LHC Computing Review

LHCb answers to the SPP questions-Round II

Document Version:	1
Document Date:	26 April 2000
Document Status:	Draft
Document Editors:	John Harvey, Pere Mato

Abstract

This is a working document for collecting the answers to the second round of questions of the Software Project Panel (SPP) for the LHC Computing Review.

Disclaimer

This document is a progress report towards answering the questions addressed by the software panel. We have not had the opportunity of consulting widely with our colleagues and we therefore reserve the right of modifying and developing the answers further.

1 Questions/Requests for all Experiments and CERN/IT

1.1 What are your expectations and recommendations how to optimize the use of common software products and solutions for more than one experiment?

Our motivation for using common software products would be to reduce our own development and maintenance load. Our expectations would therefore be that there is a commitment from participants to common projects to develop and maintain the product over timescales comparable to the lifetime of an LHC experiment.

How should the various phases of a potential ’project’ be started, resourced at the appropriate level, how should management oversight be structured and what kind of regular interaction would be helpful?

As already mentioned in the first round of answers, we believe that a fresh look should be taken at the way common projects are run. In the past the approach has been more appropriate to an R&D style of development. The existing forums for discussion are suitable for information exchange but the decision making process is not transparent. Since now the emphasis is on the development of real software products. A more formal approach should help to ensure that the products produced are eventually used by all four experiments.

A process for managing common software projects could be envisaged as follows:

- The first task is to identify items which are of interest to more than one experiment. This implies some sort of forum for discussion and for managing the ‘birth’ of new projects. This forum of discussion should also be used to monitor progress of running projects and to ensure the coherency among the different projects.
- Since the experiments have adopted different architectural styles, likely candidates for common development are toolkits and libraries that do not impose a particular software design or architecture.
- It is important that the development of the project is driven by user needs. The experiments should provide use-cases and requirements. The process should be incremental and iterative i.e. the software should evolve through a series of releases with feedback from users at each stage.
- Each project should be run on project lines with well-defined procedures for taking decisions on tasks, setting priorities and defining the strategy for the development. This implies regular project meetings attended by all persons directly involved in the project i.e. with specific roles to play. Each project would have a leader with overall responsibility for managing the project.

- Each experiment wishing to benefit from the common software should ensure that it contributes to the overall development, and at least should provide its uses cases and participate in the decision making.
- The project would be answerable to the managements of the participating groups. Typically these would come from the experiment collaborations and IT division. It could be envisaged that the project leader makes progress reports to a wider forum e.g. the discussion group mentioned above.

1.2 Please comment on the idea of software agreements and MoU's, what granularity of responsibility would you recommend, how should the resources be shared?

The timescale for any MoU for computing is clearly dictated by the timescale for producing the MoU for the experiment as a whole. LHCb expects to submit its Detector MoU to the Resource Review Board by October 2000, and we expect it to be signed by all parties by end 2001. Moreover all four LHC experiments have stated that they intend to produce the Computing TDRs towards middle/end of 2002. It is difficult to imagine signing meaningful MoUs in Computing before these TDRs are published.

Recent discussions in Panel 3 of this review have tended towards an assumption that the final MoU for computing would be prepared in 2003. The document to be produced as a result of this review should be considered as an Interim MoU specifying our intentions for the next few years, both in terms of resourcing software development and for satisfying interim simulation and analysis needs.

We assume that any commitment to take responsibility for building, maintaining and operating a detector, as specified in the Detector MoU, also implies responsibility for developing detector related software. Evidently this must be the case as a detailed understanding of the behaviour of the detector is required in order to write the software for controlling operating conditions and for calibrating, aligning and processing data collected.

In particular we see the following software components as coming under the responsibility of the institutes building the detector:

- embedded software for zero-suppression and signal processing
- software for configuring detector-specific readout components
- data quality monitoring software
- software for managing calibration of the detector
- software for controlling the operational state of the detector (high voltage sequencing, voltage for electronics, gas systems, temperature monitors, control of stepping motors etc.)
- detector geometry specification
- detector event model specification
- detector pattern recognition algorithms
- detector simulation algorithms
- detector alignment and off-line calibration procedures

Software not covered in the above list basically consists of all non detector-specific software. This includes the following components:

- Software for the data processing framework and foundation libraries.
- All aspects of distributed data management and computing i.e. database, mass storage, grid software
- Support of the software infrastructure i.e. project management, code librarian, release management, quality control, documentation, web mastership, management of data production activities. For release management we use a tool called CMT, written by C.Arnault from Orsay. The future development and maintenance of this tool will most likely guaranteed formally by Orsay.(see also Part II question 11)
- Global software that spans several different subdetectors e.g. tracking and particle identification.

The software frameworks and support activities, and the software infrastructure for providing the data handling and analysis strategy, involve a more rigorous application of software engineering practices and these activities should be staffed with people having an appropriate background and experience in modern software techniques. It is apparent that today there is a shortage of effort to cover all the tasks mentioned. It will be important to identify effort to undertake these important tasks and this should be described in the MoU. Although we expect that much of the effort will come from the CERN group, other LHCb institutes are participating already and there are also indications from other groups of their intention to work on core-software projects. We would expect that responsibility for some duties (e.g. production manager) could be taken for a limited period and might with time be cycled through several different groups. Our approach to solving this manpower shortage will be discussed in a collaboration meeting in the first half of May, and more precise information should be available after this meeting takes place.

Most importantly we would like it to be understood that we would be strongly against institutionalising responsibility for any physics related software, as we want to encourage initiative as much as possible to any physicist to contribute wherever he/she has interesting ideas. It is likely that this freedom to encourage widespread participation will cover software that is common to the detector as a whole, such as the tracking and particle identification systems.

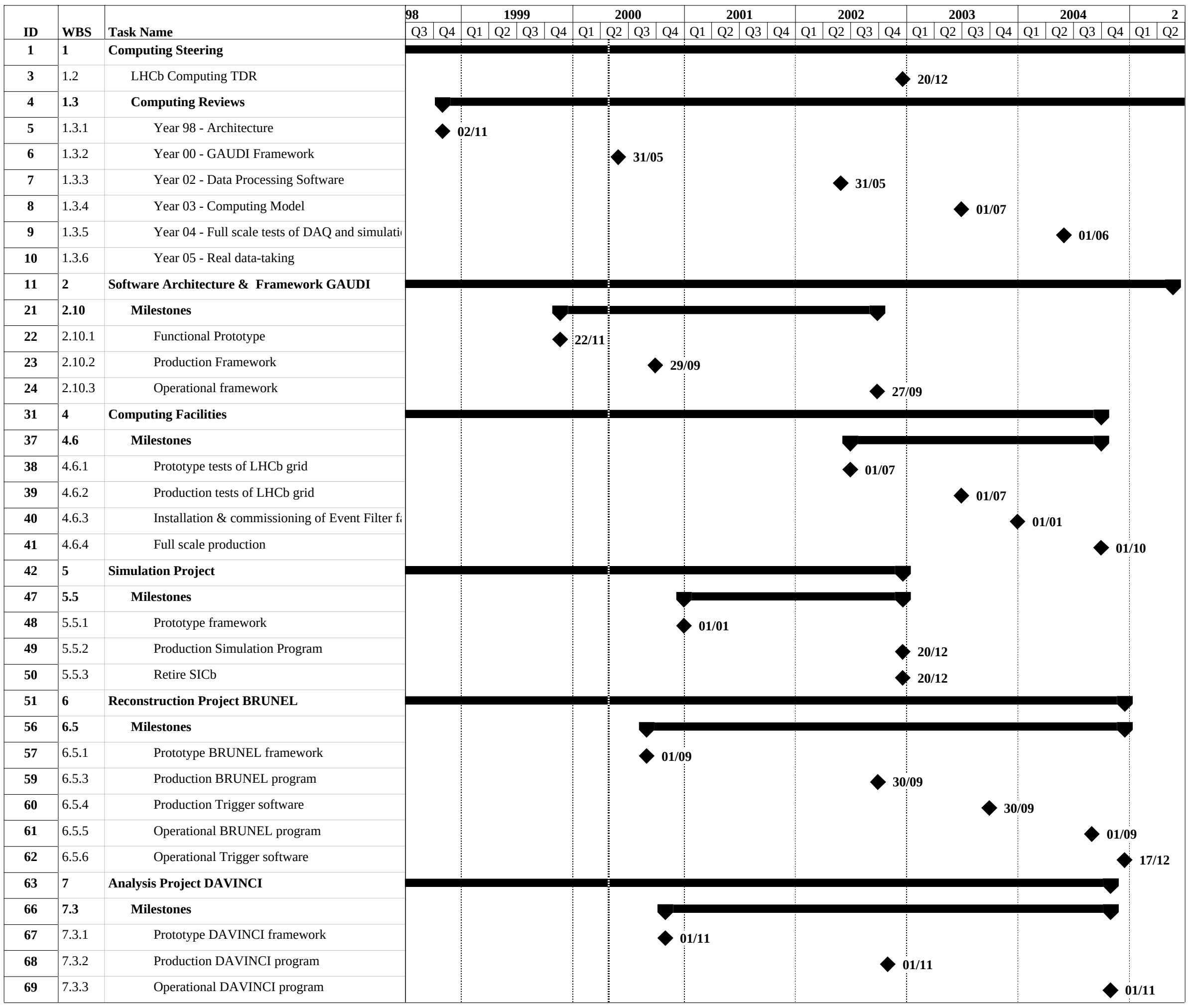
1.3 Please provide more detailed plans on

Project task breakdown

1	**Computing Steering**
1.1	LHCb Computing Coordination
2	**Software Architecture & Framework GAUDI**
2.1	GAUDI Project Coordination/ Architect
2.2	General Framework Services
2.3	Generic Event Model
2.4	Detector Description (structure, geometry)
2.5	Detector Conditions (calibration, slow control)
2.6	User interaction, GUI, scripting visualisation
2.7	Data Management (persistency/mass storage)
2.8	Data Management (bookkeeping)
2.9	Distributed data access / grid software
3	**Software Engineering Support**
3.1	Code management and distribution
3.2	Documentation management
3.3	Software test, quality, performance manager
3.4	Collaboration Tools
3.5	Training
4	**Computing Facilities**
4.1	Computing Model Project Coordination
4.2	Event Filter CPU farm
4.3	LAN Infrastructure at pit + CDR
4.4	OS system management
4.5	OS system administration
5	**Simulation Project**
5.1	Project coordination and simulation framework
5.2	SICb coordination (GEANT3 based)
5.3	GEANT4 framework
5.4	Data Production Management
6	**Reconstruction Project BRUNEL**
6.1	Reconstruction Project coordination
6.2	BRUNEL framework design
6.3	High level trigger framework
6.4	Software and data quality monitoring
6.5	Milestones
7	**Analysis Project DAVINCI**
7.1	Analysis Project coordination
7.2	Analysis framework design
8	**Event Display MONET**
8.1	Offline Event Display
8.2	Online Event Display
9	**Subdetector Data Processing Software**
9.1	Subdetector software coordination
9.2	Subdetector structure and geometry
9.3	Subdetector Event Model
9.4	Subdetector simulation algorithms
9.5	Pattern recognition algorithms
9.6	Subdetector alignment and calibration
10	**DAQ System**
10.1	Project Coordination/DAQ architect
10.3	Readout Unit Project
10.4	Timing and Fast Control
10.5	Event Builder Project
10.6	DAQ application software project
10.7	Hardware installation and commissioning
11	**Experiment Controls System ECS**
11.1	Project coordination/ECS architect
11.2	SCADA system development
11.3	Software utilities and tools
11.4	Gas system
11.5	Rack Control system
12	**Experiment Operations**
12.1	Control room installation
12.2	Standard operations software applications
12.3	LHC machine interface
12.4	Shift crew supervsion and training

Figure 1 Work breakdown structure of the LHCb Computing project

Resource loaded schedules and milestones



ID	WBS	Task Name	
1	**1**	**Computing Steering**	
3	1.2	LHCb Computing TDR	20/12
4	**1.3**	**Computing Reviews**	
5	1.3.1	Year 98 - Architecture	02/11
6	1.3.2	Year 00 - GAUDI Framework	31/05
7	1.3.3	Year 02 - Data Processing Software	31/05
8	1.3.4	Year 03 - Computing Model	01/07
9	1.3.5	Year 04 - Full scale tests of DAQ and simulati	01/06
10	1.3.6	Year 05 - Real data-taking	
11	**2**	**Software Architecture & Framework GAUDI**	
21	**2.10**	**Milestones**	
22	2.10.1	Functional Prototype	22/11
23	2.10.2	Production Framework	29/09
24	2.10.3	Operational framework	27/09
31	**4**	**Computing Facilities**	
37	**4.6**	**Milestones**	
38	4.6.1	Prototype tests of LHCb grid	01/07
39	4.6.2	Production tests of LHCb grid	01/07
40	4.6.3	Installation & commissioning of Event Filter f	01/01
41	4.6.4	Full scale production	01/10
42	**5**	**Simulation Project**	
47	**5.5**	**Milestones**	
48	5.5.1	Prototype framework	01/01
49	5.5.2	Production Simulation Program	20/12
50	5.5.3	Retire SICb	20/12
51	**6**	**Reconstruction Project BRUNEL**	
56	**6.5**	**Milestones**	
57	6.5.1	Prototype BRUNEL framework	01/09
59	6.5.3	Production BRUNEL program	30/09
60	6.5.4	Production Trigger software	30/09
61	6.5.5	Operational BRUNEL program	01/09
62	6.5.6	Operational Trigger software	17/12
63	**7**	**Analysis Project DAVINCI**	
66	**7.3**	**Milestones**	
67	7.3.1	Prototype DAVINCI framework	01/11
68	7.3.2	Production DAVINCI program	01/11
69	7.3.3	Operational DAVINCI program	01/11

Figure 2 Main milestones of the software project

Resource profiles up to 2005, especially personnel (including an estimate on software professionals and physicists contributing to software effort)

See figure Figure 3

WBS	Task	Profile	1999	2000	2001	2002	2003	2004	2005	2006
1	***Computing Steering***									
1.1	LHCb Computing Coordination	E	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Subtotal (FTEs)		**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
2	***Software Architecture & Framework GAUDI***									
2.1	GAUDI Project Coordination/ Architect	E	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2.2	General Framework Services	E	2.0	2.0	2.0	1.0	1.0	1.0	1.0	1.0
2.3	Generic Event Model	E	1.0	1.0	0.5	0.5	0.25	0.25	0.25	0.25
2.4	Detector Description (structure, geometry)	E	0.5	1.0	1.0	0.5	0.25	0.25	0.25	0.25
2.5	Detector Conditions (calibration, slow control)	E	0.0	0.5	1.0	1.0	1.0	0.25	0.25	0.25
2.6	User interaction, GUI, scripting visualisation	E	0.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0
2.7	Data Management (persistency/mass storage)	E	1.0	1.0	2.0	2.0	1.0	1.0	1.0	1.0
2.8	Data Management (bookkeeping)	E	0.0	1.0	1.0	1.0	1.0	0.25	0.25	0.25
2.9	Distributed data access / grid software	E	0.0	0.0	1.0	1.0	1.0	0.5	0.5	0.5
	Subtotal (FTEs)		**5.5**	**8.0**	**10.5**	**9.0**	**7.5**	**5.5**	**5.5**	**5.5**
3	***Software Engineering Support***									
3.1	Code management and distribution	E	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3.2	Documentation management	E	0.0	0.5	0.5	0.5	0.5	0.5	0.5	0.5
3.3	Software test, quality, performance manager	E	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0
3.4	Collaboration Tools	E	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
3.5	Training	E	0.0	0.0	0.5	0.5	0.5	0.5	0.5	0.5
	Subtotal (FTEs)		**1.5**	**2.0**	**3.5**	**3.5**	**3.5**	**3.5**	**3.5**	**3.5**
4	***Computing Facilities***									
4.1	Computing Model Project Coordination	E	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4.2	Event Filter CPU farm	E	0.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0
4.3	LAN Infrastructure at pit + CDR	E	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0
4.4	OS system management	E	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4.5	OS system administration	E	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Subtotal (FTEs)		**3.0**	**3.5**	**4.0**	**4.0**	**5.0**	**5.0**	**5.0**	**5.0**
5	***Simulation Project***									
5.1	Project coordination and simulation framework	P	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5.2	SICb coordination (GEANT3 based)	P	1.0	1.0	0.5	0.5	0.0	0.0	0.0	0.0
5.3	GEANT4 framework	P	0.0	0.5	2.0	1.0	1.0	1.0	1.0	1.0
5.4	Data Production Management	E	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	Subtotal (FTEs)		**3.0**	**3.5**	**4.5**	**3.5**	**3.0**	**3.0**	**3.0**	**3.0**
6	***Reconstruction Project BRUNEL***									
6.1	Reconstruction Project coordination	P	0.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0
6.2	BRUNEL framework design	E	0.0	0.5	1.0	1.0	1.0	1.0	0.5	0.5
6.3	High level trigger framework	P	0.0	0.0	1.0	1.0	1.0	1.0	0.5	0.5
6.4	Software and data quality monitoring	P	0.0	0.0	0.5	0.5	0.5	0.5	1.0	1.0
	Subtotal (FTEs)		**0.0**	**1.0**	**3.5**	**3.5**	**3.5**	**3.5**	**3.0**	**3.0**
7	***Analysis Project DAVINCI***									
7.1	Analysis Project coordination	P	0.0	0.5	0.5	1.0	1.0	1.0	0.5	0.5
7.2	Analysis framework design	E	0.0	0.5	1.0	1.0	1.0	1.0	0.5	0.5
	Subtotal (FTEs)		**0.0**	**1.0**	**1.5**	**2.0**	**2.0**	**2.0**	**1.0**	**1.0**
8	***Event Display MONET***									
8.1	Offline Event Display	E	0.0	0.0	1.0	1.0	0.5	0.5	0.5	0.5
8.2	Online Event Display	E	0.0	0.0	0.0	0.5	0.5	1.0	1.0	0.5
	Subtotal (FTEs)		**0.0**	**0.0**	**1.0**	**1.5**	**1.0**	**1.5**	**1.5**	**1.0**
	Subtotal (FTEs) for core Computing		**14.0**	**20.0**	**29.5**	**28.0**	**26.5**	**25.0**	**23.5**	**23.0**
9	***Subdetector Data Processing Software***									
9.1	Subdetector software coordination	P	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
9.2	Subdetector structure and geometry	P	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
9.3	Subdetector Event Model	P	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
9.4	Subdetector simulation algorithms	P	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
9.5	Pattern recognition algorithms	P	1.0	1.0	2.0	2.0	2.0	1.0	1.0	1.0
9.6	Subdetector alignment and calibration	P	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0
	Subtotal (FTEs)/ subdetector		**4.0**	**4.0**	**5.0**	**5.0**	**5.0**	**5.0**	**5.0**	**5.0**
	Subtotal (FTEs) for all detectors (V,T,R,M,C,T)		**24.0**	**24.0**	**30.0**	**30.0**	**30.0**	**30.0**	**30.0**	**30.0**

Figure 3 LHCb computing project manpower requirements

1.4 What are the criteria for accepting a new third party package into the experiment’s software system?

Technically it is always possible to incorporate a third-party package into the LHCb system. In practice we need to assess the risks involved in adding a new package against the new functionality it offers and the possibility of using alternative packages offering similar functionality. Some of these risks factors are:

- *Physical design issues.* The dependencies of the new package need to be studied. We need to understand what are the others packages this new package is relying on. Those packages will need to be also added into the system. The dependencies can be at compile time, link time or run time. Do the dependent packages create any incompatibility with other packages in use? Using packages that depend only on the set of foundation libraries used in LHCb would be an advantage.
- *Licensing and maintenance.* Who maintains and supports the new package? Having a strong support behind a package is a very important point for its acceptance. If commercial, what is the cost? In general, what are the risks associated with commercial software? Access to the source code is an advantage.
- *Platforms and compilers issues.* The new package should be supported on the experiment’s selected platforms and compilers.
- *Compliance with the experiment coding rules and guidelines.* The new package, especially the interface, should be compatible with the experiment’s policies in terms of languages and standards (i.e. ANSI C++).

2 Questions/Requests for all Experiments

2.1 Please formulate a statement expressing the experiment's requests for the future evolution of the GEANT4 collaboration and project. Explain how this will meet your needs for validation, further development and ongoing support.

Until recently the main effort of the GEANT4 collaboration has been directed towards the development of the toolkit and ongoing work is driven by the core team of developers working full-time on GEANT4. We have already had several discussions in the GEANT4 collaboration board encouraging the development team and the collaborations to now work more closely together. This should have benefits on both sides:

- Simplify the uptake of GEANT4 by the collaborations. In the past there have always been GEANT3 experts on-hand nearby to answer questions. The situation is now completely different, as almost nobody accessible to LHCb physicists has experience with GEANT4. Discussions have already taken place between LHCb and the GEANT4 spokesman to organise start-up training now. More specialised training may also be considered in the future if needed. A GEANT4 FAQ page has been created very recently. We think these initiatives are all very positive and strongly encourage further efforts in this direction. In addition having a GEANT4 expert act as a contact person for each experiment would certainly help if this was feasible. This person is not supposed to do the work for the experiment but will know the experiment people and the specific troubles the experiment is finding. He would also act as a link between GEANT4 experts and the experiments.
- Feedback from the experiments will be valuable for testing the physics and other components of GEANT4. A number of projects explicitly involving various other users of G4 (ATLAS, CMS, BaBar, ESA) were announced at the last collaboration board meeting. These projects will provide invaluable feedback to the GEANT4 team. LHCb is just now devising plans and allocating effort towards developing its new simulation program and this will be done over a two year time period. In addition, and on a shorter time scale, simulations of test-beam setups will be done and comparisons made with existing simulation results, based on GEANT3, and on testbeam data. Some of these limited size projects have already started (e.g. calorimeter). These tests may provide fast user feedback to GEANT4 about the simulation of the specific detectors LHCb will use in 2005. Meanwhile some LHCb users will acquire experience which will be very useful to write the LHCb simulation program.

Several of us have had various discussions with people both inside and outside our collaboration questioning whether an Open-Source approach should be formally adopted. The GEANT4 core team would control, filter and integrate contributions.

We hope that these initiatives will result in a good response to our requests, both from a user support perspective as well as towards directing further developments of the GEANT4 toolkit. We also reiterate our request that there should be a GEANT4 user-support service based at CERN for the use of those experiments using GEANT4. We assume that these considerations may naturally lead to some modifications of the MoU as the project enters the next phase in its life-cycle.

2.2 What part do you expect/hope to play in the evaluation of the results of Espresso and the decision process for what should be done about providing a common Object Storage solution for one or more LHC experiments?

If a common object storage solution is to be provided, LHCb should be actively involved in the definition of the requirements (use cases), in the architectural design discussions, in the implementation choices, in testing and evaluating the various prototypes, etc. This applies to whatever solution is adopted, whether it is based on a commercial product or on a home-made product.

We are of the opinion that more than one object storage solution should be available to the LHC experiments, each one having a different range of applicability. For example, we can imagine a full-fledged solution for the experiment main data store capable of storing petabytes distributed worldwide (security, transactions, replicas, etc.). On the other extreme we need a much lighter solution for end-physicists doing the final analysis with his own private dataset. Perhaps a single solution can cover the complete spectrum, but in general this may not the case. Therefore, we see the Expresso or some other alternative i.e. ROOT I/O in coexistence with a more functional, thus heavier, solution. Moreover the lightweight solution can be also seen as a starting point for developing a backup solution of the full-fledged system in the case of necessity.

2.3 Please provide a brief statement of what you would consider to be the scope and deliverables for a common project aimed at a Geometry Description system. What would be your goals in participating in such a common project and what roles would you be prepared to play?

Different expriments have very similar requirements when it comes to making data models describing their detectors and we would therefore agree that this subject would be a very good candidate for a joint project. If all the experiments cooperate on the common solution it would dramatically improve the speed of progress in design and implementation of the needed tools and frameworks.

The scope of such a project should be to produce a framework (or toolkit) for describing the detector logical structure and geometry of any experiment, integrable to the experiment's main software framework. This should include the persistency mechanisms to a detector description database, tools for populating the database from popular sources (e.g. CAD systems), semantic-aware editors with graphical capabilities, etc.

At present LHCb uses XML as a persistency format for storing the detector description data and similar approaches are being followed in the other experiments. LHCb is willing to actively participate in this area concerning the common Document Type Definition (DTD) for detector description XML data, this being the first step or deliverable. By having a common DTD it would be possible to exchange some parts of the detector description which are identical or very similar between experiments (e.g. material tables) and to start investing in building the tools mentioned before (e.g. for visualisation and editing of the XML documents for detector description). These tools could be used by each of the experiments.

2.4 As a result of participating in this software panel, where your representatives have heard details of work going on in all of the other LHC experiments

Have you identified any new areas of commonality?

There are several areas where work involving more than one experiment is ongoing, although not necessarily as a result of the software review. Areas worth mentioning include the following:

- Work on the software framework is being done in conjunction with ATLAS. There are several new services being developed by the ATLAS core software group which we expect to integrate within our GAUDI framework.
- Detector geometry description framework and database. There is an interest to put in common the ideas for a detector geometry description based on XML. The first workshop, organized by Steven Goldfarb, took place on April 14th - see http://home.cern.ch/muondoc/software/Database/Meetings/XML/2000-04-14/Summary.htm.
- Conditions database. The same arguements used for proposing the geometry description as a common project also apply here. Work on this is on-going in the context of the LHC++ project, using the package originating in BaBar as a starting point.
- Design of the overall data management system (see 2.5).
- Particle properties service based on the PDG tables.
- There is some interest in HepMC package developed by ATLAS. The HepMC package is an object oriented event record written in C++ for High Energy Physics Monte Carlo Generators
- BPACK project. A package for handling B-decays based on EvtGen. Collaboration with ATLAS, CMS and LHCb (http://home.cern.ch/~msmizans/production/Bpack).
- Software development process tools. There is a need for a code checking tool, software release tool, bug tracking tool (IT is prototyping a service based on the commercial tool Remedy), etc.
- A discussion forum on use of JAVA has recently been setup by Steve Fisher.
- Grid software and the EU proposal.

Have you formulated any opinions on what forums would encourage continuing dialogue and interchange of ideas and software between experiments?

Technical forums have been proposed as a way of encouraging continuing dialogue. Forums that have existed in the past and which have had some success include:

- ***Thursday Club:*** This was setup and run by Jim Virdee. All 4 CERN team leaders of the 4 LHC experiments attended i.e. Jim, Chris Fabjan, PG Innocenti, etc. It acted as a forum for managing the birth of common projects. Having senior members of the collaborations present was considered to be an important ingredient for making this activity work.Several projects sprang up out of these discussions, some which are succeeding (e.g. JCOP) and some which failed (e.g. SPIDER). Some forum of this kind is needed if common projects are to be taken seriously (seealso answer to question 1.1).
- ***Architecture Study Group:*** This was another discussion forum which ran over the summer of 1999 for about eight lunchtime discussions. The software architects and relevant experts from the four LHC experiments, from RD45 and GEANT4 were present. The motivation was to prepare the discussion on architecture at the Marseilles workshop. Those attending said they found this useful and some even suggested it should continue in the future. This serves to illustrate that discussion forums involving people working directly on technical issues can be very useful for exchanging opinions and getting fresh ideas. This is very valuable even if it doesn't result in a formal common project to produce a piece of common software.

Have you identified any areas where your experiment could/should use software authored by one of the other experiments or IT division?

GEANT4, physics generators, physics analysis tools, HEP foundation class libraries. statistical packages, particle properties service, etc.

Have you identified any new concerns or areas of risk?

In general the most difficult problems to solve are organisational and not technical. Significant benefits can occur from working together in a collaborative manner. However, there are a number of outstanding issues concerning future directions for our software on which the community is still preoccupied (choice of Objectivity, use of Fluka together with GEANT4, support of ROOT,etc.) . These issues should be resolved as quickly as possible, so that we can all move forward together. Difficult management decisions may have to be made. People will need to take on roles for which thay are needed and focus all their energy on doing a good job there. A lot of give and take and goodwill will be required if this is to be achieved.

2.5 Data Management has to be an engineered system, it involves a component of a program framework, the bookkeeping and logging databases, the hardware systems and networks, and the interactions of the different parts. It involves

machines/disks/robots/tapes/networks and be subject to the wider effects of transaction management and sharing of certain central resources.

Who is designing this system and how are the responsibilities of the different aspects of the data management and data handling distributed?

As stated, the overall data management system needs to be coherently designed. It involves many components and a number of abstraction layers. For that we need to develop an architecture. If possible an architecture independent of the technologies and products to be used in the implementation. We need to define a number a layers or components with well defined interfaces and assign functionality to each on these layers. The responsibility for the overall design should be within an architecture team. It is essential that experts from specialised services from the IT divisions (disks, robots, networks, etc.) participate in the design of the system. A common forum of discussion involving all the experiments would be of course an advantage.

Only after the architecture has been defined, the responsibilities for the different components or layers can be assigned and distributed. The design of the data handling system architecture has not been started. It will be desirable to start during this year.

How are these people interacting with those determining the framework, architecture, and persistency mechanisms?

Big overlap between teams. The chief architect of the data processing application framework should be in the architecture team of the data handling system.

Which role do you expect from CERN/IT?

CERN/IT should play an important role in defining the system. After all, it is expected that IT will provide and operate the infrastructure. We expect from IT the following roles.

- Technical expertise.
- Leadership role in the architecture team.
- Service provider.

2.6 How big is the participation of physicists from the collaboration on software development? How are you trying to control unnecessary duplication of software developments?

Firstly we control unnecessary duplication by developing an architecture and implementing a software framework that respects that architecture. All standard services (application manager, statistical services, job options service, message service, etc.) are provided by the framework. Physicists concentrate on the pieces that they have to provide, pieces that are specific to their subdetector or physics algorithm. A generic model is provided for describing the structure of the detector and each individual detector is described in detail by specialising this model. In the same way a generic model is provided describing the event and again the subdetector event structure is produced by specialising this generic model.

All subdetector groups are already active in producing reconstruction software. There are typically 2-3 physicists in each subdetector group already working on software for describing their detector, describing the event model and for implementing the pattern recognition algorithms. We have recently formalised this activity as a new project, BRUNEL. The framework of the reconstruction program is based on GAUDI and this will be complemented by pieces to make a real reconstruction program e.g. control the sequencing of algorithms etc. We have already had reviews of the designs of the software for the Tracking, RICH and Calorimetry software. These reviews took place in March 2000. These reviews identified a number of issues which are common to all detectors (http://lhcb.cern.ch/computing/SoftwareWeeks/Apr2000/SWApr2000.htm). At the last software workshop these issues were reviewed and guidelines are being prepared for managing them in common. In this way we aim to avoid unnecessary duplication of ways of handling standard problems which should ease the understandability and maintainability of the software. We intend to hold reviews on a regular basis as this is a very effective way of ensuring a coherent approach is followed to developing our software. This has the full support and active participation of all the physicists involved in software development.

2.7 Is your architecture flexible enough to allow the co-existence of “action on demand” and “explicit invocation” in the same application?

The primary way of scheduling *Algorithms* within the GAUDI architecture is by “explicit invocation”. We are assuming that physicists know what steps are need to be performed in order to obtain the desired result. This knowledge can be minimized by the fact that we allow *Algorithms* to be organized in a hierarchical way. In that way, each “parent” algorithm needs to know only the sequence and details of its sub-algorithms only.

Having said that, it is absolutely allowed, to enhance the *EventDataService* such that it behaves as “action on demand”. In fact it is planned to implement this in the near future. The principle is very simple and identical to the “load on demand” mechanism that is already implemented. The service needs to be instructed that in case a piece of data is not found in the transient data store it needs to schedule for execution a named *Algorithm* instead of trying to fetch the data from the persistent store. In that way, Algorithms requesting a piece of data may trigger the corresponding action for creating the missing data.

2.8 Does your persistency solution restrict you on what classes you might use to implement a given physics module?

Already in the analysis and design phase of GAUDI emphasis was put on the fact that a separation between the transient representation and the persistent representation of an object allows to optimize each representation according to its needs:

- Persistent data are usually optimised in terms of their storage allocation. This typically involves the use of data compression, minimisation of links between objects, clustering small objects together, avoiding data duplication that may cause inconsistencies, etc.
- Transient data is organised to be optimum for the execution of algorithms. For example by duplicating information if needed (caching). This caching mechanism optimize the execution physics algorithms that access this data many times.

We thoroughly kept to this decision in order to explicitly avoid any restrictions on a given physics module arising from the persistent technology. An *Algorithm* sees only the transient representation of the event data. To implement relationships between objects we use smart pointers that are resolved to raw C++ pointers when needed. These smart pointers are “generic”, meaning that the underlying load mechanism is an integral part of GAUDI only knowing about GAUDI interfaces and is not connected to the persistency technology. The load mechanism dispatches a load-on-demand request to the persistency service which is only accessed through an interface. The underlying persistent access could be implemented using any suitable storage technology. In conclusion, we do not see any technical real reason why the chosen persistency solution should restrict the design/implementation of physics modules (*Algorithms*).

2.9 Within your architecture, are you able to migrate a physics module from the reconstruction environment to the trigger environment (event filter)? Is the event data presented to the event filter algorithm in the same format?

This is exactly one of the main use-cases we did identify during the analysis and design phase of the GAUDI architecture. This use-case influenced very strongly our choice for the separation between the transient and persistent representation of the data objects. We expect to have no problem (guaranteed by design) when moving a physics *Algorithm* from the reconstruction environment to the trigger or on-line environment. Event data will be presented to *Algorithm*s in exactly the same format. Moreover, due to the fact that Algorithms use *Services* only through abstract interfaces, would allow us to replace the implementation for some of the basic services by others better suited for the on-line environment. For example to be able to exercise remote control and monitoring, multi-threaded support, etc.

2.10 Would you be ready to support other experiments using your tools? Would the experiments be able to come with new requirements? Do you think it is possible to have a common tool supported by CERN/IT?

At present we use CMT, a software tool written by C.Arnault (Orsay/ATLAS), for managing together with CVS our configuration management procedures. Our collaboration with Christian has been excellent. He has taken account of our comments and adapted improved the product in response to our requests in a remarkably quick and professional manner. We understand that Orsay intend to guarantee the long term maintenance of CMT in a long term manner. In these circumstances we see no reason not to make use of tools developed and maintained in the context of another experiment.

On the other hand the majority of tools used for managing the various software engineering tasks are commercially produced and maintained. The in-house support of these commercial tools is very time consuming. It involves technology tracking, dealing with companies, handling licences, porting to supported platforms, making new releases.This is clearly something where benfits can be obtained from economy of scale. We believe that CERN should have an IT policy with a recomended set of tools and that IT division should continue to run a tool support service.

We believe that our requirements on software tools would most likely be very similar to those of the other experiments and do not see why common solutions cannot be found. If a tool is reasonably cheap, easy to use and effective then we would be happy to use it. At present we use simple cheap and easy to use tools, usually running on NT where the market appears to be very large.

2.11 Each experiment reported use of FLUKA to a greater or lesser extent. Please formulate a statement expressing the experiment request for support of FLUKA, involving IT and possible others, and its relative priority in the spectrum of products needing support.

We intend to use FLUKA as an important cross-check of particle densities in the LHCb detector, in particular for the low energy backgrounds. This is crucial for the performance of the LHCb detector. FLUKA should be taken very seriously. We have been using up to now a couple of completely independent packages capable of similar physics: MARS and GCALOR. We have some doubts on the results from MARS and we will feel more comfortable if we could make an independent check with FLUKA.

One of the difficutities in using FLUKA is that the geometry description of the detector is independent of the GEANT description. Consequently we would need to describe the detector again, which is a considerable amount of work. A centralized geometry description to work with FLUKA and GEANT4 is strongly requested by us, and we imagine would be beneficial for all other LHC experiments. We would give the support for FLUKA high priority. We cannot be more precise without knowing to what other products needing manpower for support we should compare.

2.12 The use of ROOT as an analysis package is widespread in the HEP community and within many of the LHC experiments. Some experiments made it clear that they considered ROOT as a product that “would be there” as a backup, or in case they should choose to use it. Please formulate a statement expressing the experiment’s request for support of ROOT, involving IT and possibly others, and its relative priority in the spectrum of products needing support.

We currently rely on ROOT for two test beams out of the 5 test beams activities in LHCb. We also rely on ROOT as an I/O package within the GAUDI framework. The level of support of ROOT has been very good up to now and we require that this support continues for the foreseeable future. The authors of ROOT are also aware that the level of support could even be improved, for example with additions to the documentation such as a user’s manual.

As far as we are aware, ROOT is the only existing object oriented analysis tool (i.e. PAW-like functionality) in widespread use in HEP today and therefore it should be maintained.

What would be the effect on the experiment if ROOT were to become unsupported, or become available to “ROOT collaboration members only”?

One of the advantages of using ROOT has been its good support. To continue the use of ROOT in the test beam activities relies on the fact that current level of support is continued.

Does the experiment foresee any future development path for ROOT which could provide a bridge from what exists now to what they need in an analysis package, perhaps making it one useful tool among several available?

We are ready to participate in any discussion concerning the future developments of ROOT to take into account the particular needs of LHCb. Our situation would be clearly simplified if there were one common strategy in which the role of ROOT amongst other complementary tools was defined and agreed.

3 Questions/Requests for LHCb

3.1 Has LHCb had any external review of its Architecture and Framework to establish the cost and risks of using such a general and open approach?

In November 1998 the LHCb architecture was reviewed by a group of internal and external reviewers. Details of the architecture review can be found in (http://lhcb.cern.ch/computing/Steering/Reviews). The goal of the review was to find out if they were major mistakes in the ideas and choices that were made. It was felt important to confront the new design to a group of experts in the field of object-oriented technology before embarking on the development of the framework. The basic question to be answered by the reviewers was: would a framework with that design work? The conclusion was to go ahead with the first implementation of the framework to test some of the ideas and eventually iterate.

Costs and risks of the approach were not reviewed. We did schedule another architecture/framework external review by spring 2000, but this has been delayed due to other reviews in progress at the same time. Meanwhile, the fact that ATLAS is using the LHCb framework for their first prototype is like being thoroughly reviewed. In any case, the forthcoming review of the framework should certainly identify the risks and the cost of the approach taken by LHCb.

Are there any estimates of the performance costs, development time costs, and potential debugging costs associated with the approach?

We have measured the overhead associated to the persistency mechanism based on converters. This overhead comes from the fact of having to search into the transient data store, copy data from the persistent representation to the transient and resolving the relationships. This overhead has been estimated to 20% for an application with ‘empty’ algorithms in comparison with the same application using the native persistency mechanism accessing to the same data. This figure goes down when the algorithms have some ‘meat’. We therefore conclude that the performance penalty resulting from the use of converters is completely negligible.

We do not know how to measure development time costs and debugging costs. It is very difficult to disentangle the many factors involved: quality of the people, level of experience, training, quality of the documentation, quality of the software development tools, etc.

3.2 If LHCb is really prepared to ’use what is provided’ as a common solution for data persistency (and indeed a data management system) how will you be assured that it will meet your needs?

The LHCb requirements for data persistency in terms of scale are less stringent that the other LHC experiments. The requirements for the integration into the experiment framework and the development effort are equally stringent and perhaps with some degree of orthogonality (NT platform, separtion of transient and persistent, etc.). As mentioned already in questions 2.2 and 2.5, LHCb should actively participate in the definition of the requirements by providing use cases, eventually in the design of the architecture and in the evaluation and testing of the provided solutions. It should also participate in the decision making process.

3.3 Given the architectural choices of LHCb what constraints will be put on a decision to use Java?

The architectural choices in LHCb have bean made with Java in mind. Therefore we do not expect to put any constraint on a possible decision to use Java. On the contrary, we believe that we are facilitating a possible migration if decided. The decision making process for this migration has not been established.

Will the experiment establish some rules or guidance for language choices, persistency choices and scripting language choices?

Of course. The fact that we have designed an open and flexible architecture does not mean that every member of the collaboration is encourage to use a different language, a different persistency technology, a different scripting language, etc. This will obviously result in chaos. The openness and the flexibility of the architecture facilitates the evolution to other languages and persistency technologies. But this evolution and changes must be discussed in the collaboration. This kind of discussion is foreseen to take place in yearly computing reviews. Stable rules and guidelines will be established for sizable periods of time (1 - 2 years at least).

3.4 Do you believe that your choice(s) for data persistency will place constraints on your analysis tools and if so what will that mean for the experiment?

We believe that the possibility of adopting different data persistencies as well as the separations between persistent and transient data we have in our architecture will minimize the impact of choice(s) for data persistency on our analysis tools.

We are open in fact to the possibility that the persistency choice for user physics analysis data could be different from that of reconstruction data if the access to the data and utilization of analysis tools will require it. It is naturally desirable to minimize the number of persistency choices as well of analysis tools for the different processing stages.

3.5 How do you foresee to maintain consistency across persistent objects stored in different technologies?

We have never said that it is our desire to store objects from a processing step using various technologies simultaneously. The fact, that this is possible is a consequence of the open architecture of GAUDI. A possible use of this freedom could arise from the creation of datasets private to a physicist derived from the collaboration event database. Clearly such datasets do not necessarily have to be registered with a collaboration wide federation.

The question can be reformulated:

- How do we ensure that references to object from previous processing steps are valid?
- How is the access to the raw data ensured from the reconstructed event, the access to the reconstructed event from the analysis objects etc.?

Event data are WORM (write once, read many). In case a reprocessing step is necessary a new dataset will be created maintaining all necessary references to objects used to create this dataset. A “data manager” at some point will make this dataset the default and eventually discard previously created datasets once these are proven to be obsolete. Written data are read-only and will not be updated. This mechanism ensures that references are valid from wherever a physics algorithm accesses data created by a step of the processing chain.

Forward references, such as references from raw data to reconstructed data, are not allowed. It is not desired to update all raw data objects whenever a new reprocessing becomes the default: raw data stay as they should, read-only. This ensures the same access to the raw data for any number of simultaneous existing data from reprocessing steps. The alternative would always favour the “default” reprocessing. When testing reconstruction code such a behaviour is even undesired: the forward link from the raw data to the reconstructed data would point to the default reprocessed data instead to the created data.

3.6 How do you plan to support schema evolution and maintain the converters used?

In order to answer this question, our approach to data persistency must be briefly explained. The GAUDI persistency mechanism is based on converters. Converters can be written specifically “by-hand” for each class that we wish to store and retrieve or can also be “generic” based on data serialization mechanism as it is done in Java, MFC and ROOT. The first case is used when we have no influence on the actual format of the data in the persistency storage (i.e. in case of legacy data). In the second case, the converter uses the serializer method on the object being converted to perform the specific work. This approach creates a flat machine-independent byte stream from the object’s data. This byte stream can then be stored in any suited storage technology allowing for arbitrary size binary objects. Data serializers have to be written by hand; however they are trivial, typically a few lines per class. Support is given for all primitive data types (int, float, char, double,...) and object references which are encapsulated in smart pointers. All persistent-capable base classes support object versioning. It is obvious that the order member data are streamed plays a role and cannot be changed arbitrarily. Since data are WORM, it is not required to update neither object data nor object links.

Schema evolution:
Simple schema changes can easily be implemented by customizing the serialization dependent in the object’s version. The code steers the interpretation of the flat byte stream into useful data. Data members no longer present in the implementation must be serialized to dummy automatic variables. Data members not present in the persistent representation acquire a default value. If possible, one could also imagine to re-compute a meaningful value of the missing data member using data present in the object. If the inheritance structure changed, the serializers of the base and sub-classes must ensure the correct order of the data members.

If the new behaviour is an extension to the existing behaviour and requires a major schema change, a new class can be created which implements all extensions. This changes the persistent type identifier and hence another converter will be responsible for the conversion. A customized converter must be written (and maintained) which supplies the necessary values. Since the customized converter is running within the GAUDI framework it has a better chance to supply meaningful data than simple modules as they are supported by Objectivity. Existing client code would not be affected because it would only use the existing transient representation, whereas new code would benefit from the extensions. This allows a smooth transition until all affected client code is adapted to use the new transient representation.

Converter maintenance:
As mentioned above most of the work is done inside data serializers. Real maintenance load only arises in the (hopefully) rare case of rigid schema evolution.