LHCb

# GAUDI

*LHCb Data Processing Applications Framework*

## Scenarios and Requirements

|  |  |
|---|---|
| Issue: | 1 |
| Revision: | 1 |
| Reference: | LHCb 98-065 COMP |
| Created: | 20 October 1998 |
| Last modified: | 20 November1998 |

|  |  |
|---|---|
| **Prepared By:** | LHCb software architecture group<br>Editor: Paul Maley |

# Abstract

This document collects together requirements and example scenarios / use-cases for the "Offline" framework project. The scenarios serve for the extraction of further requirements and for the testing of the functionality and robustness of the system architecture.

This document will be under constant revision for the forseeable future.

# Document Status Sheet

**Table 1** Document Status Sheet

| **1. Document Title: LHCb Data Processing Applications Framework** | | | |
|---|---|---|---|
| **2. Document Reference Number: LHCb 98-065 COMP** | | | |
| **3. Issue** | **4. Revision** | **5. Date** | **6. Reason for change** |
| 1 | 1 | 20 November 98 | Architecture review |

# Table of Contents

# 1 Introduction

This is a relatively informal document whose aim is to capture the essential requirements of the LHCb computing framework (mainly in the offline environment for the moment). Throughout the document we have placed the emphasis on the capturing of an idea rather than on its formal expression.

To a certain extent this document has evolved in parallel with the design of the architecture and it will almost certainly be updated as our understanding of the system and the system users improves.

The document consists of three sections: Scenarios, Requirements and Desirables.

We use the term scenario quite loosely and synonymously with the term use-case. We have used scenarios as a way to deduce the system requirements. We require scenarios and requirements from throughout the life-cycle of the project.

In the course of our discussions with potential system users we came across things which “it would be nice to have”, but were not really formulated either in terms of scenarios nor as requirements, we have grouped these together under the term: Desirables.

The language used in the document reflects the two types of individual who have contributed. On the one hand people directly involved in the design of the system architecture have necessarily dreamt up example scenarios. These are generally phrased in terms of the components which we think to use. The other source of input comes directly from physicists, who have no pre-conceived ideas on the architecture and frame their example use-cases and requirements in a more physics-based vocabulary. The mapping of this second class of scenarios and requirements is much more demanding (and useful) and we have made no attempt to convert these into framework based language.

We have grouped the scenarios according to the type of individual who is most affected by or interested by that particular usage. Ofcourse, often several different types of user are involved. In places we refer to “users”. Usually this term refers to someone who does not write code, but only “uses” the software. In our environment such people do not exist since everybody writes code for one reason or another. We have identified the main groups of people who will work with the framework, as follows:

1. Physicist users.
2. Physicist developers.
3. Data production managers.
4. Framework developers.

These roles are explained in the Architecture Design Document [1], and ofcourse, the same people may perform different roles at different times.

# 2 Scenarios

## 2.1 Physicist-Users

### 2.1.1 Job Configuration

**S-PU-1.**An analysis job runs several algorithms to select events and produces a file of histograms.

**S-PU-2.**A detector calibration job is run. Several algorithms are called once per event to collect statistics. After a certain number of events have been analysed an algorithm is run to analyse these data and produce a new set of calibration constants.

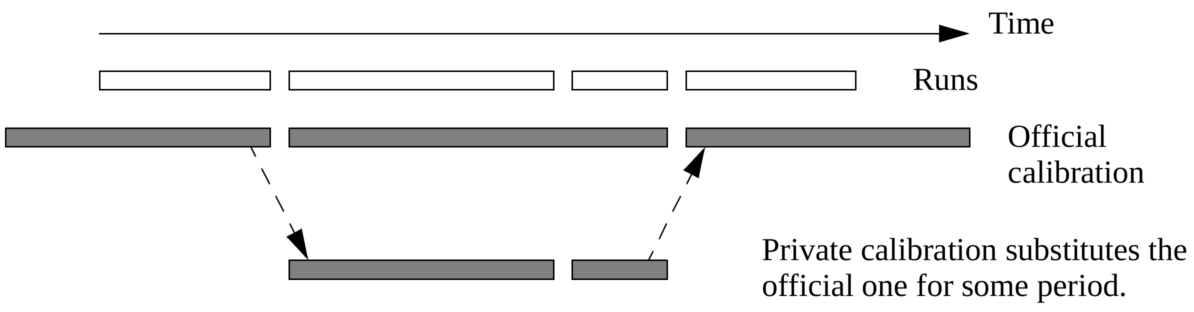
**S-PU-3.**An algorithm wishes to make histograms on a per-run basis and analyse several runs within a single job. The histograms are to be stored.

**S-PU-4.**A physicist wishes to simulate only the vertex detector and reconstruct vertex detector raw data. All other subdetectors are ignored.

**S-PU-5.**A physicist wishes to reconstruct only muon detector hits, however, the simulation must be run including all sub detectors.

**S-PU-6.**A physicist runs a set of algorithms in order to select events. At the end of the job this new event selection is stored. It may be used by other applications at a later date.

**S-PU-7.**A physicist wishes to replace some of the standard calibration constants with a private set.


Time
Runs
Official calibration
Private calibration substitutes the official one for some period.

**S-PU-8.**An algorithm is used both offline and as a monitoring process online. In the online environment it should send an alarm to the DCS whereas offline it suffices to write a warning in a log file.

**S-PU-9.**Two independent detector alignments are available. A physicist wishes to compare them by measuring the difference in the fitted momentum of individual tracks.

### 2.1.2 Algorithm Configuration

**S-PU-10.**A track-fit algorithm is executed on a set of hits and a track produced. The algorithm is then re-run with parameters modified according to the momentum of the fitted track and a new track produced.

**S-PU-11.**An algorithm is run twice, with different parameter settings, the results of the two algorithms are compared on an event by event basis.

**S-PU-12.**A track candidate is extrapolated to the next tracking station and two possible hits are found. It may be necessary to extrapolate both track candidates to the following station.

**S-PU-13.**Noisy tracking planes may be excluded from the track-finding procedure.

**S-PU-14.**The track fitting algorithm is set not to use a particular tracking plane, instead the hits from this plane are compared with the fitted tracks in order to calculate residuals, efficiencies etc.

**S-PU-15.**A fitted track which is identified as an electron by the RICH is refitted, taking into account its energy loss.

**S-PU-16.**It is required to apply some correction to reconstructed tracks after they are input from storage and before they are used in an analysis. Different corrections might be applied to the same track in order to compare the results.

**S-PU-17.**To perform a constrained fit, a fast algorithm is first tried; if this fails to converge a more robust (but slower) algorithm is run instead.

**S-PU-18.**An algorithm wishes to use another algorithm (nested) many times per event.

### 2.1.3 User interactivity

**S-PU-19.**An event is rejected by an analysis program. The physicist wishes to know at what point of the program it was rejected.

**S-PU-20.**A physicist runs the framework interactively. He/she wishes to do the following:

1. Load the next event.
2. Analyse the following n events.
3. Set a “break point” in the analysis to allow a particular event (type) to be examined.
4. Browse the event/detector data.
5. Alter the level of verboseness of an algorithm.
6. Re-run specific algorithms on specified data.
7. Change algorithm parameters.

**S-PU-21.**A track is displayed. A physicist wishes to select a subset of the track’s hits (by clicking on the event display) and refit the track using only those hits. The new track is to be displayed along with the original.

**S-PU-22.**A physicist would like to visualize the effect of imposing a vertex constraint when fitting tracks. Both sets of tracks (with and without constraint) should be visible at once.

### 2.1.4 Use of data

**S-PU-23.**An analysis selects a set of tracks as the decay products of a candidate neutral B. A reconstructed momentum vector, plus the selection of daughter tracks must be stored.

**S-PU-24.**An algorithm creates a new directory in the event data store and adds objects to it. These objects are to be saved.

**S-PU-25.**It is desired to “mark” raw data hits so that they are used by an algorithm once only.

**S-PU-26.**The average vertex position is calculated for groups of several hundred events. This information is required by some algorithms.

**S-PU-27.**A physicist wishes to study the correlation between some “slow control” data, e.g. a temperature, and a quantity derived from the event data.

**S-PU-28.**A histogram of some monitored machine parameter verses time is required.

## 2.2 Physicist-Developers

**S-PD-1.**There is a central database of generated data. Two independent reconstruction developers read this data and generate their own user types. Both wish to save their objects along with references to the objects in the original database.

**S-PD-2.**A reconstruction developer has event data in a Zebra store and detector description data in objectivity.

**S-PD-3.**An algorithm originally developed for offline use is incorporated into the level 3 trigger.

**S-PD-4.**A user invents a new data type (e.g. a kinked-track) and wishes to store it to disk.

**S-PD-5.**A physicist wishes to define a new graphical representation of a reconstructed object.

**S-PD-6.**A new physics generator is developed. A physicist wishes to integrate it into the framework so as to generate events to analyse.

**S-PD-7.**A new sub detector is added to the experiment, software must be developed for analysing it's data.

**S-PD-8.**It is desired to use the framework to analyse test-beam data. Reconstructed objects must be stored, the data model will change as the understanding of the data and the necessary analysis improves.

**S-PD-9.**The definition of an object must be changed. The developer would like to be able to read data containing old versions of the object and data containing new versions of the object together.

**S-PD-10.**In order to optimize the geometry of a sub detector a physicist wishes to run the event simulation on two different geometries and compare the results. For some aspects an event by event comparison would be useful.

## 2.3 Data Production Managers

**S-DPM-1.**The detector configuration manager wishes to label a particular version of the detector geometry.

**S-DPM-2.**All of the data taken in one year must be reprocessed. Alignment parameters have changed and some reconstruction algorithms have also been modified.

## 2.4 Framework Developers.

### 2.4.1 Software changes

**S-FD-1.**A new 3D graphics library becomes available and will be used to replace the currently used library.

**S-FD-2.**It is necessary to replace the database system used for event storage.

**S-FD-3.**We decide to change programming language, e.g. to Java.

### 2.4.2 Environment changes

**S-FD-4.**The OS or windowing environment of a supported platform changes.

**S-FD-5.**An analysis job may be run interactively for debugging, or in batch mode for event crunching.

**S-FD-6.**The framework is used to build a “real-time” monitoring system. It receives events, analyses them to produce distributions and displays the distributions in real-time. The application can raise an alarm in the case of an error being detected.

**S-FD-7.**An application may be distributed over several physical processors communicating via a network.

## 2.5 System Maintainers

To be added: scenarios dealing with the long term maintenance of the code:

- Testing of new releases
- Porting to new platforms
- Code management, versioning, bug fixing

# 3 Requirements

We make no attempt to be exhaustive, relying instead on an “on-the-fly” analysis of the scenarios.

## 3.1 I/O

**UR-1.**It must be possible to define new object types and, if desired, store these along with references to existing objects etc.[S-PU-23.]

**UR-2.**It must be possible to specify input and output data sets.[S-PU-7.]

**UR-3.**The framework must shield algorithms (reconstruction, simulation etc.) from the details of particular storage (database of otherwise) technologies.[S-PD-2.]

**UR-4.**Detector description constants (alignment, calibration, etc.) presented to the algorithms must be kept in sync with the event data automatically by the framework.[S-PU-7.]

## 3.2 Configuration

**UR-5.**A user must be able to override all default parameters (for algorithms etc.). It should be possible to store these user-defined settings and re-use them.[S-PU-11.]

**UR-6.**The simulation must be capable of simulating detector defects and non-functioning channels and electronic noise.

**UR-7.**The description of the detector geometry must exist in multiple versions, to allow for the moving of detector elements in between different running periods.

**UR-8.**It must be possible to run each “stage” (event generation, reconstruction, analysis) either independently (i.e. in different programs) or all together in a single program.

**UR-9.**The framework must support a high degree of configurability.[S-PU-4.,S-PU-5.]

## 3.3 Transient data

**UR-10.**An application must be able to create objects with different lifetimes, e.g. objects that exist only for a single event, those that exist for a run, or a job.[S-PU-2.,S-PU-3.]

**UR-11.**It must be possible to base applications around data other than raw event data, e.g. slow control data.[S-PU-27.,S-PU-28.]

## 3.4 Usability

**UR-12.**The framework must be adequately documented.

**UR-13.**Individual developers must be able to develop independent software packages in parallel without mutual interference.[S-PD-1.,S-PD-8.]

## 3.5 Modifiability

**UR-14.**The system must facilitate the use of new software components as they become available.[S-PD-6.,S-FD-1.,S-FD-2.]

## 3.6 Performance

**UR-15.**Central data stores must support concurrent access by many users [Quantify].

**UR-16.**[The framework must not be the limiting factor in the execution speed of a reasonable application - Quantify]

# 4 Desirables

This is a list of things which it would be nice to have. They are not formulated in a very precise manner and thus are not classed as requirements per se. In time, with clarification, they could be promoted to requirements.

1. Calculation of systematics is often done by running a job several times with different parameter settings. It would be nice to be able to do this in a single pass.
2. A large fraction of analysis code is just for book-keeping and calculating statistics - Much of this could be either automated or made much easier by the provision of a set of utility classes.
3. An analysis which implements a series of event cuts would be made simpler by automating the counting of how many events passed / failed the cuts. The final calculation of efficiencies and the printing of the results could also be automated.
4. A histogram display which updates in real time.
5. It would be nice to have a “geometry editor”, e.g:
   - Take a pre-defined geometry (pad / strip layout)
   - Edit it efficiently (i.e. edit something simple and replicate it)
   - Visualise it to make sure that it’s correct.
   - Save it back into the database.
6. When saving a set of histograms, save also a “header” containing the algorithm versions used, generator settings, etc.
7. An algorithm which takes generator level data and converts it to data usable directly by an analysis program without passing through the simulation and reconstruction phases (if possible !) might be useful.
8. The detector geometry is described by essentially:
   - An ideal geometry - i.e. where detector elements would be placed in the absence of measurement (and human) imperfections.
   - Perturbations around this geometry. With the possibility to have different geometries for simulation and reconstruction.

## 5 References

**[1]** GAUDI: Architecture Design Document, P. Mato et al., LHCb 98-064 COMP