Analysis for Physics Experiments (Version 2.0.1)

by Andreas Pfeiffer

Analysis for Physics Experiments (Version 2.0.1)

by Andreas Pfeiffer

Published June 2000

Table of Contents

List of Figures

List of Tables

Chapter 1. Presenting Anaphe

A short historical introduction

In 1995 a working group, called *Anaphe* , consisting of representatives of IT Division's ASD Group and of the various LHC experiments, was created to investigate how the approximate equivalent of the current CERNLIB environment could be provided in the LHC era. Although the primary objectives of the discussions concentrated on proposing solutions for future LHC experiments, it is essential that other HEP experiments which are coming online in the next couple of years will also be able to benefit from the proposed strategy.

It is clear that (industry) standard solutions should be used whenever possible. This is nothing new since in the past commercial libraries or programs, such as GKS/GTS, Numerical Algorithms Group (NAG) libraries, Phigs, Historian, and CMZ) have been used in the CERNLIB environment. Intensive investment of manpower should only be made in HEP-specific developments, where the needed functionality cannot be obtained *off the shelf* at affordable prices. An instance of such a HEP development is the CLHEP class library, which is also a nice example of reuse.

Working closely with the LHC collaborations and other HEP Laboratories in 1996 and 1997 an interim strategy was defined and a number of licences for the commercial components was obtained. When several *a priori* solutions were available, a study was made to determine which was the most suited in the HEP context. We also tested the proposed solutions at a few sites and ensured that it is a workable environment.

Objectives

The main objective of Anaphe is to satisfy the requirements of the LHC experiments in terms of the overall software environment that today is provided by CERNLIB, as requested in these experiments' *Computing Technical Proposals*

This translates into the following key points for the short to medium term:

- identify and provide key HEP-specific functions;
- define affordable solutions for the non-HEP-specific parts;
- monitor the non-HEP world for possible future useful software;
- study and understand the requirements of the LHC experiments for C++-based mathematical libraries, and evaluate existing and future developments, both commercial and otherwise in this area;
- study the requirements for a minimisation package (Minuit-replacement);
- undertake a number of pilot projects with the experiments to test the overall functionality of the Anaphe environment;
- supplement, where necessary, the existing documentation, both printed and online, with a set of user guides and tutorials;
- agree with the experiments and other HEP laboratories on a scheme for managing licenses, so that the best possible deals can be negotiated.

The scope of the Anaphe project covers the following:

- foundation level class libraries;
- mathematical libraries;
- graphical libraries;

- visualisation tool-kits, data analysis, histograms;
- event generators (in collaboration with, e.g., Lund);
- detector simulation (GEANT-4);
- object persistency (Objectivity/DB via IT/DB group at present).

The primary focus is on C++-based solutions, although, of course, developments in the software arena, in particular the increased importance of Java, are closely watched.

Collaboration with the experiments

The Anaphe project is a joint effort between IT Division (mainly ASD Group), the LHC experiments, plus NA45, Compass,... Andreas Pfeiffer is responsible for its overall coordination.

Regular (bi-weekly) Anaphe and RD45 meetings are held in Building 40. This is an ideal forum for communication with the physics community since its allows us to bring our users regularly up to date with the latest news. Also, and more importantly, it provides us with an input channel from the experiments about how they use the software, which are possible problem areas which have to be addressed, and what are the future developments they would like to see. The minutes as well as material presented on the progress in the various areas are available on the Web [http://wwwinfo.cern.ch/asd/lhc++/meetings/index.html] .

Twice a year a formal Anaphe Workshop [http://wwwinfo.cern.ch/asd/lhc++/workshops.html] takes place where progress reports are presented by all HEP-wide collaborations who are using Anaphe software. This Workshop provides an efficient forum for feedback from the experiments and permits us to steer long-term development in the right direction by taking into account constraints and requirements of as wide a user base as possible.

Current Situation

By using commodity solutions where-ever possible, we ensure that the proposed solutions are widely used, well-debugged and well documented, and are also more affordable. These solutions are being complemented by HEP-specific components, where needed, by building a HEP user-layer on top of standards-conforming products.

To ensure that the commercial components work well together, the Anaphe strategy closely adheres to standards - both *de-facto* and *de-jure* . Examples of *de-jure* standards include the Standard C++ Library [http://www.objectspace.com/Products/CCS/Standards/standards.html] and ODMG-compliant Object Database Management Systems (ODBMS), while instances of *de-facto* standards are industry standard graphics packages, such as OpenGL, Open Inventor, all originally from Silicon Graphics (SGI).

Table Table 1.1 below shows this layered structure more schematically. Licenses for all commercial components are available at CERN, where the software has been installed on mainstream Unix platforms (Linux, HP, and Sun) on AFS in `/afs/cern.ch/sw/lhcxx` , while on Nice/Windows/NT it is available under `z:\p32\lhcxx` .

All of the commercial Anaphe components come with excellent online documentation. In most cases, printed documentation, often in the form of published books, is also available and can be bought from the *Computing Bookshop* in IT Division at CERN. HEP-specific examples and other information specific to the HEP environment is clearly not available from the vendors.

Table 1.1. Layered structure of Anaphe system

GEANT-4, MCLIBS++
OpenGL, Open Inventor
HepFitting and GEMINI
NAG C library (with C++ headers)
CLHEP
HTL
HEPODBMS
ODMG, ODBMS (Objectivity/DB) + persistent STL
Standard C++ Libraries

The main purpose of the present manual is to provide a tutorial introduction to the use of Anaphe tools for physicist new to the Anaphe computing paradigm. The current guide describes the present state of some of the HEP extensions. Their precise form and application program interface (API) probably needs to be refined or extended in a few places. Therefore we invite all users of the Anaphe software to forward their comments and suggestions to the Anaphe team, preferably at the Anaphe regular meetings mentioned above.

An overview of the commercial components

The commercial components of Anaphe are chosen because they offer a coherent set of inter-operable solutions. They are built on standards and often come as part of the standard hardware or software bundled with the computer. Cost effectiveness has also been optimised both for CERN and for the general CERN HEP program participants.

OpenGL

OpenGL [http://www.sgi.com/Products/Dev_environ_ds.html] is an industry standard for graphics. It is vendor-neutral and multi-platform, and is optimised for building environments for developing 2D and 3D visual applications. Several vendors already offer a hardware implementation of the standard, thus ensuring that rendering speed will be optimal.

The about 250 OpenGL procedures provide a wide range of graphics features, such as a set of geometric and raster primitives, various colour modes, display list or immediate mode, viewing and modelling transformations, lighting and shading, hidden surface removal and translucency, anti-aliasing, texture mapping, effects using fog, smoke, or haze, etc. As all licensed OpenGL implementations are required to pass a set of conformance tests, and implement the same specification and language binding document full portability between multiple platforms is guaranteed.

Documentation is available as two books: the *OpenGL Programming Guide* , and the *OpenGL Reference Manual* , both published by Addison and Wesley (available from the *Computing Bookshop* in IT Division).

Open Inventor

Open Inventor [http://www.sgi.com/Technology/Inventor/index.html] is an object-oriented 3D toolkit to provide a comprehensive solution to interactive graphics programming. Its programming model is based on a 3D scene database optimised to ease building graphics applications. It includes a large set of objects, such as cubes, polygons, text, materials, cameras, lights, track-balls, handle boxes, 3D viewers, and editors.

Open Inventor is built on top of OpenGL. It defines a standard file format (IV) for 3D data interchange and introduces a simple event model for 3D interaction. Animation is provided with *Engines* . Open Inventor offers a convenient multi-platform 3D graphics development environment, which allows efficient manipulation of objects in a windows and operating system independent way.

Open Inventor's IV files serve as the basis for the *VRML (Virtual Reality Modelling Language)* standard [http://vrml.wired.com] . The Open Inventor toolkit is conveniently documented in three books *The Inventor Mentor* , *The Inventor Toolmaker* , and *The Open Inventor C++ Reference Manual* published by Addison-Wesley (available from the CERN Computing Bookshop).

Objectivity/DB, the Object Database

In order to study solutions for storing and handling the multi-Pbyte data samples expected with LHC, the *RD45 Project* [http://wwwinfo.cern.ch/asd/cernlib/rd45/index.html] was established in 1995. The proposed solution should also be able to cope with other persistent objects, such as histograms, calibration monitoring data, etc. It was found that the best candidate for handling this problem is a *ODMG* (Object Database Management Group) [http://www.odmg.org/] , compliant object database used together with a mass storage system, based upon the IEEE reference model for mass storage systems. After considering a few alternatives, the presently favoured solution uses *Objectivity/DB* [http://www.objy.com/] and *HPSS* (High Performance Storage System) [http://www.sdsc.edu/hpss/] .

The Objectivity/DB database provides object-persistency services for GEANT-4 and the experimental data. It must fully support HEP meta-data, not only the persistent data collections themselves, but it must also handle the selections producing these collections, and the predicates themselves. Replication of large database images on local area and wide area network configurations containing heterogeneous hardware must allow collaborators all over the world to actively participate in data analysis. Given the large time scales (the lifetime of the LHC software will span at least twenty years) schema evolution and versioning are important aspects which must be taken into account. Objectivity/DB comes with a set of administrative tools to ease database management. Objectivity/DB also comes with an *Advanced Multi-threaded Server* (AMS) and an interface to HPSS. This provides fast access to data and offers a large performance improvement when updating data stored in remote databases. It is expected that each experiment will run starting in 1998 a production service of one or more of these servers.

Objectivity/DB has a layered logical storage level, with at its top the *federated database* . Each federated database logically contains one or more *databases* , with the latter containing the *objects* , which are clustered, for efficiency, inside *containers* . An *object* itself consists of standard C++ constructs, variable-size arrays, relationships and references to other objects, and type constraints. Persistent objects can be created and deleted dynamically by any application. The data model, or *schema* is stored inside the federated database.

Mathematical Libraries

CERN no longer has any in-house mathematician supporting mathematics libraries. Therefore, we shall have to rely on libraries developed outside CERN, and it was decided to make the *NAG C-language* [http://www.nag.co.uk/numeric/CL.html] library available. Although the NAG C Library provides the basic functionality required by HEP, a small number of routines (basically special functions) are currently unavailable. A future release of the above library is likely to incorporate these routines.

Statistical Data Analysis: the Gemini package

Gemini is a GEneral MINImization and error analysis package implemented as a C++ class library. Minuit's functionality is provided in a *Minuit-style* (even if, internally, another minimizer may actually do the work) and new functionality offered by NAG C minimizers is added. Gemini thus provides a unified C++ API both to standard Minuit and to NAG C family of minimizers. For the common subset of functionality, it is up to the user which minimization engine does the work: Minuit or NAG C. The user can easily switch between various minimizers without essential changes in the application code. The currently supported set of minimizers (Minuit and NAG C) can be extended without essential changes in the API.

The abstract base class `GEmini` defines an interface to the common functionality. The `CMinuit` class is derived from `GEmini` and provides a Minuit-based implementation of the GEmini functionality plus Minuit-specific extensions. Similarly, the `NAGmin` class is derived from `GEmini` as well and provides a NAG-based implementation of the `GEmini` functionality plus NAG-specific extensions.

There is no single class which contains references both to Minuit and to NAG C, so that orthodox Minuit or Nag C users are not forced to link the other library.

Gemini finds a minimum of an objective function, possibly subject to general constraints, and performs an error analysis. The concept of errors is that of Minuit, so that it is the user's responsibility to properly scale the inversed Hessian, and to properly interpret the results. Both Hessian based errors and Minos errors are implemented. Correspondingly, two types of function contours (or confidence regions, in statistical problems) are available: elliptical and Minos ones. Minos error analysis is, however, possible only for bound constraint problems.

Overview of the HEP-specific components

Although commercial and public-domain packages offer a great deal of functionality, there is a clear need to supplement them with HEP-specific extensions. Some of these extensions take the form of complete class-libraries, such as CLHEP. Others represent large toolkits, such as *GEANT-4* [http://wwwinfo.cern.ch/asd/geant4/geant4.html] , in areas such as graphics and visualisation, the basic tools, such as Open Inventor for basic 3D graphics and Qt for 2D graphics for visualisation, need to be extended to cope with the specific needs of the HEP experiments.

HEPODBMS

HepODBMS is a set of class libraries built on top of the ODMG C++ interface. Their purpose is to provide a higher level interface than is specified by the ODMG, to simplify the porting of existing applications and provide a minimum level of support for transient-persistent switching. Furthermore, these libraries help to insulate applications against changes between releases from a given vendor and between the products of different vendors.

CLHEP

The *CLHEP* project [http://wwwinfo.cern.ch/asd/lhc++/clhep/index.html] was initiated at CHEP'92; it intends to provide *foundation level* classes required in HEP. At present they include:

- `Alist` for lists and list iterators;
- `Combination` ;
- `Geometry` for vectors, rotations, transformations;

- `Matrix` for matrix manipulations;
- `Random` for random numbers;
- `String` for different string types;
- `Units` for system of units and physical constants;
- `Vector` for vector operations (3-vector and Lorentz-type).

CLHEP became formally part of Anaphe in 1995. The first official release of CLHEP (V1.0) took place in April 1997 (CHEP'97). CLHEP-based classes will be integrated in the beta-release of GEANT 4 early 1998. The complete user documentation, with a detailed description of all classes is being written and will be available by the end of 1997.

The HTL class library

The HTL class library provides Object Oriented histograms. They come in two versions:

- Persistent histograms (based on Objectivity/DB)
- Transient histograms (text file I/O)

XML based persistency for Transient HTL is currently under study.

HepVis

The goal of the *HepVis Project* [http://physics.web.cern.ch/Physics/Workshops/hepvis/] is to create and distribute a toolkit library consisting of graphical objects capable of representing the most common entities of a collider physics experiment. Previous experience has shown that mere representations of objects on a workstation screen is insufficient, and that native support for picking objects with user-defined actions, and a high-degree of interactivity, both local and global, is needed. Therefore, the HepVis toolkit is being implemented as an extension to Open Inventor, providing common physics objects as subclasses or as real extensions. Only the graphical representation of the objects will be defined, leaving it up to the experiments to define physics objects and their behaviour, and whether to integrate these with the graphical objects in question.

Problem Reporting

Due to the phase out of the GNATS problem reporting system at Cern, users should submit their problem reports by sending a mail to `<Heplib.Support@cern.ch>`.