LCnote LC-TOOL-2004-016

Simulation and Software Tools for the Linear Collider

Ties Behnke
DESY, Notkestrasse 85, D-22603 Hamburg, Germany

The development and the optimization of the physics case of the linear collider and of a detector for a linear collider rely to a large extent on sophisticated software packages. In this paper the currently available software packages are briefly reviewed, and a view of the possible development of a world wide coordinated software development environment is discussed.

1 Introduction

The international community of particle physicists has embraced a linear electron positron collider as the next big project in the field. This accelerator should reach energies of around 1 TeV and deliver high luminosity in excess of $10^{34} cm^{-2} s^{-1}$. A key role in the design of the experimental program at this collider is played by software tools. The complexity of modern particle physics experiments is such that extensive simulations and Monte Carlo experiments are needed to optimize the design, and to demonstrate that the proposed solution will meet the requirements.

2 The Current Status

The steps needed in a complete simulation of a detector at a linear collider are illustrated in Figure 1.

The simulation of the interaction of the detector with particles is typically done by complex packages like GEANT3. A simulation framework then is providing an interface to this tool-box. The main challenge of the other three steps is to provide a common framework, which allows the user to easily include new reconstruction or analysis packages, but which ensures and manages the information exchange between different parts of the program.

3 LCIO, the Linear Collider Input Output System and Data Model

LCIO [1] is a software system which has been developed in collaboration between DESY, SLAC and LLR. It defines a common data model and persistency framework for linear collider studies. As such this package is at the core of any software environment for linear collider studies. The requirements for the system were that it be light-weight and simple to use, that it provide a set

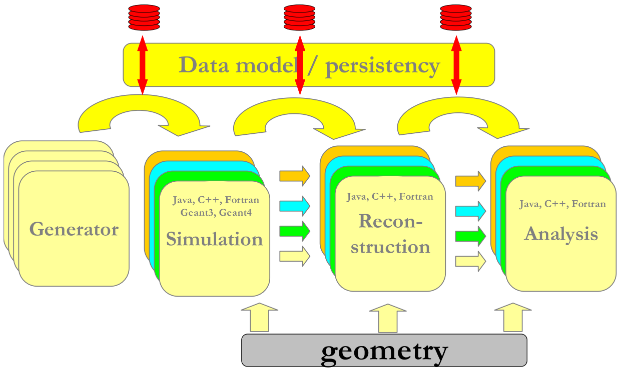

Figure 1: Schematic representation of the role of LCIO in a software environment at the linear collider.

of generic data classes to describe things like hits, tracks, clusters etc, that it be easily expandable, portable, and that it be available in a number of different programming environments. Currently implementations of this system are available in C++, Java and Fortran.

To ensure that the implementations in the different languages all provide the same functionality and structure, the definition of the class hierarchy and the API is done in a separate, abstract language called AID [2]. In this way the consistency between the implementations is always guaranteed. The FORTRAN implementation models as closely as possible the OO design of the JAVA and C++ versions, and is built as a series of wrapper functions around the C++ implementation.

LCIO has been released to the community in the fall of 2002. It has been accepted as the standard for IO and for the data model by the European linear collider workshop, by the American linear collider physics group, and is being considered by the Asian groups.

4 Simulation Programs

Over the last few years GEANT4 has become the accepted standard for simulation studies in the linear collider community. Nevertheless BRAHMS, the GEANT3 based simulation and reconstruction framework developed within the European linear collider study [3], continues to be used and still serves as the reference implementation of the TESLA detector.

The detector response simulation program MOKKA [4] has been under development for a number of years. MOKKA is based on the GEANT4 toolkit, and provides a layer around this to allow easy geometry definition and interrogation. Geometry information is stored in a relational database, which is queried at run time. C++ driver routines translate the information in the database into GEANT4 geometry objects. An interface layer called CGA gives access to the geometry information at any time through a series of simple access methods. In this way simulation, reconstruction and possibly analysis can access the geometry information in the same way, from the same information source.

During the last year two other GEANT4 based simulation programs have become available: LCDG4 and LCS [5]. Both have been developed within the American linear collider community. The main difference to MOKKA is the way geometries are defined: LCDG4 and LCS use a simple XML description to define the geometry. The XML files are parsed at run time, and translated into GEANT4 geometry objects. At the moment only projective geometries are supported, and the only shapes implemented are cylinders and disks.

Within the Asian community the JUPITER framework has been developed. It is based on root, which provides both the user interface and the persistency mechanism.

The authors of all programs are committed to make the different simulators LCIO compliant.

Many physics studies rely on the simulation of large numbers of events. For this often the full GEANT based simulation programs are too slow, making fast simulation programs a necessity. Lately a movement is visible to introduce an intermediate form of simulation program. These programs create, using a simplified algorithm, hits in the different detector components, and thus produce files of simulated detector data which are very similar to the ones produced by a full simulation system. Examples for this are SGV [6] or, more recently, LeLaps [7].

5 Reconstruction Tools

The paradigm of particle flow [8] has been widely accepted as the appropriate method to reconstruct the events at the linear collider. Particle flow means that every particle in the event is reconstructed, both neutral and charged particles. It has been demonstrated that this might be possible in a detector which combines excellent tracking with a highly granular calorimeter. The emphasis of the reconstruction is thus primarily on the separation and recognition of particles, and only thereafter on the determination of the parameters of the particles.

This paradigm requires totally new reconstruction tools. A number of groups are working on such an algorithms, and have reported in detail elsewhere in these proceedings.

Efforts are under way to develop a software framework which will ease the development and use of such algorithm. There is agreement that LCIO will provide the data model for this framework, defining basic reconstructed quantities like tracks and clusters, and more abstract objects like reconstructed particles. Basing this effort on LCIO has the advantage that many separate tools can then be used without additional effort. A good example of this is the WIRED event display [9]. Using the JAS package [10] WIRED can be used on data recorded in the LCIO format, without additional programming.

6 Conclusion

Physics studies and detector development for a linear collider require powerful software tools. The international linear collider community is trying to provide such tools to the community. Over the last couple of months, an important first step towards a common simulation and reconstruction framework has been taken by the development of LCIO, the Linear Collider Input Output system. The collaborative model used in the development of this software has worked very well, and is currently being applied to expand the scope of the software to include simulation and reconstruction tools.

References

1. F. Gaede et.al, LC-TOOL-2003-053, see *http://www-flc.desy.de/lcnotes*
2. See the homepage at *http://java.freehep.org/aid/index.html.*
3. T. Behnke et.al, LC-TOOL-2001-005; see *http://www-flc.desy.de/lcnotes*
4. See *http://polywww.in2p3.fr/geant4/tesla/www/mokka/mokka.html.*
5. see e.g. N. Graf, ALCPG workshop at SLAC, 2004, *http://www-conf.slac.stanford.edu/alcpg04/Default.htm*
6. See *http://home.cern.ch/b/berggren/www/sgv.html.*
7. W.G.J. Langefeld: see *http://www-conf.slac.stanford.edu/lcdsimworkshop/talks/lelaps.ppt.*
8. ECFA/DESY LC physics working group: TESLA TDR part 3, e-print hep-ph/0106315
9. See *http://wired.freehep.org.*
10. See *http://jas.freehep.org.*