LCIO
Persistency and Data Model for ILC Software

Frank Gaede, DESY
SLAC Simulation Meeting 2005
March 16-17 2005
Outline

- Introduction
- Implementation/Design
- Data Model
- Status
- Summary
Motivation for LCIO

Generator

Simulation

Reconstruction

Analysis

LCIO Persistency Framework

Java, C++, Fortran

Geant3, Geant4

geometry
LCIO Persistency Framework

LCIO

data model

contents

data handling

API

implementation

data format

persistency
LCIO Project Overview

- DESY and SLAC joined project:
  - provide common basis for ILC software

- Requirements:
  - need Java, C++ and f77 (!) API
  - extendable data model for current and future simulation and testbeam studies
  - user code separated from concrete data format
  - easy to adapt LCIO in existing applications
  - no dependency on other frameworks

- keep it simple & lightweight
LCIO SW-Architecture

- JAS/AIDA
- root
- hbook

- common API
  - generated from one source using AID

- Java API
- C++ API
- f77 API

- LCIO Java implementation
- LCIO C++ implementation

- *.slcio files (SIO)

  - compressed records, pointer retrieval
Data format (persistency): **SIO**

- **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 😊
- no direct access 😞
  
  -> use fast skip 😊
Implementation Details

- common Java, C++ interface:
  - AID-tool from freehep.org
  - define interfaces in Java-like language with C++ extensions
  - -> generates files with Java interfaces
  - -> generates C++ header files with pure abstract base classes

- Fortran interface:
  - use C++-wrapper functions and cfortran.h
  - one function for every class member function
  - use integers to store pointers !
  - -> OO-like code in fortran
LCIO class design
Data model 1

Run and Event

LCEvent
- Event number
- Collections

LCCollection
- Type
- Name
- Elements

LCOBJECT

Reconstructed Particle

MCParticle

SimCalorimeterHit

CalorimeterHit

SimTrackerHit

TrackerHit

TPCHit

Track

User extensions

LCFloatVec

LCIntVec

LCGenericObject

The LCEvent serves as a container of named collections of the various data types in LCIO (LCOBJECT subclasses)
Data model III

Monte Carlo

**MCParticle**
- Kinematics (4Vector)
- Parents/Daughters
- Generator Status
- PID
- Vertex
- ....
- -> all of HEPEVT
- + Simulator Status
- + Endpoint

**SimCalorimeterHit**
- CellID
- Energy/Amplitude
- Position (opt.)
- MCParticle Contributions

**SimTrackerHit**
- Position
- dEdx
- MCParticle Contribution
Data model IV

RawData and Digitization

RawCalorimeterHit
- CellID
- Amplitude
- Time (optional)

CalorimeterHit
- CellID
- Energy
- Time (opt.)
- Position (opt.)

TrackerHit
- Position
- Covariance
- dEdx
- raw hit

Serve as interface classes to reconstruction

TPCHit
- CellID
- Time
- Charge

Add more specialized classes as needed -> user request!

SimHits
Data model V

Reconstruction & Analysis

Cluster
- Energy
- Position
- Direction
- Shape
- Hits
- Clusters

Track
- (d0, φ, ω, z0, tanλ)
- dEdx
- Errors
- Chi2
- Hits
- Tracks

ReconstructedParticle
- Kinematics (4Vector)
- Charge, Mass, ...
- Reference point
- errors
- ParticleIds
- Clusters
- Tracks
- ReconstructedParticles

ReconstructedParticles can be simple particles and compound objects like jets, vertices,...

ParticleID
- Type/PDG
- Likelihood
- AlgorithmID
- Parameters
The LCEvent can be used as container for transient data in an application, e.g. reconstruction.

Application will call list of modules that read existing collections from the LCEvent and add resulting new Collections.

LCIO has (Event/Run)-Listener classes that can serve as base classes for modules.

Easy to define an application framework based on LCIO for reconstruction and analysis:

- org.lcsim (Java)
- Marlin (C++)

(see talks)
LCIO in a modular application

MyInput2.slcio
MyInput1.slcio
MyInput0.slcio

LCEvent

read and add collections

collection0

ModuleN

MyOutput.slcio

MyInput2.slcio
MyInput1.slcio
MyInput0.slcio

Module0

Module1

Module2

...
LCIO Releases

- first official release v01-00 (Nov 2003)
- simulation data model
- release v01-03 (Sep 2004)
  - reconstruction model (Rec.Particle,Tracks,Cluster)
  - user extensions (LCGenericObject)
- raw data classes
- support for CLHEP four vectors
- transient data collections
- StdHep interface
- run, event and collection parameters
LCIO Status I

- new release v01-04 (March 2005)
  - **C++**
    - `LCEvent::getTimeStamp()` now long64
    - `UTIL::LCTime` conversion between real date/time and time stamp
    - `UTIL::LCFixedObject` - convenient and efficient way to define `LCGenericObject` subclasses
    - `LCEvent::takeCollection( name )` (technical: need way to tell event that it is no longer the owner of the collection memory management)
  - handling of multiple events and I/O streams
    - for event mixing
    - already in Java
  - **Java**
    - support for `LCGenericObjects`
  - **f77**
    - removed deprecated method calls
LCIO Status II

- all
  - unique definition of 64bit time stamp in LCEvent:
    - ns since 1/1/1970 (UTC)
- subset collections
  - hold pointers/references to objects already existent in the event, e.g. LeptonCandidates from ReconstructedParticles
  - transient and persistent
    - if persistent, only pointers/references are stored in the file
    - files are downward compatible, i.e. LCIO 1.3 can read new files (except subset collections)
- bug fixes
- improvements in API doc
LCIO Online documentation

Documentation for LCIO v01-03

- Users manual (also available as pdf and ps) Read before you get started
- Java API documentation
- C++ API documentation
- (printable version of the C++ API reference: lcrcrefman.ps)
- XML data format description, lcio.xml

Last modified: Thu Sep 23 14:51:51 2004

Building the library

A few variables have to be set depending on your development environment, e.g.

- Linux (and both):
  - export LCIO=/usr/local/lib
  - export PATH=/usr/local/lib
  - export LCIO=~/bin:
- Windows/Cygwin - DOS shell
  - set LCIO=~/bin:
  - set LCIO=~/bin:

IMPL::ReconstructedParticleImpl Class Reference

Implementation of ReconstructedParticle. More...

Inheritance diagram for IMPL::ReconstructedParticleImpl

EVENT::Context
EVENT::ReconstructedParticle
IMPL::ReconstructedParticleImpl

List of all members

Public Member Functions

ReconstructedParticleImpl() Default constructor, initializes values to 0.
LCIO on the web

- LCIO homepage: http://lcio.desy.de
  downloads and documentation

- LCIO forum at: http://forum.linearcollider.org
  user/developer questions and comments
  discussions on new developments

- LCIO bug reports at: http://bugs.freehep.org
  bug report and new feature requests
LCIO Users

- Geant4 Full simulations:
  - LCDG4, LCS, Mokka, Jupiter (under development)

- Reconstruction:
  - org.lcsim, Brahms, Marlin

- Fast simulation
  - Lelaps, Simdet

- Testbeam
  - Calice: Ecal, Hcal PPTs
  - TPC prototypes

- Analysis Tools
  - JAS3: file browser, code wizard
  - WIRED: generic event display (picking of LCIO objects)

-> has become de facto standard for ILC-software
JAS3 and LCIO

JAS3 provides native interfaces to LCIO: browser, code wizard, event display WIRED

Summary & Outlook

- LCIO is a persistency framework and data model for the linear collider
  - DESY and SLAC joined project
  - Provides Java, C++ and f77 API
  - New version v01-04 released now
    - subset collections
    - LCGenericObjects in Java
    - multiple I/O streams now also in C++
    - bug fixes ...

- Future plans:
  - react to user requests
  - provide convenient methods
  - handling of relationships
  - handling of meta data
  - bug fix patches

- Idea to develop a Java/C++ binding for LCIO
  -> would allow to have a mixed language reconstruction framework
  -> not so straight forward…