Software Review Panel

LHCb answers to non-experiment specific software questions

CAVEAT:

All answers are preliminary and subject to discussion in the collaboration

Preamble

- **LHCb production software has, until this year, been entirely FORTRAN based**
  - Uses many third party FORTRAN libraries
- **During the last 18 months we have developed an OO architecture (GAUDI)**
  - Designed to shield user applications from choices of third party products
- **Migration of production software from FORTRAN to C++ has started**
  - Migration planning dictated by TDR commitments
  - Some FORTRAN likely to be with us until at least end 2001
- **Evaluating existing third party components**
  - And drafting user requirements if we don't find what we need

Model for reuse of "external software"

- Try to understand what we need
- Look for existing components that we can reuse
  - HEP libraries, commercial libraries
- Assess impact of component on GAUDI architecture
  - Inter-working with existing components
- Assess impact on long term maintenance
  - support, financial cost, risks etc.
- Develop the component ourselves only if we do not find anything suitable

Q4.1 Upon which third party software is LHCb relying?

- Event generators
  - Present:
    - Pythia 6.1
    - QQ 9.2 (CLEO MonteCarlo) for decays
    - STDHEP to interface to GEANT 3
  - Future:
    - Pythia 7.x, Herwig++
    - BPACK for decays
- Simulation
  - Present:
    - GEANT 3, members of GEANT 4
  - Future:
    - Migrate to GEANT 4 during 2001

Foundation libraries

- **Present:**
  - CERNLIB
  - C++ STL, CLHEP, (NAG C)
- **Future:**

  Use foundation libraries that have wide acceptance in HEP community (common language)
  - C++ STL
  - CLHEP
  - AIDA interfaces
    - used to access other third party components
    - use components suggested by "LHC++" wherever possible

Persistency

- **Present:**
  - ZEBRA + Root/IO for event data
  - ORACLE for book-keeping data
  - Flat ASCII files for geometry (XML format)
    - Parsed by commercial XML parser
- **Future:**
  - GAUDI Architecture shields applications from storage technology
    - Choose technology best suited to type of data
    - Can wait until risks are better understood
  - Does NOT shield us from potential need to migrate vast quantities of data to a new technology
    - Choose same technology adopted by other large experiments
    - Rely on CERN-IT to make risk assessment
    - In worst case, expect CERN-IT to organise data migration

Data distribution

- Assume that tools will be provided for distributed data access, remote job submission etc.
- Expect this to be outcome of projects such as MONARC and the grid initiatives

GUI, visualisation, interactive analysis

- **Present:**
  - HBOOK/PAW for histogram+ntuple analysis and storage
  - ROOT for some test beam analysis
  - Evaluating WIRED and OpenScientist
- **Future:**
  - Intend to use public domain and/or HEP software
  - Strong requirement is possibility to interact with data in GAUDI transient stores
  - No decision yet

Software Development Environment

- Platforms, compilers:
  - Visual C++ on WNT/W2000
  - g++ on RedHat Linux
- Code repository
  - CVS on AFS
  - WinCVS plus Windows AFS client to access from WNT
- Configuration management
  - CMT
- Design tools:
  - Visual Thought, Rational Rose
- Documentation
  - Framemaker for guides/manuals
  - Object Outline for code
- Bug reporting and tracking
  - Plan to use IT division Remedy service

Q 4.2 Service and support agreements?

- **Requirements:**
  - Software must run on all current and future supported platform/compiler combinations
    - Linux, WNT for foreseeable future
  - Bug fixes and enhancements on reasonable timescale
  - Guaranteed support for lifetime of the experiment
    - Or provide alternative solution (new product, source code)
  - Equal access and licensing terms to all collaborating institutes
- **Minimise LHCb specific agreements**
  - Prefer to adopt LHC-wide or HEP-wide solutions
    - And collaborate to evaluate/develop/maintain these solutions
  - Maintain in-house only what is strictly LHCb specific

4.2 - role of IT in this?

- **CERN-IT should agree, with the experiments, a set of supported components**
  - Providing a "guarantee" that support requirements are met
    - By negotiating appropriate licensing deals with commercial providers
    - By negotiating support agreements with other HEP providers
    - By supporting in-house other mission critical components, even if developed elsewhere (e.g. open source)
    - By identifying replacements for obsolescent components
  - And set up the necessary distribution infrastructure
    - Including HEP-wide licensing
- **CERN-IT should accept to be central distribution point for widely used HEP software**
  - Even if contributed from outside CERN
  - Even if not officially supported
    - With no commitment on content or support

4.3 What are the risks?

- **Commercial software**
  - Provider ceases to support the software
    - Either through choice, or because it ceases to exist
  - Provider supports a set of platforms / OS versions / compilers which are incompatible with other components
  - Software misses required functionality and/or has bugs
    - Which are not added/fixed within reasonable delay or at reasonable price
- **Above points must be addressed when negotiating purchasing and licensing agreements**
  - Particularly where niche products are involved
  - Access to source code must always be possible
    - Either directly or through escrow agreements
      - In case provider ceases support, or ceases to exist
      - In case provider reacts too slowly in fixing major problems
  - Rely on CERN-IT to carry out negotiations

- **Public domain / HEP community software**
  - Similar potential problems as with commercial software
  - No leverage on authors to provide required support
- **BUT**
  - Can (must) make sure source code is freely available
  - Preferably following open source model
    - So that, if necessary, locally made bug fixes or enhancements can be fed back to public code base
- **Risk: might end up maintaining large part of this**
  - Choose software which has wide acceptance in HEP
  - Form collaborations to develop and maintain HEP software
- **CERN-IT should play major role in fostering collaborative efforts**

Development of new software

(a.k.a. Q 4.4, 4.5)

- **Most of the software that we are likely to use in 2005 is currently in the R&D or prototyping phase**
- **We need to make sure that what is developed by third parties fulfils our requirements**
- **How can LHCb (and other experiments) collaborate with CERN-IT (and other laboratories) to develop the components that are missing?**

LHCb participation in development of new software

- **R&D phase**
  - informal approach appropriate - explore, prototype - quick & dirty
  - LHCb role is to follow progress and give suggestions
  - need good visibility
  - meetings for information exchange with unlimited participation
- **Product development phase**
  - to be sure we can use product, more formal approach required
  - as users we are part of project with a formal role to play
  - users express needs in written form e.g. requirements and use-cases
  - total visibility - follow all important decisions
  - formal project meetings with limited participation
- **Transition between R&D and Product development needs to be made at the right moment, not too late**

Observations on existing (LHC++) meetings

- **Work well as a forum for information exchange**
- **Discussion often throws up different points of view**
- **Not clear how this input is used in planning work programme**
- **LHCb not always well-prepared and correctly represented to take part in crucial decisions**
- **There can be potential dangers**
  - some idea or suggestion is picked up and acted upon without thorough discussion
  - mechanism for setting priorities unclear
  - decisions based on incomplete and imprecise information

A possible model for development projects

- Scope of each project specific to one component
  - e.g. histogramming, espresso, particle properties
- Each component should conform to a baseline architecture as set out by the system architect
- Participation of project team members : typical roles to be filled include
  - project leader
  - developers
  - users e.g. 1 per experiment
  - architect, librarian, documentation manager, test manager

A possible model for development projects (2)

- Team discusses and agrees on tasks, priorities and responsibilities
- Publish everything (project plan etc) on web
- Frequent project meetings for project team only e.g. weekly in development phase
- Brief status reports at less frequent, open, information meetings e.g. monthly

Conclusions

- **GAUDI Architecture designed with third-party components in mind**
  - Abstract interfaces to allow plug and play solutions
- **LHCb keen to reuse third party software wherever possible**
  - And to minimise risks by choosing widely accepted solutions
- **LHCb keen to participate in development of common solutions**
  - But worried about definition and accountability of existing projects