LCIO
A persistency framework for LC detector simulation studies
Frank Gaede, DESY, IT
4th ECFA/DESY LC Workshop
Amsterdam April 1st-4th 2003

Outline

- Introduction
- Data model (brief, see talk in detector performance session)
- Design and Implementation
- Status
- Summary
Introduction

- At Prague workshop decided to have
  
  **Data format/persistency task force:**
  
  “Define an abstract object persistency layer and a data model for linear collider simulation studies until the Amsterdam workshop.”

- People:
  - Ties Behnke - DESY/SLAC
  - Frank Gaede - DESY
  - Norman Graf - SLAC
  - Tony Johnson - SLAC
  - Paulo Mora de Freitas - IN2P3

Motivation

**LCIO Persistency Framework**

- Generator
- Simulation
- Reconstruction
- Analysis

geometry
The Persistency Framework

LCIO

- data model
  - contents
- data access
  - API
  - implementation
- data format
  - persistency

Meetings

- Meeting at SLAC 12/09-12/13/02:
  (T. Behnke, F. Gaede, N. Graf, T. Johnson)
  - agreement to have common persistency framework in one US group
    (hep.lcd) and in the European group: LCIO
  - agreement on the (first) implementation format
  - first definition of the data model
- Meeting at Ecole Polytechnique 01/14-01/15/03:
  (F. Gaede, P. Mora de Freitas, H. Videau, J.-C. Brient)
  - agreement to use LCIO as the output format for the Mokka simulation framework
  - further discussions and refinement of the data model (reconstruction)
- Presentation and discussion of the data model at CERN miniworkshop of detector performance group 25/02/03
- Several phone meetings
LCIO requirements

- need Java, C++ and f77 (!) implementation
- extendable data model for current and future simulation studies
- user code separated from concrete data format
  - -> want to be flexible for future decisions on persistency
- three major use cases
  - writing data (simulation)
  - reading and updating data (reconstruction)
  - read only access to data (analysis)
- needed a.s.a.p.  -> keep it simple!

A brief Look at the Data Model:

RunHeader → Event → ReconstructedObject

SimHeader

RecoHeader

SIM

CalorimeterHit → TrackerHit → Track → Cluster → Reco

MCParticle

for details see talk in detector performance session
API – simulation data

- Interface for:
  a) writing data (simulation)
  b) read only access (analysis)

untyped collections

user extensions
data entities
tagging interface

API & implementation

abstract event

abstract io

concrete classes

persistance implementation
API definition for Java and C++

- use AID Abstract Interface Definition
  - tool from freehep.org
  - used successfully in the AIDA project
- define interfaces in Java-like language with C++ extensions
  - generates files with Java interfaces
  - generates C++ header files with pure abstract base classes
  - use javadoc for documentation
- independent implementations in Java and C++
  - keep Java “pure” i.e. machine independent

LCIO Fortran interface

- Fortran support for
  - legacy software (e.g. BRAHMS reconstruction)
  - non OO-analyses code (“old guys”)
- not a third implementation of the library – use C++-wrapper functions and cfortran.h instead:
  - one function for every class member function
  - use integers to store pointers!
  - OO-like code in fortran
LCIO f77 example:

```
5  --- event loop -------
10  do 10
20    event = lcr3rdReadNextEvent ( reader )
30    if ( event.eq.0 ) goto 11
40         runnum = lcr3rdGetEventNumber ( event )
50         evtnum = lcr3rdGetEventNumber ( event )
60         detname = lcr3rdGetDetectorName ( event )
70  write(*,*) 'run:', runnum
80  write(*,*) 'evtnum:', evtnum
90  write(*,*) 'detname:', detname
100     continue
110  continue
120  c  --- end event loop -------

--- anajob.f (Fortran)---L45---599---
```

Persistency Implementation

- use SIO: Simple Input Output
- developed at SLAC for NLC simulation
- already used in hep.lcd framework
- features:
  - on the fly data compression
  - some OO capabilities, e.g. pointers
  - C++ and Java implementation available
Status of LCIO

- C++ implementation available
  - integrated into Mokka simulation framework now
    (latest release mokka-01-05 writes TrackerHits in LCIO)
- f77 prototype
  - demonstrating the design
- Java implementation development ongoing
- complete integration into simulation software chains in the next months:
  - US: hep.lcd (Java)
  - Europe: Mokka (C++)/BRAHMS-reco(f77)

LCIO - To Do

- consolidate the API
- make the implementation more robust
  - error handling
  - debugging
- implement reconstruction data model
- add fast skipping mechanism
- get user feedback
  - -> lots of improvements …
Summary

- LCIO is a persistency framework for linear collider simulation software
- Java, C++ and f77 user interface
- LCIO is currently implemented in simulation frameworks:
  - hep.lcd
  - Mokka/BRAHMS-reco
- -> other groups are invited to join
- see LCIO homepage for more details:

Intermediate Software Chain

- gen. evts (STDHEP)
- Geant4/C++ Simulation Mokka
- Geant3/f77 Reconstruction Brahms
- zio-files (brahmsio)
- GEOM (SQL)
- GEOM f77
- Events (Jas, Root, ...)

Additional output format for Mokka, compatible with latest version of Brahms Reco.