LCIO
Overview and Status

ECFA Workshop, Montpellier
November 14, 2003
Frank Gaede  DESY  -IT-

Outline

• Introduction
• Data model
• Software design
• Implementation
• Status/Features
• Customers/Users
• 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.”

- -> **LCIO** – Linear Collider Input/Output
  - DESY/SLAC/LLR joined project
  - design of data model and software introduced at Amsterdam
    workshop
  - now production version

Motivation for LCIO
ECFA Workshop, Montpellier - Nov 14, 2003
Frank Gaede, DESY -IT-

LCIO persistency framework

LCIO

data model contents

data handling API implementation

data format persistency

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
• needed a.s.a.p.
  • -> keep it simple (lightweight)
**LCIO Datamodel**

- **LCRunHeader**
- **LCEvent**
- **MCParticle**
- **ReconstructedParticle**
- **SimTrackerHit**
- **Track**
- **Cluster**
- **CalorimeterHit**
- **SimCalorimeterHit**
- **SIM/Real**
- **TPCHit**
- **LCIntVec**
- **FloatVec**

**Data model - Design**

- **user extensions, e.g. calibration constants**
- **simulation output**
- **Prototype Data**
Implementation - Design

LCIO SW-Architecture

- JAS/AIDA
- root
- hbook

common API

Java API
C++ API
f77 API

LCIO Java implementation
LCIO C++ implementation

*.slcio files (SIO)

compressed records, pointer retrieval

generated from one source using AID
C++ and f77 example code

```cpp
// event loop
const LCEvent* event;
while ((event = ldr->readNextEvent()) != 0) {
    int runNum = event->getRunNumber();
    std::cout << runNum << std::endl;
    std::cout << event->getEventName() << std::endl;
    Continue;
} // event loop
```

LCIO Status

- production version 1.0 released:
  - C++, Java, f77 complete for simulation data
  - and generator data (HEPEvt<->LCIO)
  - simple example code for all languages
  - 'real world' examples (JAS3, root, hbook)
  - documentation
    - users manual
    - API documentation HTML (javadoc, doxygen)
  - available for download via CVS
    - linux (gcc), windows (cygwin)
  - schema evolution from now on (reading old files)
  - API stable (only extensions)
Javadoc example

Doxygen example
LCIO Customers/Users

- Mokka simulation (v08-00)
  - update to v01-00 within ~1 week
- Brahms reconstruction
  - under development (H.Vogt)
    -> to be release in ~2 weeks
- JAS3
  - provides convenient file browser
  - will have LCIO-WIRED plugin
    -> generic event display!
- Calorimeter group (DESY)
  - will convert MiniCal raw data to LCIO files
  - to be used also for HCalPPT
- TPC groups (DESY & LBNL?)
  - will use LCIO for prototype
- other groups looking into using LCIO

JAS3 – LCIO file browser

Future developments

- implement reconstruction data model
- add convenient methods
  - looping over MCParticles
  - analyzing parent/daughter relationships
  - ...
- add possibility to store more generic user data (calibration constants etc.)
- respond to user requests

Towards a simulation framework

Persistency

Generators

Mokka

OO-Reco

LCIO

Geometry

Analyses

JAS3, root, (PAW)

Simulations

Digitization

Reconstruction

Generators

Mokka

OO-Reco

Geometry

Persistency


calibration constants etc.)

respond to user requests
Summary

- LCIO, a persistency framework for the LC:
  - Java, C++ and f77 user interface
  - Java and C++ implementation
  - data model for simulation and prototype data
    - reconstruction soon to follow
- production version released (1.0)
- used by several groups and tools
  - others invited to join!
- see LCIO homepage for more: