# LHC++ - "CERNLIB" for LHC?

Jamie Shiers

*CERN, Geneva, Switzerland*

In response to the stated requirements of the LHC collaborations, a working group was set up in 1995 to investigate how the approximate equivalent of the current CERNLIB environment could be provided in the LHC era. As the name of the working group suggests, the focus has been primarily on C++ based solutions, although recently this has expanded to include Java. The working group has placed strong emphasis on the use of standard solutions where-ever possible, be they *de-facto* or *de-jure*. In addition, existing work, such as the CLHEP class library, has been viewed as an important component of the overall strategy. As a result of the emphasis placed on standards, HEP effort is concentrated on the HEP-specific part of the problem domain. We describe the overall LHC++ architecture, the implementation status and future plans. We also discuss a possible plan for handling the licenses for the commercial components of the proposed solution.

## 1 Introduction

The current CERN Program Library, which has been in use by numerous experiments around the world for many years, is largely Fortran based. Much of it was developed for very different computing environments to those that are in use today, and it was designed to satisfy a totally different set of requirements than those of the LHC experiments.

For some time now, the stated direction of the ATLAS and CMS collaborations has been towards object-oriented solutions, and the use of the C++ language in particular. Thus, there was a clear need to understand how something roughly equivalent to the current CERN library - but not a blind rewrite or conversion - could be provided for LHC-era experiments. Indeed, the ATLAS Technical Proposal explicitly mentioned this as a requirement.

In mid-1995, a small working group was set up to address these issues. This working group has expanded considerably since, and now holds regular, open,

*Preprint submitted to Elsevier Preprint* *21 May 1997*

meetings and workshops.

In addition to addressing the requirements of the LHC experiments during the production phase, there was a clear need to satisfy the needs of pre-LHC experiments, such as NA45 at CERN, as well as the prototyping and research activities being conducted and other experiments world-wide.

¿From the start, the goal has been to reuse existing work - such as the Class Library for High Energy Physics (CLHEP), that was initiated at a previous conference in this series - CHEP 92 in Annecy, France.

As well as building upon existing work, there was a strong desire to use standard solutions where-ever possible - be they de-facto or de-jure. Examples of such standards range from the standard C++ library to OpenInventor to the use of an ODMG-compliant ODBMS.

Over the past 18 months, an interim strategy has been defined, in collaboration with experiments at CERN and outside, and software licenses acquired, initially for the CERN site.

More details on the current status of the implementation, including plans to extend the current license agreements to institutes outside CERN, are given below.

## 2 Why LHC++

As mentioned above, the current CERN Library was not designed to satisfy the stringent requirements of the LHC experiments - already today, experiments are encountering its limitations. Much of it was designed during the mainframe era, and it fits poorly with today's distributed computing environment. Finally, we are already facing maintenance problems, as many of the original authors have left, and this will become increasingly severe in the future.

As a bottom-line, something significantly more powerful is required for the LHC era, it must be maintainable within the projected man-power envelope, any commercial components must be affordable, and at least parts of it are required today.

## 3 Why use standards?

There are clear benefits from the adoption of standard solutions - indeed, this has been a common policy within HEP for many years (Fortran, GKS, PHIGS etc.) Of particular interest are those standards that are widely adopted. This has a number of clear benefits:

(i) We can benefit from the broad user base, which corresponds (hopefully) to good support, documentation, and a robust, well-debugged product,

(ii) The development costs have been amortized, translating into an affordable product. In some cases, e.g. OpenGL on NT, the software is even free!

(iii) A migration path to follow-on products can be expected. It is clear that standards will continue to evolve, and those libraries that we chose today may be superceded during the lifetime of the LHC. We must plan for this change, including a smooth migration path to replacement products. An example of such an evolution is OpenInventor. A follow-on product is already planned, although this will support the existing interface for at least three years. However, we need to plan for this migration, and maybe even a subsequent migration, *prior* to the start of LHC data taking!

## 4 Why use commerical products?

Similar arguments can be given for the use of commercial products. Again, this is not new to HEP, examples of commercial products that have been used in the past include GKS and GPHIGS for graphics, and HISTORIAN and CMZ for code management. Industry-standard, commercial products, where applicable, are arguably both cheaper *and* better than those developed inhouse.

As an example, a state-of-the-art 3D graphics library can be acquired for the cost of a technical student for one year. A site-license for a state-of-the-art mathematical library in turn costs only 1.5 associate months. Finally, a highly-functional ODBMS weights in are around 9 technical student month-equivalents per LHC experiment per year.

In addition, one can expect professional support and documentation, examples of which include the excellent OpenGL and OpenInventor books.

As a side-effect of the use of standard, commercial solutions, the effort that would otherwise be required to develop, document and maintain equivalent software can be devoted to HEP-specific problems - precisely those areas where a commercial solution cannot reasonably be obtained, e.g. GEANT-4, CLHEP, HEP analysis classes etc..

## 5 Current LHC++ strategy

The current LHC++ strategy is basically composed of two lines:

(i) Standard solutions, including OpenGL, OpenInventor, visualisation tools, mathematical libraries and so forth,
(ii) physics applications or components, such as CLHEP, detector simulation and event generators, random number generators, CLHEP, ODBMS-helper classes etc..

More details on each of these areas are given below.

## 6 Initial steps

Basically, the initial steps performed by the working group were:

- To understand our requirements in all areas covered by the current CERNLIB,
- To survey the market, be it commercial, HEP or free, to find out what was available in each of these areas, and what were the appropriate "standards" if any,
- To compare different implementations/solutions, such as alternative mathematical libraries, various container/collection classes etc.,
- To identify components that could be re-used, such as CLHEP, to clarify where HEP has specific requirements that cannot be met by existing solutions.

More details on these investigations, and indeed on LHC++ in general, can be found through the Web via the LHC++ pages, accessible from the "Computing" link on the CERN home page.

## 7 HEP-specific areas

Below we describe on-going work in a number of areas for which HEP-specific solutions were felt necessary. This should not be taken as an exhaustive list of HEP requirements, but more as an indication of where initial efforts have been concentrated.

### *7.1 Histogram working group*

A working group on OO histograms was initially established in CN/ASD group simply to exercise the skills gained by following the OO "thread" of the CERN training program. With the advent of the LHC++ working group, a clear need for histogram classes was identified, following an evaluation of available implementations, including those provided in Rogue Wave's math.h++ class library. Having failed to identify satisfactory histogram classes, this subgroup initially defined the requirements, and subsequently developed an initial design and prototype implementation. It is important to stress that these classes do not attempt to blindly reimplement the functionality provided by HBOOK, although convenience classes, with the well-known HBOOK1 and HBOOK2 subroutines, are provided. In addition, the classes are de-coupled from I/O and display issues, the former being provided through the mechanisms being investigated in the CERN RD45 project, such as an ODBMS, and the latter through tools such as IRIS Explorer.

At the time of writing, these classes are being tested by a number of users, although further development is clearly required.

### *7.2 Random number generators*

The current CERNLIB includes several state-of-the-art random number generators, developed to meet HEP's particular requirements. These have been made available through CLHEP for sometime, via wrapper classes. A comparison of these classes with those provided by Rogue Wave revealed advantages in both approaches. As part of the GEANT-4 collaboration, a new set of classes were developed, providing great flexibility in terms of random number engine or distribution, incorporating and improving on the strong points of the existing CLHEP and Rogue Wave implementations.

### *7.3 Fitting*

The de-facto standard fitting package used within the HEP community has been MINUIT. However, there are good reasons to look for a replacement for the LHC era, including:

- The current package has long term maintenance problems,
- Better algorithms are now available,
- MINUIT itself was developed very much for a batch environment, whereas today's requirements are largely, but not exclusively, oriented towards interactive fitting.

Encapsulations of MINUIT in both C and C++ already exist, and offer perfectly valid short/medium-term solutions. Recently, we were able to identify a potential long-term replacement, namely those solutions being developed in the STABLE project, of which the Numerical Algorithms Group (NAG) is a member. (NAG is the current supplier to CERN of many of the commercial components of LHC++, including their traditional Fortran and C numerical libraries).

More details can be found through the LHC++ Web pages, or via the NAG pages directly.

### *7.4 Event generators*

Existing event generators, such as ARIADNE, HERWIG, ISAJET, JETSET, PYTHIA and others, are written exclusively in Fortran. Coupling to other packages, such as detector simulation tools, is typically performed using the HEPEVT Fortran common block. Ideally, for the LHC era, one would like updated generators, written in C++ with a clean coupling to e.g. GEANT-4. Indeed, a proposal to rewrite the Lund family of generators has been recently made by Leif Lonnblad and Torbjorn Sjostrand, whereby a initial release, with minimum physics, would be made in mid-1998, with a fully functional version approximately one year later. This is viewed by the working group as an extremely important activity.

### *7.5 GEANT-4*

Just as GEANT-3 has been distributed through the CERN Program Library for many years, it is planned that GEANT-4, once released, will be made available through the LHC++ distribution mechanism. In the interim, there is significant bi-directional collaboration between LHC++ and GEANT-4 - many of the elements of the LHC++ strategy are used within the GEANT-4 collaboration who, on the other hand, have many numerous contributions to CLHEP and other components of LHC++.

### 7.6 *HepInventor and HepExplorer*

HepInventor consists of a set of (proposed) extensions to OpenInventor, aimed at physics analysis. HepExplorer are IRIS Explorer modules, based largely on HepInventor. Examples include histogram display and operation modules, fitting modules and so on. Clearly, a number of these items are not strictly HEP specific, and we are working with NAG, the current maintainers of IRIS Explorer, to understand to what extent the required functionality can be made part of the standard product.

### 7.7 *HepODBMS*

HepODBMS is a set of helper-classes, written in the context of RD45, which:

- help to minimise dependence on a given ODBMS implementation (vendor or version),
- provide a high(er) level interface to the ODBMS,
- help to minimise the effort required to port existing applications that require persistence.

These classes, in early prototype form, are heavily used within the RD45 collaboration. It is planned to make them available, as a seperate component within the LHC++ framework, towards the end of 1997.

## 8 Current strategy

The current LHC++ strategy is based upon the following components. Although represented as a layer-model, this has merely been used for convenience in formatting.

- Physics applications, e.g. GEANT-4, event generators etc.,
- Objectivity/DB and HepODBMS,
- IRIS Explorer and HEP-specific modules,
- OpenGL, OpenInventor, Inventor Master and HepVIS extensions,
- NAG C library,
- CLHEP,
- LAPACK++/TNT,
- Tools.h++ and the standard C++ library.

## 9 Future activites

The main activity for the short term is to offer a realistic alternative to the current CERNLIB environment for experiments that have moved to C++. This includes the provision of the required HEP modules to IRIS Explorer, the necessary HEP-specific class libraries, such as CLHEP, HepODBMS etc., and also user manuals and tutorials. Pilot projects with OPAL and CMS have already been established using the new tools, and similar projects are anticipated with other experiments. A large amount of documentation is already available online, although this clearly needs supplementing with user-oriented tutorials and introductions.

## 10 Licensing issues

Key elements of the licensing strategy are the minisation of costs - not only to CERN, but also to collaborating institutes, and a flexible, manageable model with minimal overhead.

In general, it is proposed to acquire run-time licenses for all external institutes centrally, without passing on the cost of acquisition to the outside institutes. On the other hand, developer licenses would normally be paid for by the institutes directly, although again a centralised management model is proposed, to minimise both overhead and cost. That having been said, it is important to stress that developer licenses are *not* required at all institutes *nor* for all users. In addition, for two of the main components, namely Objectivity/DB and IRIS Explorer, full developer licenses will be made available to all collaborating users at no cost to external institutes. In summary, a contribution by the LHC experiments of 50KSF per year to a central fund is currently sufficient to acquire, in a staged manner, *all* of the run-time licenses for the commercial components for collaborating institutes.

## 11 Summary and Conclusions

By using standard solutions, we have managed to build up a powerful, open, coherent strategy in a very short time. Significantly, this has allowed us to address the requirements of the LHC experiments for a "CERNLIB-like" environment with *diminishing* manpower, and includes planning for the inevitable changes that will take place during the construction and production phases of the LHC. Every opportunity has been taken to discuss these ideas as widely as possible, and general consensus on the key elements of the strategy has been reached. The cost of the proposed solution compares extremely favourably to that of home-grown solutions, particularly when long-term maintenance issues are factored in.

The main components of the strategy have already been installed at CERN, user-guides are in preparation, and a model exists whereby licenses for external institutes can be handled in a centralised, manageable, and highly affordable way.