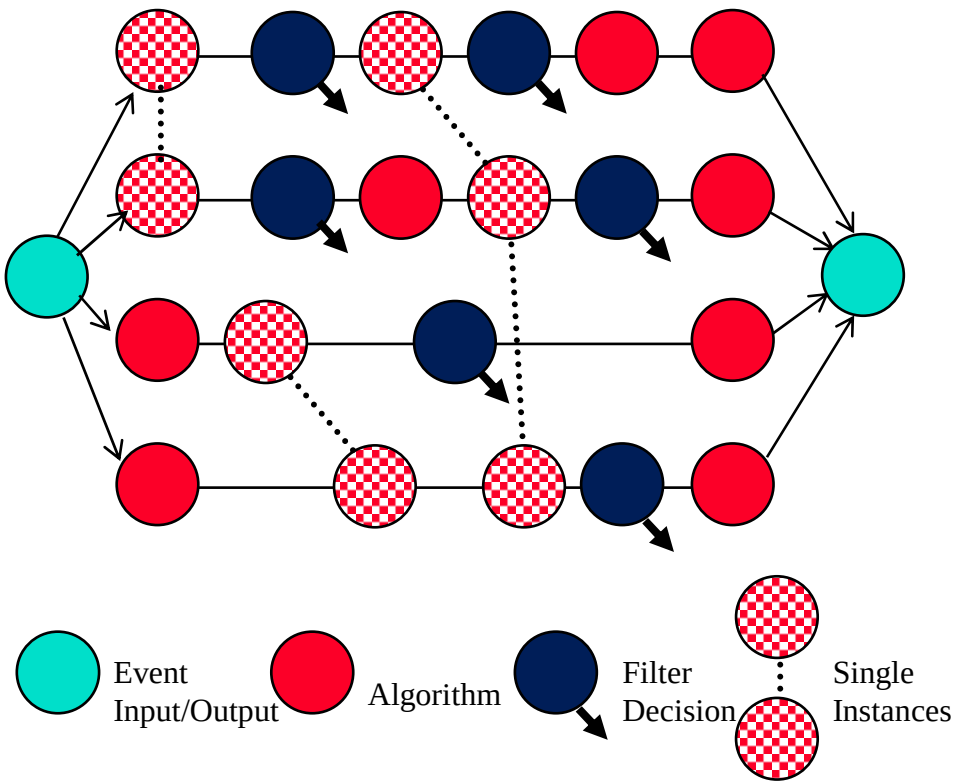
Interfacing Python to Gaudi

- Adding “interactivity” to a Gaudi application
- Python as one possibility for adding scripting capability to GAUDI
- Python interpreter takes control of the Gaudi application
- See C.Day et al. *Adding a Scripting Interface to Gaudi* (paper 3-065)



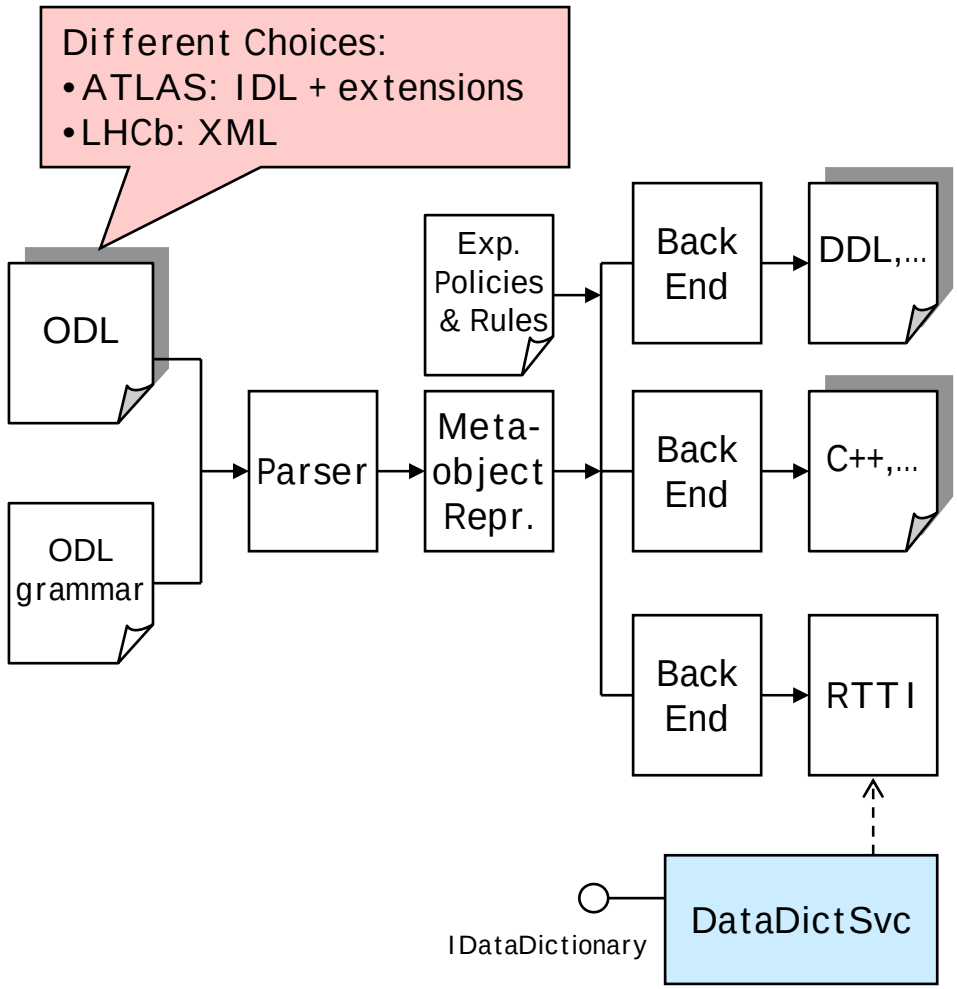
Complex Control Sequences

- Concept of sequences of *Algorithms* to allow processing based on physics signature
  - Avoid re-calling same algorithm on same event
  - Different instances of the same algorithm possible
- Event filtering
  - Avoid passing all the events through all the processing chain



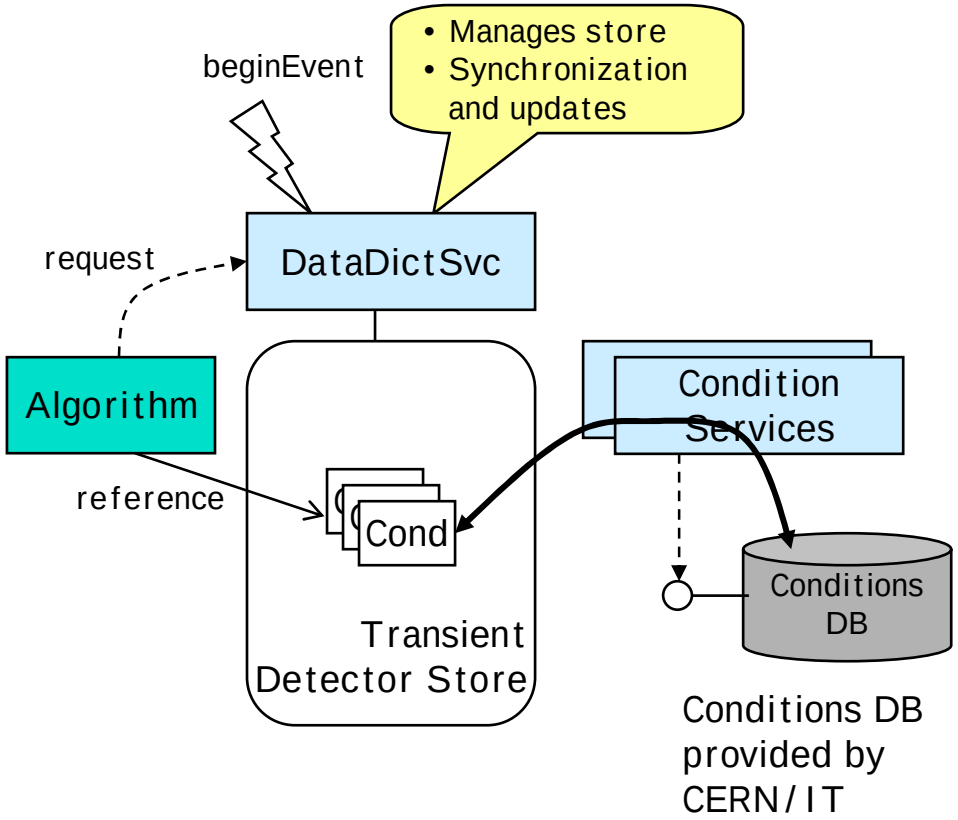
Ongoing work: Data Dictionary

- Introducing an Object Definition Language independent of the programming language
- The goals are:
  - Code Generation:
    - » C++ header
    - » Java/Python classes
    - » Persistency Converters
    - » ...
  - Introspection/Run-time Type Information
    - » Interactivity and Scripting
- See A. Bazan et al. *"The Athena Data Dictionary and Description Language"* (4-051).



Ongoing work: Detector Conditions

- Detector conditions: time dependent information
  - Calibration, alignment, environmental parameters (temperature, pressure, etc.), detector HV parameters, dead channels, etc.
- Interfacing to GAUDI the Conditions DB provided by CERN/IT
  - Framework takes care of the synchronization with current "Event" time
  - Make life easy to Algorithm writers
  - The actual representation of the condition data may be different for different experiments

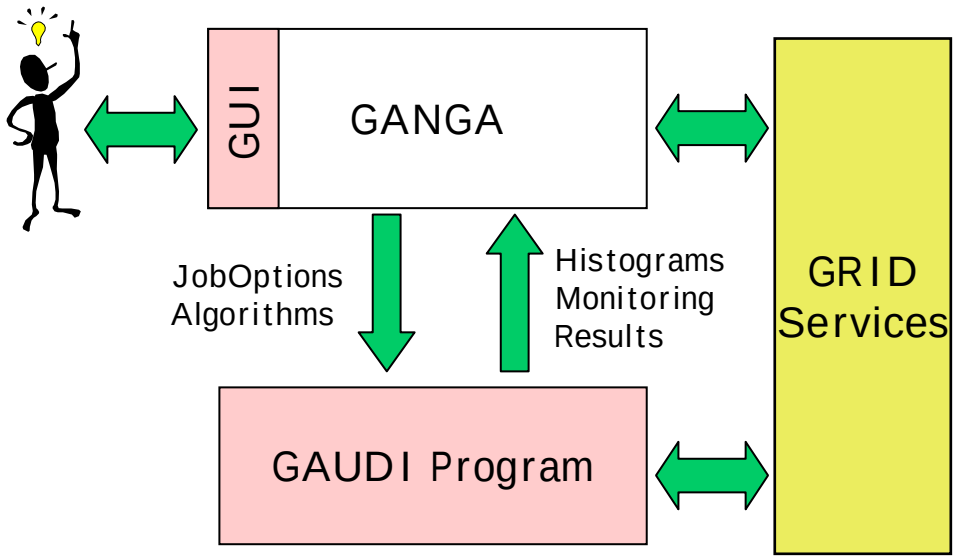


Ongoing work: Integrating Geant4

- Building the experiment simulation application around GAUDI requires integration with the Geant4 toolkit
  - Single source of detector geometry information
  - Re-use simulation components in other applications (reconstruction, analysis, etc.)
- Number of plugable components (*services)* being developed
  - To encapsulate the G4RunManager (configuration and control), G4 Actions (customization).
  - Conversion of transient objects to/from Geant4 representation
- Most of these services are experiment independent
- See I. Belyaev et al. *"Integration ofGeant4 with Gaudi"* (5-009)

On going work: Interfacing to the GRID

- Making the framework work in the GRID environment requires:
  - Collecting use-cases and architectural design
  - Identify the [Gaudi] components that need to be adapted/re-implemented to make use of the Grid services
- Insure that the framework is “Grid-capable" without being “Grid-dependent"
- Started to identify areas of work:
  - Data access (persistency)
  - Event Selection
  - GANGA (job configuration & monitoring, resource estimation & booking, job scheduling, etc.)



Setting-up a Collaboration

- Collaboration between ATLAS and LHCb
  - To develop and maintain the common set of packages from which the specific experiment frameworks (ATHENA, LHCb Framework) are based
  - Open to the other experiments also using the framework
- Expected benefits:
  - Development sharing
  - Better product quality
    - » Different environments/platforms/problem domains
    - » Better use-case coverage
    - » Diverse design teams
  - Maintenance sharing

Common software repository

- Identified the need to have a single common software repository decoupled from the experiment repositories
  - Consisting of all the experiment-neutral packages (abstract interfaces, base classes, common services, etc.)
  - Avoids divergences, thus easing later maintenance
  - Facilitates re-use (migration of experiment developed services to common services)
  - Its own set of rules and release schedule
- Done first classification of experiment-specific and experiment-neutral packages

Project Area

- AFS space at CERN (/afs/cern.ch/sw/Gaudi)
- CVS repository
  - Own set of policies and conventions
  - CVS server for remote access
- Configuration management and build system based on CMT(*)
- Release area
  - Package organization given by CMT
  - Supported platforms: Linux, Windows, (Solaris)
  - Directly used by experiments (LHCb, HARP, ATLAS soon)
- Web site (http://cern.ch/gaudi)

(*) See C.Arnault et al. "*Experiencing CMT …*" (8-006)

Project Management

- Lightweight structure to manage the project
- Three roles identified:
  - Project Manager: coordination, priority definition, keeps project focused on the right goal
  - Software Architect: leads technical activities and establishes the overall structure
  - Librarian: supports the development activity
- Scheduling releases and defining their contents done by the contributing experiments by consensus
- Regular meetings are essential

Difficulties

- Differences in the requirements and constraints in the different experiments
  - Find compromise. Different implementations for selected parts (e.g. event data store)
- Differences in the development environment (tools, platforms, compilers, etc.)
  - Big advantage to use same tools (e.g. CMT)
- Dependencies on external packages: finding a coherent set.
- Different coding rules and conventions

➔ Resolving potential problems requires good communication between development teams and clear common goal.

Conclusions

- The common Gaudi kernel packages are released a few times per year with added functionality developed mainly by ATLAS and LHCb core software development teams.
- The Gaudi framework is used by several other experiments that contribute with valuable feedback.
- A common project organization is in place and has been kept very lightweight.
- The project area provides the common software repository and release area.
- Perhaps it is too early to see the full benefits of this collaboration but we are convinced that it will pay in the long term.