Not Invented Here:
The Re-use of Commercial Components in HEP Computing

J.P.R.B. Walton[1 2]

[1] The Numerical Algorithms Group Ltd, Wilkinson House, Jordan Hill Road, Oxford OX2 8DR, U.K.
[2] jeremyw@nag.co.uk

Abstract

We describe the way in which software components from a commercial supplier have been used within HEP computing libraries and applications. The components include routines from the NAG numerical library, the Open Inventor 3D graphics library and the IRIS Explorer visualization toolkit. They have been incorporated into the LHC++ project (a collection of libraries for future HEP computing) and have been used by experiments such as ATLAS and CMS. The re-use of standard, commercially supported components has allowed the HEP computing effort to concentrate on the implementation of HEP-specific functionality.

In addition, a degree of synergy has emerged which has seen some of these functions and requirements being incorporated into the commercial product, thus obviating the need for their future support by HEP computing. We will present a number of examples of this synergy including: the inclusion of the CERNLIB special functions in the NAG library, improved support for histogram display within IRIS Explorer, NAG's support of Minos-style error calculation and the increasing importance of Linux as a commercial platform.

keywords software re-use; components; libraries; commercial suppliers

1. Introduction

This paper describes the way in which recent High Energy Physics (HEP) computing libraries and applications have made use of software components from a commercial supplier. The incorporation of these standard components into their software has enabled the HEP developers to concentrate on the creation and addition of HEP-specific features on top of the generic functionality layer provided by the components. This has led to the more efficient use of HEP computing effort, since the requirement to (as in the past) implement the full functionality of each application has been obviated. It has also resulted in certain functionality, originally viewed as being HEP-specific, being incorporated into the commercial components when it was realized that it had a more general application. Thus, the relationship between the commercial supplier and the HEP community has led to benefits for both sides: the functionality of the commercial components are enhanced, while the HEP developers are free to concentrate on the further development of the features which are specific to their domain.

Our paper is arranged as follows. In the following section, we give a brief account of the history and development of LHC++, the HEP libraries (§2.1) and of NAG, the commercial supplier (§2.2). Section 3 contains some detail of the way in which the commercial products – including a numerical library (§3.1), a 3D graphics library (§3.2) and a visualization toolkit (§3.3) – have been incorporated into LHC++, and presents examples of their use. We then (§4) discuss some examples of the way in which HEP-specific features have made their way back into the commercial software, thus removing the requirement for future HEP support. We conclude our paper with (§5) some final remarks.

2. Background

2.1. The LHC++ Library

The strong tradition of internal software development at CERN, the European Organisation for Nuclear Research, is well-known and widely respected. Examples of important applications and libraries which have found extensive use within (and in some cases, outside of) the HEP community include CERNLIB[1] (a large collection of Fortran callable libraries) and PAW[2] (a interactive physics analysis tool), not to mention HTTP and HTML, which have become the *lingua franca* of the Internet.

However, coinciding with the development of CERN's newest experiment, the Large Hadron Collider (LHC), this policy of internal development was reviewed in the light of problems with support and diminishing technical resources[3]. Internal development has the advantage of promising complete control over the features and functionality of an application. However, problems of support and maintenance can arise if the developers' attention is diverted elsewhere owing to moves to other projects or locations. Similar problems can be caused by the retirement of the developer (although this last is perhaps difficult to contemplate, it should be noted that the long time-scales of current HEP experiments make them comparable with developer lifetimes).

A more fundamental reason for reviewing a policy of internal development is the notion that, in an environment with limited technical resource, it is important to ensure that developer time is allocated to areas where most value can be realized. Thus, for example, given the choice between – say – the implementation of a algorithm (which may come from a standard textbook, or from the research literature) for the inversion of a matrix, and the development of a method within a detector simulator such as GEANT4[4], it is clear that, while the former task would be within the ability of many developers, only one with HEP experience could effectively implement the latter. Put another way, it could perhaps be viewed as an inappropriate use of resource if the HEP developer found themselves working on the implementation of features which could be viewed as generic (i.e. not HEP-specific) and which could possibly be obtained from other sources.

LHC++[5] is a suite of tools for the storage, archiving and analysis of data from experiments in the LHC era. Following the considerations above, the strategy for the development of LHC++ include the use of commercial components wherever possible, with the HEP development time being devoted to the implementation of HEP-specific features. More details of the separation between these two strands are given in Section 3, below. In the following subsection however, we describe some of the commercial components used for the LHC++, and their supplier.

2.2. The Numerical Algorithms Group (NAG)

NAG[6] is a scientific software company based in Oxford, UK, with other offices in France, Germany, Japan and the US. In 1971 NAG developed the first version of a mathematical software library that now has over 10,000 users worldwide and contains over a thousand mathematical and statistical functions. The range of products and services that NAG offers has expanded into statistical, symbolic, visualization and numerical simulation software, together with compilers, application development tools and a consultancy service.

One of the features of NAG's history has been a strong degree of collaboration – both in formal research projects (such as those in the EU 4th Framework) and on a day-to-day basis with international experts in a variety of fields. Many of the routines in their numerical library, for example, have been developed and contributed by external authorities; NAG then further refines the code (in collaboration with the author), and checks it against their quality assurance scheme before adding it to the library.

3. Commercial tools

In this section we briefly describe some of the software components offered by NAG and the way in which they have been used inside LHC++.

3.1. The NAG C Library

NAG's numerical libraries collect together a range of reliable and robust numerical and statistical routines in areas such as optimization, PDEs, ODEs, FFTs, correlation and regression, and multivariate methods. The original implementation of the library was written in Fortran, while the C implementation[7] was originally developed in 1990 to satisfy user requirements for a high-quality numerical problem-solving library in C, following its emergence as a dominant programming language.

At an early stage of the LHC++ project, it was recognized that there was a strong degree of overlap between the functionality offered by the NAG C Library and the numerical requirements of HEP developers. A measure of the latter was provided by mathematical routines in CERNLIB that were almost completely duplicated in the NAG Library, with the exception of some of the special functions (see §4, below).

3.2. Open Inventor

Open Inventor[8] is an object-oriented graphics library that provides a comprehensive environment for creating a 3D scene and managing interactions with it. In Open Inventor, rendering of the scene is performed using the OpenGL graphics library, which has become a cross-platform standard for 3D graphics. It also defines a standard file format for storing 3D scenes, used as the basis of the so-called Virtual Reality Modeling Language (VRML) for the interchange of 3D scenes over the Web. Finally, the library can be extended to deal with new objects and methods, usually through standard C++ subclassing. For example, a supplemental library called MasterSuite[9], which incorporates objects and methods for high-level plotting and analysis has been built on top of Open Inventor in this way.

Most of the functionality represented by the GRAFLIB part of CERNLIB (in particular, the plotting of histograms) was determined to be covered by MasterSuite. In order to facilitate the use of these components, the LHC++ developers built another library, called HEPInventor[10], on top of MasterSuite and Open Inventor; this gave a simple object-oriented interface to tasks like creating axes, drawing the histogram, laying out the page and printing the resultant plot.

3.3. IRIS Explorer

IRIS Explorer[11] is a visualization toolkit with a visual programming interface. Working with IRIS Explorer, users interactively create their application by using a collection of software modules, each of which is represented on the screen as an object that can be manipulated via a

familiar point-and-click paradigm. The application is built by connecting module inputs and outputs together. The suite of modules which comes with IRIS Explorer offers a range of functionality for analyzing and displaying numerical data; moreover this suite is extensible, since users can add their own modules, either writing them from scratch, or incorporating legacy code.

Taking advantage of the visual programming paradigm offered by IRIS Explorer, LHC++ developers created HEP Explorer[12], a small set of HEP-specific modules, which are focussed on interactive analysis. The modules handle histograms (which are stored in an object database that is another component of LHC++) and allows the user to perform actions like creation, reading, cloning and plotting, as well as histogram arithmetic and fitting. A screen shot[13] from a HEP Explorer session is shown in Figure 1.

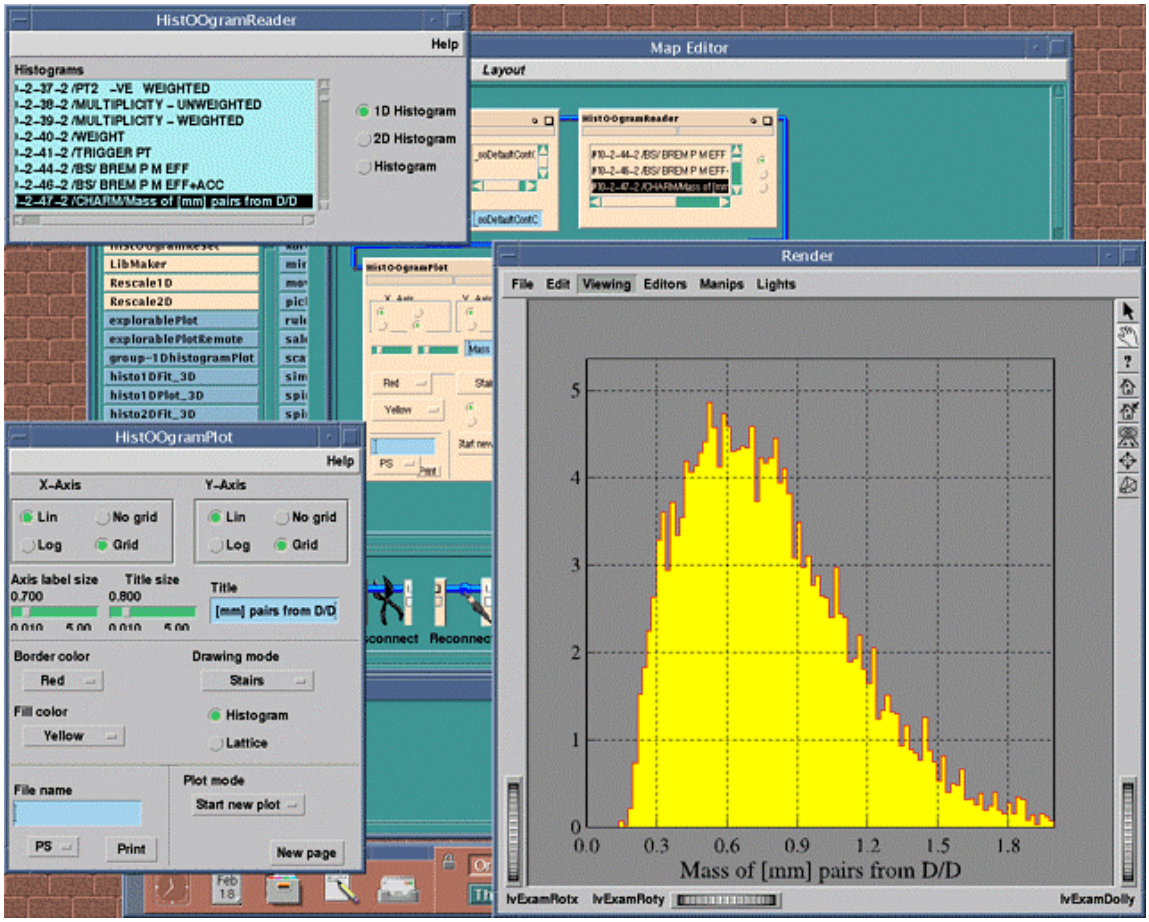

Figure 1. Using IRIS Explorer in LHC++ to analyze event data from one of the experiments at CERN. The first module in the map (partially obscured at top left in the map editor) is used to browse through collections of data (stored in a high-capacity database), while the second module reads specific events from the collection. The *HistOOgramPlot* module creates the plot of the data, which appears in the *Render* module window at bottom right. The *HistOOgramPlot* module also allows the user to write the plot out as a vector PostScript file, for printing or inclusion in other documents.

One of the features of IRIS Explorer as a comparatively high-level tool is that it is based on other components offered by NAG. For example, it uses routines from the NAG C Library (§3.1, above) in its modules for speed and accuracy[14], while its creation and manipulation of 3D visualizations is based on Open Inventor (§3.2), whose rendering is, in turn, done via OpenGL. This has been reflected in the way in which the HEP Explorer modules were built; they incorporate objects from the HEPInventor object library, along with an interface to the LHC++ object store, as previously noted.

4. Feedback from the HEP community

We have seen how the development of LHC++ has built upon, where possible, the use of standard commercially available components, with the extra pieces of HEP-specific functionality being added by HEP developers. Once the commercial components were in place, one of the objectives of the LHC++ project was to explore the possibilities of the commercial supplier taking over the support of some of the HEP-specific components. Increasing the proportion of commercially-supported components reduces the workload for the HEP developers, and frees them up to devote more resource to other areas. On the commercial side, such an arrangement would work best in cases where there is a strong possibility of (what were originally perceived as being) HEP-specific components finding application in other areas as well. Some examples of this synergy between the commercial and HEP sides are discussed in the following subsections.

4.1. Special functions in the NAG C Library

As noted above (§3.1) the NAG C Library has been judged to give a good coverage of the HEP requirements for mathematical software, with the exception of its coverage for some of the special functions. Examples of these include[15] the logarithm of the Gamma function, the zeroes of Bessel functions and the modified Bessel function of non-integer order. CERN have since provided their implementation of these functions (from CERNLIB) to NAG, who are currently reviewing them for inclusion in the next release of the NAG C Library. This will increase the Library's coverage of HEP requirements, and may also benefit users of the Library working in other fields.

4.2. MINOS-style error calculation in the NAG C Library

The NAG C Library has been used within CERN in the development of a number of projects, including Minuit++, a project aimed at finding a replacement for Minuit (a general function minimization package used by the HEP community for about 30 years) and for some of the homemade fitting tools built on top of Minuit. Following the analysis[16] of the user requirements of Minuit++, it was concluded that most of these were covered by the minimization chapter of the NAG C Library, although a small number of requirements[17] would require extensions to the Library. For example, Minuit users have been able to analyze the errors associated with the minimization via the algorithm used in the MINOS package; this is currently not offered by the C Library minimization chapter. Once again, CERN's user requirements are currently under review for inclusion in the next release of the NAG C Library.

4.3. Plotting histograms in IRIS Explorer

The original version of IRIS Explorer which CERN used had a range of functionality for the display of 3D and 2D datasets, but its ability to deal with 1D data (such as, for example, a histogram) was more limited. This was because one of the early design aims for IRIS Explorer (which was originally developed by SGI) was to highlight 3D graphics hardware. Since taking the package over from SGI, NAG has added some 1D functionality (using its own Graphics Library), but it was clear that more work in this area was needed to improve the display of histograms. Accordingly, the MasterSuite library (see §3.2, above), which contains a wealth of routines for 1D displays, was used to build the HEPInventor library, and modules based on this library were incorporated into HEP Explorer. For the next release of IRIS Explorer, NAG incorporated MasterSuite itself into the package, offering new modules and API based on that library. This improved the ability of IRIS Explorer to deal with 1D data, an enhancement that was welcomed by NAG customers in other areas (for example, the finance industry).

4.4. IRIS Explorer on Linux

Finally in this section, we note the feedback from the HEP community to NAG with regard to supported hardware platforms. That community were among the first to recognize the importance of Linux as an operating system, and their feedback led to NAG's decision to port many of its offerings – in particular, IRIS Explorer – to that platform. The Linux port of IRIS Explorer was announced at CHEP'98[18] and was released the following year[19], since when it has proved to be one of the most popular implementations – not only within the HEP community, of course, but in a wide variety of other applications areas.

5. Conclusions

This paper has looked at the way in which scientific software components from a commercial supplier have been used – where appropriate – within the LHC++ project. The re-use of these standard components, where possible, has freed up the HEP development effort, allowing it to be focussed on the implementation of HEP-specific functionality. This division has been seen as a sensible compromise between the HEP developers building all of the applications from scratch (which, as noted above, could be viewed as an inappropriate use of resource) and a commercial supplier building them to order (which would probably be too expensive, even if the appropriate experience could be found). As the relationship between the project and the supplier developed over time, we saw examples of the way in which some of the features, which were initially thought to be only applicable in the HEP area, were passed back to be incorporated into the commercial system. This had benefits on both sides of the relationship: the HEP developers no longer have to implement those features themselves, while the supplier is able to offer increased functionality to the rest of its user community. In fact, this type of relationship is characteristic of the way in which – as noted above (§2.2) – NAG works with its long-term collaborators. NAG has found the relationship with the HEP community to be a valuable one, and looks forward to further collaboration in the future.

Acknowledgements

I thank Dr Ian Reid (NAG) and Dr Jamie Shiers (CERN) for helpful discussions, and for commenting on an earlier draft of this paper.

References

[1] http://wwwinfo.cern.ch/asd/index.html
[2] http://wwwinfo.cern.ch/asd/paw/index.html
[3] http://wwwinfo.cern.ch/asd/lhc++/about.html
[4] http://wwwinfo.cern.ch/asd/geant4/geant4.html
[5] http://wwwinfo.cern.ch/asd/lhc++/index.html
[6] http://www.nag.co.uk/
[7] http://www.nag.co.uk/numeric/CL.html
[8] http://www.tgs.com/Products/openinv-index.html
[9] http://www.tgs.com/3DMS/index-c++.html
[10] http://wwwinfo.cern.ch/asd/lhc++/dat/hepinventor.html
[11] http://www.nag.co.uk/Welcome_IEC.html
[12] http://wwwinfo.cern.ch/asd/lhc++/HepExplorer/index.html
[13] http://www.nag.co.uk/visual/IE/iecbb/Render/Issues10/issue10_4.html
[14] http://www.nag.co.uk/doc/TechRep/PS/tr9_96.ps
[15] http://wwwinfo.cern.ch/asd/lhc++/requirements/special_functions.html
[16] http://wwwinfo.cern.ch/asd/lhc++/requirements/stable/URD/html/
[17] http://wwwinfo.cern.ch/asd/lhc++/requirements/stable/mark6urd.html
[18] http://www.nag.co.uk/new/1998/Iris_Explorer_pr.html
[19] http://www.nag.co.uk/visual/IE/iecbb/Render/Issue11/issue11_1.html