CLHEP - New Developments and Directions

M. Fischler[1]*, A. Pfeiffer*[2]

[1] Fermilab, USA
[2] CERN, Switzerland

Abstract

CLHEP is a broadly based collaboration, including developers at CERN, Cornell, FNAL, SLAC and elsewhere, to provide core C++ class libraries for the use of the HENP community. Recently, the collaboration aspect of CLHEP has been rejuvenated and expanded.

To provide structure for the expanded collaboration, the developers have agreed to a consensus based organization centered around an "editors list." An overview of the new structure, including the mechanism for ensuring CLHEP remains responsive to the needs of HENP users and other developers, will be presented.

To guide directions of development, the nature of CLHEP has been clarified: The "core" CLHEP is structured as a set of packages which must have freely available sources and must be independent of external packages. These must work on at least a large fraction of the platforms in general use. The selection of these packages is driven by those needs and timetables of the High Energy and Nuclear Physics community which would not likely be met by code emerging from the general computing community.

Coordination of CLHEP with other packages developed in HENP will be discussed, and recent and planned extensions to CLHEP will be presented.

Keywords: CLHEP, libraries, C++

1 Introduction–Organization of CLHEP

CLHEP represents an effort in the High Energy and Nuclear Physics community (HENP) to take advantage of the "reusability" of Object-Oriented code, by providing dependable classes to accomplish those tasks which are more crucial to HENP developers than to the world at large. As HENP has successfully shared subroutine libraries in the past, developers searched for a means of providing the benefits of these collections (such as the CERNLIB [1] libraries) in the new context of object-oriented programming in C++. Out of this, CLHEP originated in 1992 [2] as a collection of useful C++ packages, with voluntary contributions and only loosely defined standards of package suitability, maintenance responsibility, and overall organization. This evolved into a loose structure in which designated "editors" took responsibility for each package.

Last Spring, several CLHEP editors conferred with potential collaborators in the US and elsewhere, reaching agreements which define the present CLHEP, expand the direct collaboration, and provide guidelines for how CLHEP is to operate and evolve. The main features of these agreements are (see also `wwwinfo.cern.ch/asd/lhc++/clhep/base-0.0.0.html`):

- The "Core CLHEP" is independent of external licenses or products, and contains only packages for which sources are available.
- CLHEP decisions are made by consensus within an "editors list". This list consists of package editors and key representatives of the user community. No formal voting procedures or

designation of management authority were adopted; the feeling is that if the limited group of editors cannot reach consensus, then the collaborative endeavour is doomed in any event.

- Each CLHEP package will avoid casual dependencies on other packages. Any fundamental dependencies which cannot be removed will be indicated, and must follow a "no cyclic dependencies" rule. Useful classes which would be associated with one package but which depend on another are placed in a special "Tools" package.

The current CLHEP packages, and the assignment of editors, may be found at the "Packages and Dependencies" link from the CLHEP web page `wwwinfo.cern.ch/asd/lhc++/clhep`.

Each package is assigned an editor who maintains the code and documentation and who leads any enhancement efforts for that package. An editor may or may not be the original developer of the code, and contributions and improvements from non-editors are welcome. The package editor will filter these contributions to avoid arbitrary expansion, ensure consistent interfaces, and validate the correctness of the added code.

In addition to maintenance of existing packages, the editors list controls the incorporation of new packages into CLHEP, as well as organizational, release, and presentation issues. A most important role is change control–any *changes to interfaces* of packages and feature enhancements are considered by the editors before they are exposed to a broader community of users. Routine enhancements without the potential for breaking user code need not be subject to extended debate, and innocuous changes such as bug-fixes and documentation clarification may be made by the package editor without much discussion.

The source code for each package is is kept in a central CLHEP cvs repository. Periodically, a "production release" will be declared, with a solid tagged version of each package placed into a publicly accessible code area. Development releases, for users requiring more recent features, are also prepared. A development release may be declared because of a major enhancement of one specific package. In any event, code in any release is not placed in the public code areas until it has been validated by running the appropriate test suites for all the packages.

Only people working on CLHEP development will directly access the heads of the the repository, which may at any given time contain code which has not yet been validated.

2 Relation to Other Efforts

This section describes the place of CLHEP in the context of other major HENP software efforts. This represents consensus among the editors rather than formal agreements with each of the other groups. However, the consensus was not formed in isolation, and there appears to be little contention about the role of CLHEP in regard to these efforts.

Perhaps the largest HENP software project is the LHC++ project, centered at CERN. This aims to provide class-libraries and object-oriented tools for physics data analysis used by present and future experiments, based on C++ (and possibly Java). [3]

It is intended that core CLHEP be a component of this software: a kernel of classes well suited for this purpose which can be drawn upon without concern for cost and licensing issues. The agreed requirements for CLHEP packages were designed with this in mind. LHC++ software needs are likely to include many varieties of packages. Those packages which are of general interest, and can sensibly be developed such that the sources and packages are freely available, are natural candidates for inclusion in CLHEP. Thus, close coordination between the CLHEP editors and the LHC++ software efforts is important; fortunately there is substantial overlap among these groups.

At Fermilab, the ZOOM [5] task force has libraries of code aimed at needs of the CDF and D0 Run II experiments. Although ZOOM has from the start provided a "CLHEP" module for the convenience of Run II users, until recently some similar packages were developed in both libraries.

A major thrust of the CLHEP organization agreements has been to sketch how to coordinate with, and take advantage of, various ZOOM packages. PhysicsVectors will be integrated into the Vectors package; portability, error logging, and the ZOOM Exceptions modules will be adapted for incorporation into CLHEP. New Fermilab-developed packages including a C++ manifestation of the StdHep [6] code (StdHep++) are also being incorporated into CLHEP. Not every pair of similar packages can easily be combined. It was felt that integrating the ZOOM LinearAlgebra and CLHEP Matrix packages would not be worth the considerable effort required.

An example of other software collaborations with special ties to CLHEP is Geant4. [4] Historically, several CLHEP packages were contributed by Geant4 developers, or were provided to meet Geant4 needs. Although CLHEP should certainly continue to meet such needs, Geant4 is a relatively mature program and requires few new core packages.

Similarly, CLHEP has ties to large experiments at various labs. Several CLHEP packages developed at the SLAC BaBar experiment, or were influenced by its requirements, and a substantial portion of BaBar's code is to some extent based on existing CLHEP packages.

It is important that CLHEP avoid "breaking" Geant4 or experiments' working code by introducing changes into CLHEP classes. Coordination is necessary so that these key collaborations do not feel forced to protect their code stability by maintaining separate versions of crucial CLHEP packages–divergent versions of the core CLHEP library would be an undesirable situation.

3 Extensions to Current Packages

3.1 Vectors

Of the current packages, the Vector package is undergoing significant extension: the ZOOM PhysicsVectors classes are being integrated with CLHEP Vectors. These packages have classes including ThreeVector (or SpaceVector), LorentzVector, Rotation, and so forth.

Although the fundamental designs of the two packages have largely converged in the last few years, the ZOOM classes have quite a few useful physics methods which are lacking in the CLHEP Vector classes. As a consequence, however, the interface to PhysicsVectors is so extensive that for many uses, physicists desire the smaller CLHEP classes. Also, several important HENP packages work with CLHEP Vectors; coders should be able to pass the ZOOM classes where CLHEP objects are expected. Therefore, we are recasting the ZOOM classes to inherit from the existing CLHEP classes. Given that approach, the remaining concern is to ensure that the new PhysicsVectors code works on the wide variety of platforms that CLHEP is used on. To facilitate this, two simplifications are being applied:

- PhysicsVectors makes use of classes in the ZOOM Exceptions package, which has not yet been brought into CLHEP. In the interim, a "dummy" exceptions package, which uses only trivial C++ constructs, will be attached to the Vectors package.
- PhysicsVectors provides an unusual syntax for accessing components in various coordinate systems, for example, `v.r() = 2`. This syntax will be abandoned; the more familiar (and simpler to implement) `v.setR(2)` remains supported.

The extensive ZOOM validation suite will be applied to the re-formulated derived classes, testing not only the transition, but the correctness of the underlying CLHEP methods as well.

3.2 Other Packages

The newly created Tools package contains a RandMultiGauss class. This random distribution fills the last gap with respect to the distributions listed in Review of Particle Physics, but it was not placed in the Random package because the natural way to accept the covariance matrix uses con-

structs from the Matrix package.

A new and significantly faster form of the RandPoisson distribution, and new distributions representing still faster approximations to Gaussian and Poisson variates have been added. For example, RandGaussQ (the Q indicates "quick") uses table techniques to rapidly supply variates which match the Gaussian almost (but not quite) perfectly. In the same vein, we intend to supply a very fast Random engine for applications which require high speed and can accept possible small imperfections in randomness.

4 New and Planned Packages

It has been agreed that contributions of new packages are subject to approval of the editors distribution list. Conditions imposed on the new package include:

- The package should not depend on outside products and the sources must be available.
- The submitter must commit to providing documentation in LaTeX and HTML forms, and an automated validation suite.
- The interface for the package must be complete and intended to remain stable; some implementation must already exist.

An editor must be identified, to take responsibility for supporting the new package. The developer submitting a package does not necessarily become its editor, though if the time commitment can be made that is a natural choice.

The new StdHep++ package [6], edited by Lynn Garren, provides a bridge between various event generators and simulation software. This provides in C++ the capabilities of the Fortran stdhep package [7], and is being enhanced to take advantage of object-oriented concepts.

The ZOOM Exceptions package, supporting user-defined handling of various Exceptions with or without C++ exceptions being enabled, and the ErrorLogger package providing uniform formatting, statistics, and filtered dispatching to multiple logs, are slated for inclusion in CLHEP.

A package of Kalman filter and related classes, developed by J. Boudreau, is being cast into CLHEP form. New classes representing templated basic physical vectors are being developed.

The ZOOM SIunits [8] package, which provides automatic units checking with no runtime speed or space cost, is being evaluated–the issue being whether enough compilers can handle its template operations to be worth including it in CLHEP. Lastly, a newly developed C++ portability package [9] is being considered for augmenting the existing CLHEP configuration mechanism.

References

1 *CERNLIB Short Writeups*, Applications Software Group, CERN (1995).

2 L. Lönnblad, *Comput. Phys. Commun.* **84** (1994) 307.

3 *Libraries for HENP Computing – the LHC++ project*, A. Pfeiffer, Proceedings of CHEP 2000, abstract number 83; and `wwwinfo.cern.ch/asd/lhc++/guide.html`.

4 Simone Giani, *Proceedings of CHEP 95* (1995) 147.

5 *The ZOOM Fermilab Physics Class Libraries*, M. Fischler, W. Brown, P. Canal and J. Marraffino, Fermilab preprint Conf-98/322 (presented at CHEP 98).

6 *StdHepC++*, L. Garren, Proceedings of CHEP 2000, abstract number 205.

7 *StdHep 4.08 Monte Carlo Standardization at FNAL*, L. Garren, Fermilab PM0091.

8 *Introduction to the SI Library of Unit-based Computation*, W. Brown, Fermilab preprint Conf-98/328 (presented at CHEP 98).

9 *ISOcxx: The C++ Portability Package*, W. Brown, Proc. CHEP 2000, abstract 139