Status of the data format/ persistency task force

Intermediate Miniworkshop of the
Overall Detector Performance Group
CERN 02/25/2003

Frank Gaede DESY -IT-

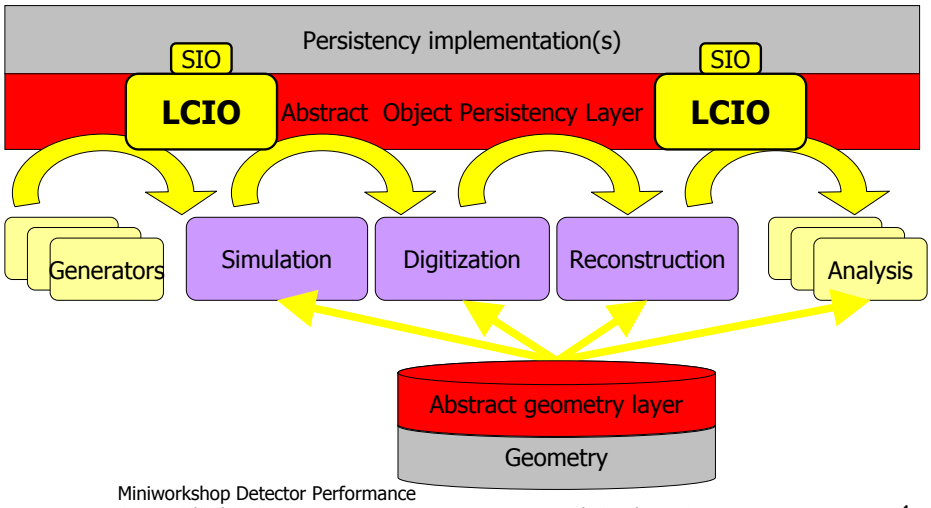
Outline

- Introduction
- LCIO - the persistency framework
  - Software design
  - Implementation
  - Status
- Data Model
  - Simulation (hits)
  - Reconstruction (clusters, tracks, eflow)
- Summary/Outlook

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. Maybe have a first implementation by then?"
- People:
  - Ties Behnke (DESY/SLAC)
  - Frank Gaede (DESY)
  - Norman Graf (SLAC)
  - Tony Johnson (SLAC)
  - Paulo Mora de Freitas (LLR/in2p3)

The big picture – where are we?



Meetings

- Meeting at SLAC 12/09-12/13:

  (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:

  (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)
- Several phone meetings

LCIO – Basic Requirements

- abstract object layer – user code should be independent of concrete implementation
- Java, C++ and f77(!) version needed
- common interface in Java and C++
- machine independence of files
- data model should be extendable by users
- fast skipping mechanism needed
- direct access nice to have
- Keep it simple !

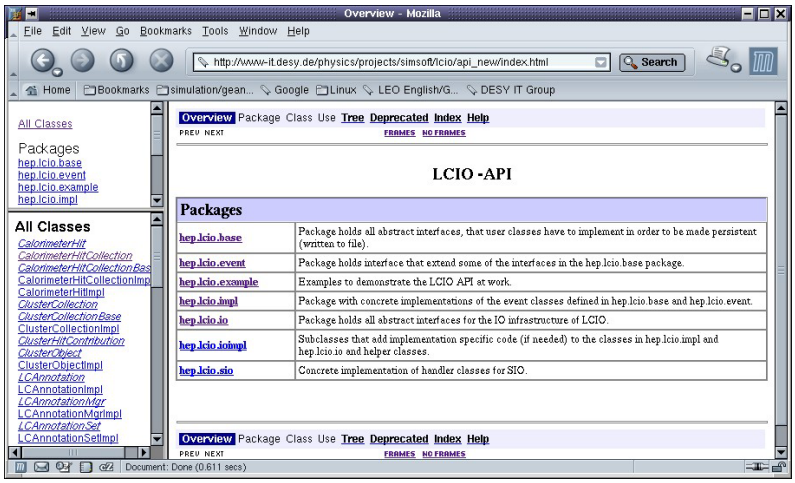
LCIO – technical design

- use SIO (Simple Input Output) as first implementation:
  - already used by hep.lcd group
  - simple, machine independent and available
- use AID (Abstract Interface Definition):
  - generates Java and C++ code from abstract files
  - used successfully within the AIDA project
  - need only to worry about f77…
- separate Java and C++ implementations:
  - keep Java pure (machine independent)
- pure OOAD approach:
  - implement f77 as wrapper functions to C++ implementation

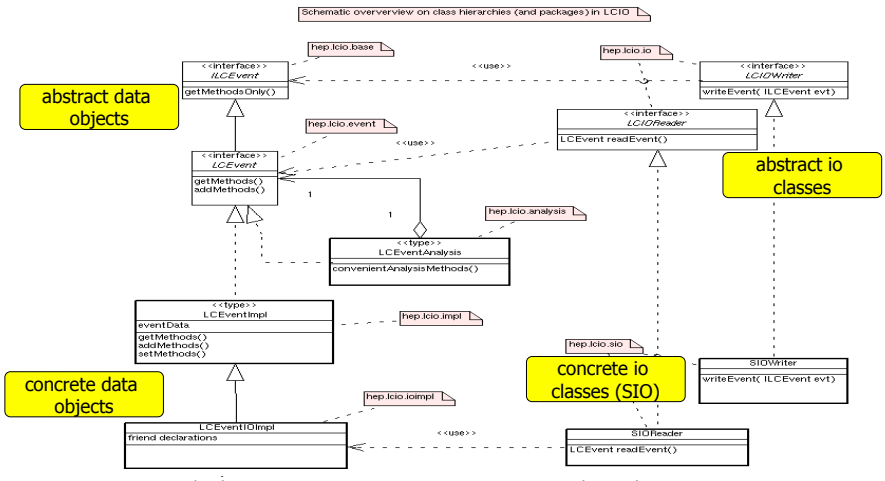
LCIO – class design

- basic interfaces that user classes have to implement in order to make objects persistent (read only)
- abstract interface to LCIO (no knowledge of SIO)
- concrete implementations of all classes:
  - that user code might use for writing data
  - are used for reading but not exposed to the users (prevent accidental data corruption)
- additional interfaces for analysis code (parameter transformations etc.) as decorators
- see LCIO webpage for details:
  http://www-it.desy.de/physics/projects/simsoft/lcio/index.html

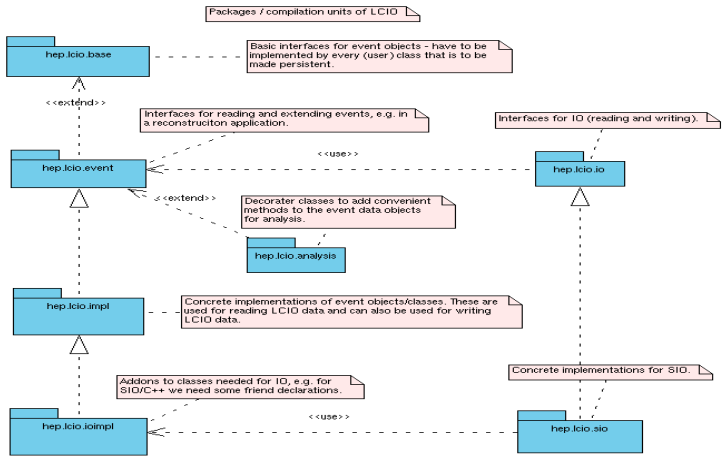
LCIO -API documentation



LCIO – class design example



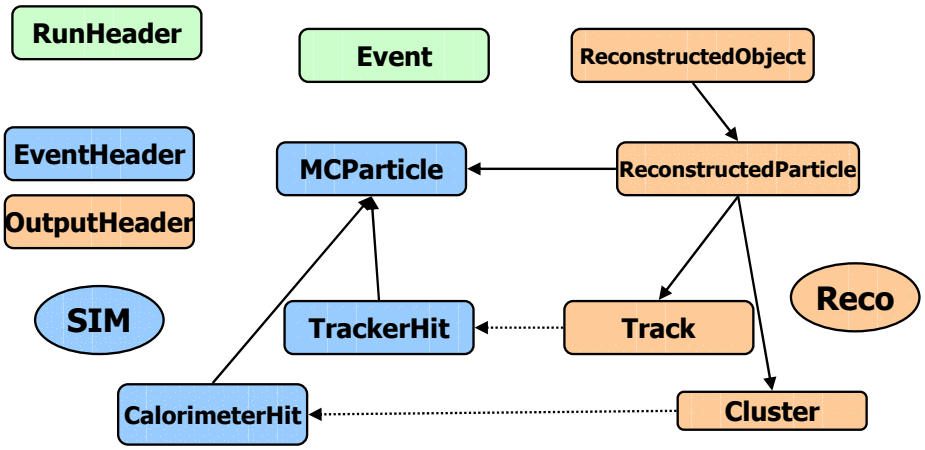
LCIO – package layout



LCIO - Status

- first version of the API ready (simulation only)
  - (slight) modifications during implementation likely
- C++ - prototype to demonstrate the API and the class design
- f77-C++ wrapper prototype
  - using integers as pointers to reflect OO-style
- cvs repository set up
- start implementing now !
- first version of data model:
  - simulation – OK
  - reconstruction – OK/further iterations needed?

Data model – Overview



Data model - LCRunHeader

- block: RunHeader
  - int: runNumber
  - string: detectorName
  - string: description
  - string[]: activeSubdetectors

=> describes the run setup

Data model - LCEventHeader

- EventHeader
  - int: runNumber
  - string: detectorName
  - String[]: subdetectorName
  - blockNames:
    - string: blockName
    - string: blockType

=> describes event data –
needed for fast skip

Data model – LCEvent (sim)

- Event
  - int: runNumber
  - string: detectorName
  - String[]: subdetectorName
  - long: timeStamp

Data model – LCEvent (sim)

- MCParticle
  - pntr: parent
  - pntr: secondparent
  - pntr[]: daughters
  - int : pdgid:
  - int : hepevtStatus
    (0,1,2,3 HepEvt)
    (201, 202 sim. decay)
- MCParticle cont.
  - double[3]: start
    (production vertex)
  - float[3] : momentum
    (at vertex)
  - float: energy
  - float: charge

Data model - LCEvent (sim)

- TrackerHit
- string: subdetector
  - int: hitFlags (detector specific: Id, key, etc.)
  - double[3]: position
  - float: dEdx
  - float: time
  - pntr: MCParticle

Data model - LCEvent (sim)

- CalorimeterHit
- string: subdetector
  - int: cellId0
  - int: cellId1
  - float: energy
  - double[3]: position – optional (file size!)
  - particle contributions:
    - pntr: MCParticle
    - float: energyContribution
    - float: time
    - int: PDG (of secondary) - optional

Data model - LCEvent (reco)

- OutputHeader
  - int: isrFlag
  - float: colliderEnergy
  - int: flag0 (to be defined)
  - int: flag1 (to be defined)
  - int: reconstructionProgramTag
  - float: Bfield

-> could be combined with global header…

Data model - LCEvent (reco)

- Track
  - int: tracktype (full reconstr, TPC only, Muon only, etc.)
  - float: momentum
  - float: theta
  - float: phi
  - float: charge
  - float: d0 (Impact Parameter in r-phi)
  - float: z0 (Impact Parameter in r-z)
  - float: reference point (x, y, z)
  - float[15]: covmatrix
  - float[10?]: dEdx (weights and probabilities)
  - TrackerHits: - optional
    - pntr: TrackerHit

Data model - LCEvent (reco)

- Cluster
  - int: detector (type of cluster: ECAL, HCAL, combined…)
  - int: clustertype (neutral, charged, undefined cluster)
  - float: energy
  - float[3]: position (center of cluster x, y, z)
  - float[6]: errpos (cov. matrix of position)
  - float: theta (intrinsic direction: theta at position)
  - float: phi (intrinsic direction: phi at position)
  - float[3]: errdir (cov. matrix of direction)
  - float[3-9?]: shapeParameters
  - float[3]: weights (compatible with em., had., muon)
  - CalorimeterHits: - optional
    - pntr: CalorimeterHit
    - float: contribution

Data model - LCEvent (reco)

- ReconstructedParticle
  - int: primaryFlag (0: secondary, 1: primary)
  - int: ObjectType (charged/ neutral particle)
  - float[3]: 3-Vec (px, py, pz)
  - float: energy
  - float[10]: covariance matrix
  - float: charge
  - float[3]: reference position for 4-vector
  - float[5]: PID_type (hypotheses for e, g,pi,K,p,...)
- ReconstructedParticle cont.
  - MCParticles:
    - pntr: MCParticle
    - float:weight
  - Tracks:
    - pntr: Track
    - float:weight
  - Clusters:
    - pntr: Cluster
    - float:weight

Data model - LCEvent (reco)

- ReconstructedObject
  - int: ObjectType (jet, vertex, ...)
  - float[5]: 4vec (4-vector of object (px, py, pz, E, M)
  - float[3]: reference (position)
  - float[15?]: covariance matrix
  - reconstructedParticle:
    - pntr: ReconstructedParticle
    - float: weight

  => generic reconstructed objects, linked to reconstructed particles

Summary/Outlook

- LCIO: persistency framework for linear collider simulation studies:
  - design phase is over – time to implement
  - beta version ready by the Amsterdam workshop?
- Data model:
  - first version for sim and rec data
  - input from this workshop (for rec) ?
- near term plans:
  - use LCIO in Mokka and Brahms – others ?

in the meantime use Mokka/Brahms… (next slide)

Intermediate Software Chain