Linear Collider Detector R&D Software

September 4, 2015

1 LCIO

1.1 Introduction

1.2 Recent Milestones

The LCIO software toolkit [**?**] was developed to provide a common event data model (EDM) and persistency format for Linear Collider physics and detector simulations. It was developed as a joint effort between SLAC and DESY and has been adopted by all of the detector concepts for both ILC and CLIC. Many of the subdetector R&D groups (e.g. CALICE and LCTPC) have also adopted LCIO for both their simulation needs and for testbeam data. Major R&D Efforts: The software toolkit consists of an Application Programming Interface (API) as well as reference implementations in Java and C++ and a binding to python. This, plus its deliberately simple design and well-documented EDM, has allowed the LC community to mix-and-match its software applications. Events simulated in C++ can be reconstructed in Java and analysed in python, providing enormous flexibility to the end user, who can concentrate on analyses and not be hampered by programming language limitations.

1.3 Engineering Challenges

The implementation of a complete EDM for all HEP applications is very difficult, if not impossible. LCIO has succeeded by reducing the problem to its simplest solution, but providing end users flexibility to adapt to their specific needs. Custom classes can be implemented using extensions to existing classes or using generic objects and relations between collections of data. The LCIO development team has also added functionality as new classes have been requested by users. Maintaining strict control over the structure of the LCIO EDM and persistency has ensured that any LCIO file can be opened and interpreted without having access to the code which created it.

1.4 Future Plans

Continued development of LCIO will be driven by user demand and developers resources.

1.5 Applications Outside of Linear Colliders

The Heavy Photon Search experiment at Thomas Jefferson National Laboratory has adopted LCIO as its event data model and data persistency format. Physics and detector studies for CLIC and the Muon Collider have also used LCIO. The authors of the Whizard [**?**] event generator have expressed interest in using LCIO as their binary persistency format for Monte Carlo events. This could lead to its integration into other experiments. Because of its simple and well-documented persistency format LCIO is a perfect candidate for HEP data archiving applications.

2 DD4hep

2.1 Collaborating Institutions

The Detector Description for HEP: **DD4hep** [1, 2] is a joint development between CERN and DESY. The project originally started as part of the AIDA project WP2 and is currently further improved and extended in the context of AIDA-2020.

2.2 Introduction

DD4hep provides a generic, consistent and complete detector description, including geometry, materials, visualization, readout, alignment and calibration. It supports the full experiment life cycle: from detector concept development over detector optimization and construction to the operation phase. A single source of information is used for simulation, reconstruction and analysis, where different interfaces and formats are provided as needed. DD4hep is implemented using the ROOT geometry package TGeom.

2.3 Recent Milestones

The core of DD4hep provides the necessary code and tools for a complete and flexible detector description, based on C++ classes per sub detector and corresponding XML files holding parameters. It provides a palette of simple and generic sub detector geometry classes, which allows new users to get started very quickly with using DD4hep by simply adapting the XML parameters files to define a new particle physics detector. Advanced users can write their own detector descriptors to incorporate any level of detail that is needed. Recently a complete toolkit (DDG4) [3] for running a Geant4 based detector simulation based on a DD4hep detector model has been developed. It provides software modules for fully configuring and running a simulation application, including reading of generator files in various formats, overlaying several events, linking Monte-Carlo truth information to hits and creation of the final output files in the LCIO file format. The programs can either be run as a python application or a C++ application with XML configuration files. Recently the current Mokka simulation models for ILD (ILD_o1_v05/ILD_o2_v05) have been fully ported to DD4hep (in the *lcgeo* package) and the CLICdp group has described their new detector model exclusively in DD4hep/lcgeo.

2.4 Engineering Challenges

One of the most challenging aspects in the implementation of DD4hep layed in hiding some of the complexity and technicalities involved in detailed geometry models from the user in order to facilitate the development of maintainable experiment detector description code. A considerable fraction of the complexity is created by the fact that ROOT and Geant4 have independent implementations of the geometry classes, which partly differ in constructor arguments (meaning and order) as well as in a different set of units used in the two systems. Another challenge will be to make DD4hep compatible with multi-threading applications for simulation and reconstruction/analysis.

2.5 Future Plans

The improvement of the core functionality as well the development of new features in DD4hep will continue over the next years. Besides addressing the multi-threading needs of the community, an interface to conditions and alignment data is on the list of extension projects already identified

for DD4hep. Additional requirements and requests brought forward by the user community will have to be addressed.

2.6 Applications Outside of Linear Colliders

DD4hep has been designed from the start as a generic tool that can be applied to any particle physics experiment. It is currently used by the ILD and CLICdp detector concepts as the main source of detector geometry information and simulation application (based on DDG4). The three FCC studies (FCC-ee, FCC-hh, FCC-eh) have recently also decided to base their software chain on DD4hep and will use it for the conceptual design reports in the next years.

3 Marlin

3.1 Collaborating Institutions

The application framework **Marlin** (Modular Analysis and Reconstruction for the Linear Collider) is developed and maintained at DESY.

3.2 Introduction

Marlin [4] is a C++ application framework for processing LCIO event data files. It is modular, lightweight and easy to use. Marlin applications are fully configured with XML files. Software modules - called processors - have their own section in the configuration file, where all parameters local to the processor are defined. By design every parameter has to be registered by the author including a short documentation, which allows a Marlin application to provide a complete and fully documented example configuration file. The LCIO event data model is used as a so called internal data bus (or whiteboard), i.e. every Marlin processor can read its input collection(s) from the LCIO file and create one or more output collections.

3.3 Recent Milestones

As Marlin is at the core of the the data processing software for the Linear Collider cummunity, an effort has been made to keep it stable and robust. It is used by ILD, CLICdp and partly SiD, as well as by almost all test beam collaborations in the context of the Linear Colliders. Wherever possible new features have been added in backward compatible way, such that existing steering files would work as before. Some of the improvments that have been recently added to Marlin are:

- addition of command line parameters, where every parameter present in the XML file can be overwritten on the command line - useful for scripting and bulk processing
- optionally the LCIO collections that are read from the file can be limited, possibly resulting in greatly improved processing speed
- introduction of the global flag: *AllowToModifyEvent* for cases that input data collections need corrections or additions

3.4 Engineering Challenges

There were no major engineering challenges involved in the development and maintenance of Marlin. As pointed out above, the main challenge layed in keeping Marlin stable, usable and robust. Adopting Marlin to parallel processing and multithreading, which is planned for the near future, will however be a very complex and challenging process, where we will need to learn from the work done by the LHC experiments in this context.

3.5 Future Plans

The improvement of the core functionality as well as the development of new features in Marlin will continue over the next years. Making Marlin capable of processing several events in parallel in order to make better use of new multi-core hardware will be the most demanding new development in the near future. Additional requirements and requests brought forward by the user community will be addressed.

3.6 Applications Outside of Linear Colliders

Even though Marlin as well as LCIO have Linear Collider in their names, both are rather generic software tools that can and actually are used outside of the LC community. For example the EUTelescope software framework is based on Marlin and is used by Atlas and CMS groups in the context of the detector upgrade R&D programme.

4 LCSim

4.1 Collaborating Institutions

The core software has been developed at SLAC. A number of packages were contributed by university and other national lab groups when such efforts were supported by DOE.

4.2 Introduction

4.3 Recent Milestones

Simulation of physics processes and detector response is crucial to the design of new HEP experiments such as those proposed for the ILC. There are stringent requirements on the design of tools for detector R&D which differentiate them from typical experiment-specific simulation and reconstruction code. They must:

- allow easy reconfiguration to support different detector geometries and technologies,
- make it easy to develop, implement and compare new reconstruction algorithms,
- be very easy for users to set up and quickly become productive with,
- work on a wide variety of operating systems and computing platforms,
- be easy to develop and support using a fraction of the manpower that would be available to an established experimental collaboration.

The lcsim physics and detector response simulation and event reconstruction toolkit was developed at SLAC to provide a flexible and performant suite of software programs to allow fast and efficient studies of multiple detector designs for the ILC. The primary goal of the group has been to develop computing infrastructure to allow physicists from universities and other labs to quickly and easily conduct physics analyses and contribute to detector R&D. These tools include the Geant4-based detector response simulation program (slic), and the Java-based reconstruction and analysis tools (org.lcsim) [**?**].

4.4 Engineering Challenges

The lcsim software pioneered the use of runtime-defined detector geometries. Although LCIO [**?**] provided a common event data model for all ILC groups, the community was never able to agree upon a common geometry system for both simulation and reconstruction. DD4hep [**?**] is an effort within the AIDA Common Software Tools project to provide such functionality. Although it adopted many lcsim concepts, the implementation of the software has taken its own course. The largest challenge to the lcsim effort at the moment (beyond lack of funding) is to maintain some connection to the geometry definition functionality promised by DD4hep.

4.5 Future Plans

The core functionality is being kept current by upgrading to the latest versions of Geant4, etc. Due to lack of funding, the project is currently primarily responding to user requests for additional functionality.

4.6 Applications Outside of Linear Colliders

The flexibility and power of this simulation package make it not only useful for the application domain for which it was developed (viz. HEP collider detector physics), but also for other physics experiments, and could very easily be applied to other disciplines, e.g. biomedical or aerospace, to efficiently use the full power of the Geant4 toolkit to simulate the interaction of particles with fields and matter. The Heavy Photon Search experiment at Thomas Jefferson National Laboratory has adopted slic as its detector response simulation package and the org.lcsim toolkit for its event reconstruction needs. Physics and detector studies for CLIC and the Muon Collider have also used both slic and the org.lcsim software. The software could be easily used for physics and detector studies at detectors at future circular colliders.

5 PandoraPFA

5.1 Collaborating Institutions

Development of the Pandora framework and algorithms has been based exclusively in Cambridge, but detector optimisation studies have involved close collaboration with other institutes. ECAL and analogue HCAL studies have involved DESY, Shinsu University (Japan), and CERN. Upcoming (semi-)digital HCAL studies will involve the University of Lyon (France).

5.2 Introduction

5.3 Recent Milestones

The PandoraPFA software package [**?**, **?**] has been developed entirely in Cambridge. It consists of a robust and efficient C++ software development kit (SDK) and libraries of reusable pattern-recognition algorithms that exploit functionality provided by the SDK. Algorithms have been developed to provide a particle flow reconstruction of events in fine-granularity detectors, such as those proposed for use at the ILC or CLIC. The reconstruction uses over 60 algorithms in order to carefully trace the paths of visible particles through the detector. The output is a complete list of the particles in an event, each with a reconstructed four-momentum and an identified particle-type. The algorithms represent the state-of-the-art in particle flow calorimetry at a Linear Collider. The Pandora Linear Collider algorithms have recently been used for extensive detector optimisation studies, assessing the physics performance of the ILD_o1_v06 detector model with different configurations of the electromagnetic and hadronic calorimeters. A selection of the key plots is shown overleaf. A summary document is under construction and will be submitted for publication.

5.4 Engineering Challenges

The implementation of large numbers of pattern-recognition algorithms in C++ can be extremely difficult. Algorithms must work as intended, be easy to maintain/extend and have tight control of memory management. The Pandora SDK addresses these issues directly: it provides a sophisticated Event Data Model and performs all event memory- management. Access to event objects and modification of these objects can only occur via algorithms requesting services provided by the Pandora SDK. A key remaining challenge is to ensure algorithms are efficient and scale kindly with the number of input objects in an event. This is a matter for the algorithm author, rather than the framework, but the Pandora SDK provides a number of constructs to help address performance. These include KD-trees, which provide log(n) look-up of e.g. hits within a search-volume around a specified space-point. There is a cost associated with constructing KD-trees, but the reduction in e.g. hit-hit permutations can be enormous.

5.5 Future Plans

Continued development of Pandora SDK and pattern-recognition algorithms, plus provision of support to users of Pandora. On-going Linear Collider work (including work performed by new PhD students in Cambridge) includes improvement of π^0 reconstruction and efforts to further improve the ability to identify and separate neutral hadrons from nearby charged hadrons. Detector optimisation studies will continue and will include full examination of performance of Pandora algorithms with digital and semi-digital HCAL detector models.

Figure 1: Jet energy resolution as a function of jet energy, including a breakdown of the resolution into contributing confusion terms. Illustrates performance of Pandora algorithms for ILD_o1_v06 with Silicon (Si) or Scintillator (Sc) as ECAL active material.

Figure 2: Jet energy resolution as a function of the ECAL cell size, for 250 GeV jets in ILD_o1_v06. As expected, the photon confusion term (ability to separate photons from nearby hadrons) drives performance changes.

Figure 3: Jet energy resolution as a function of the number of layers in the HCAL, for a range of different energy jets in ILD_o1_v06.

5.6 Applications Outside of Linear Colliders

The Pandora SDK has been designed to aid development of pattern-recognition algorithms in generic fine-granularity detectors. As such, its use is not limited to the ILC. Pandora algorithms now provide a successful particle flow reconstruction in an upgrade model of the CMS detector, even in dense pile-up conditions. Algorithms have also been developed for reconstruction of cosmic ray and neutrino-induced events in liquid argon time projection chambers, having significant impact in the neutrino-physics community.

R&D Technology	Participating Institutes	Description / Concept	Milestones	Future Activities

References

[1] DD4hep webpage, http://aidasoft.web.cern.ch/DD4hep.

[2] Markus Frank, F. Gaede, C. Grefe, and P. Mato. DD4hep: A Detector Description Toolkit for High Energy Physics Experiments. *J. Phys. Conf. Ser.*, 513:022010, 2014.

[3] Markus Frank, F. Gaede, N. Nikiforou, M. Petric, and A. Sailer. DDG4: A Simulation framework based on DD4hep and Geant4. *talk given at CHEP 2015*.

[4] F. Gaede. Marlin and LCCD: Software tools for the ILC. *Nucl.Instrum.Meth.*, A559:177–180, 2006.