Towards LCIO 2.0

Improving the EDM and the I/O

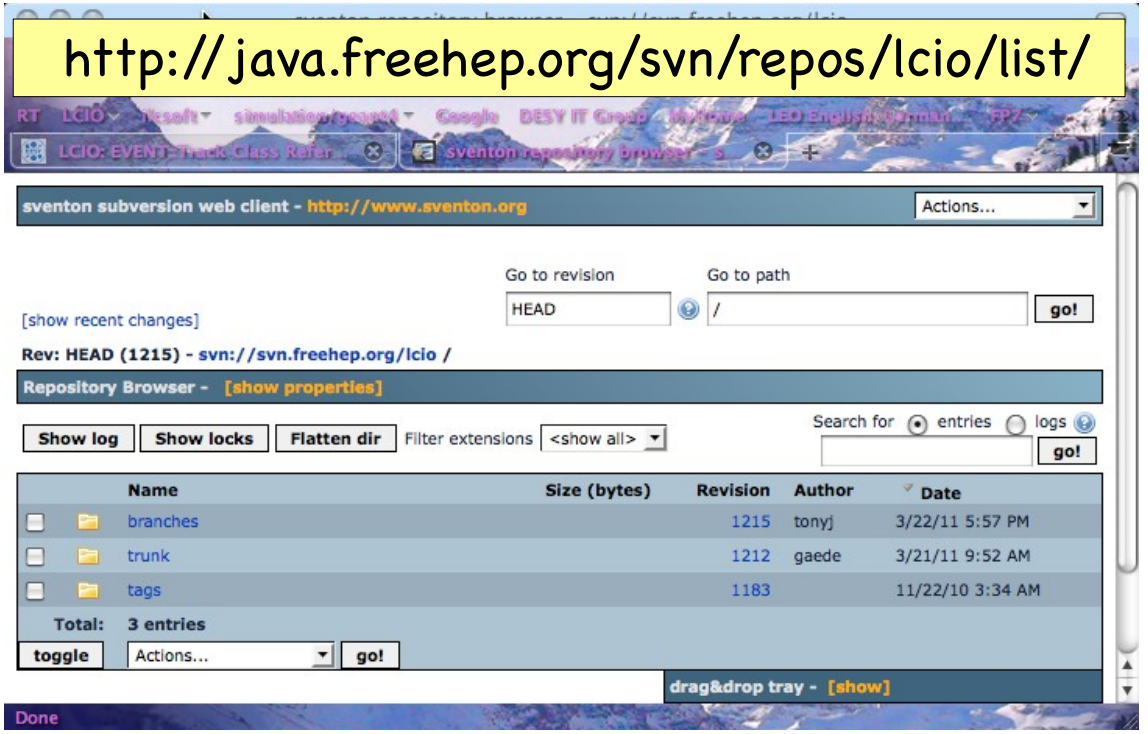
Frank Gaede, DESY

Software WG Phone meeting

DESY, March 30th, 2011

LCIO repository moved to SVN

svn webinterface:



checkout released versions:

svn co svn://svn.freehep.org/lcio/tags/v01-51-02 v01-51-02

checkout HEAD version:

svn co svn://svn.freehep.org/lcio/trunk trunk

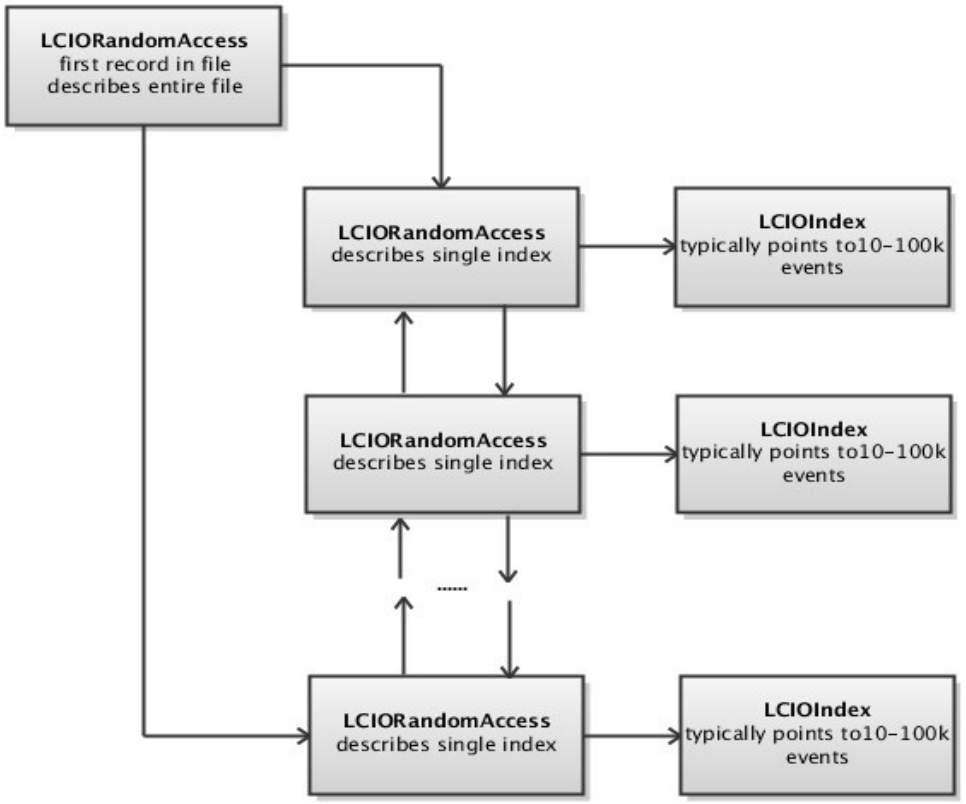
old CVS still works for checkout of released versions !

LCIO 2.0 - new features

- LCIO 2.0 (AKNA LCIOv2) is planned for some time now
- goal is to improve LCIO while still being backward compatible
- planned/requested features:
- **direct access to events -> Done**
- **partial reading of events -> ?**
- **splitting of events over files -> ?**
- **storing of (arbitrary) user classes -> currently not planned**
- **simplify using LCIO with ROOT -> Done**
  - (ROOT macros, TTreeViewer, I/O (?) ,...)
- **improving the event data model -> Under Way (this talk)**
  - (1d,2d hits, tracks/trajectories)

direct access to LCIO events

- direct access to LCIO events:
  - overlay of random background events
  - physics analysis - reading of pre-selection
  - previously only via fast skip or creation of TOC on opening (slow)
  - → introduced two additional records **LCIORandomAccess/LCIOIndex**



- records written at end of file on close()
- can append to file
- **can add direct access to existing file**
  - if opened in append mode on writable file system (not tape)

- released in v01-51

a ROOT dictionary for LCIO

- LCIO now comes with a ROOT dictionary for all LCIO classes (optional) - with this one can:
  - use LCIO classes in ROOT macros
  - write simple ROOT trees, e.g. std::vector<MCParticleImpl*>
  - use TTreeDraw for quick interactive analysis of LCObjects:

```
//---gamma conversions:
TCut isPhoton("MCParticlesSkimmed.getPDG()==22" ) ;
LCIO->Draw("MCParticlesSkimmed._endpoint[][0]:
            MCParticlesSkimmed._endpoint[][1]",isPhoton ) ;
```

  - write complete LCIO events in one ROOT branch
  - see: $LCIO/examples/cpp/rootDict/README for details & help

- -> we are interested in feedback from the users if this provides already the requested features

partial reading & splitting of events

- would be needed for performance and cost (disk space) :
  - read only objects of interest in analysis (PandoraPFOs)
  - store simulation and reconstruction output in separate files
- main obstacle: need pointer/reference mechanism across I/O records and files, e.g. with index based pointers independent of I/O, e.g.:
  - long64 index = HASH(collName) << 32 | collIndex

- non trivial change to LCIO/SIO
- might not have the manpower to do it now
- -> could have 'workaround' for ILD DBD production: split SIM and REC files and have dedicated processor for merging:
  - tradeoff between usability and disk space
  - (need to keep SIM and REC files in synch and always read both...)
  - SIM: ~1MByte/had. evt -> 1TByte / 1M had. evts

plans for next release

as discussed at

LCIO developers meeting after ALCPG

(N.Graf, T.Johnson, S.Aplin, F.G.)

cleanup of build systems

- C++
  - remove old Makefiles - have CMake only
- Java
  - remove old ant scripts
  - have Maven only
  - -> include Maven in release
    - no dependency for C++
  - -> Maven plugin for creating header files only once
    - interesting for developers - (no rebuild after install)

extensions of MCParticle

- add spin information:
  - float[3] getSpin()
- add color flow information
  - int[2] getColorFlow()
    - are these pointers to other MCParticles (indices) ?
- -> both copied from stdhep/HepEvt4 as written by Whizzard
- user request:
  - have simProcessId for particles that decayed in simulator
  - -> will use lower 16 bits of SimStatus word + collection parameters: SimProcessID, SimProcessName
  - short getSimProcessID()
    - need to define details of processIDs
    - implement this in Mokka and SLIC the same way

Meta data in event

- MC truth information from generator:
  - processID
  - processName
  - alphaQCD
  - alphaQED
- store as collection parameters in LCEvent
  - to be implemented in StdHepReader and Mokka/SLIC
- will be documented at:

  https://confluence.slac.stanford.edu/display/ilc/LCIO
- other meta information needed ?

Track – multiple track states

- agreed to store multiple track states for Track:
  - @IP, first hit, last hit, face of calorimeter, others ?
- will introduce TrackState object and
  - TrackStateVec& getTrackStates()
  - TrackState will have:

virtual float	**getD0** () const =0 *Impact paramter of the track in (r-phi).*
virtual float	**getPhi** () const =0 *Phi of the track at the reference point.*
virtual float	**getOmega** () const =0 *Omega is the signed curvature of the track in [1/mm].*
virtual float	**getZ0** () const =0 *Impact paramter of the track in (r-z).*
virtual float	**getTanLambda** () const =0 *Lambda is the dip angle of the track in r-z at the reference point.*
virtual const **FloatVec** &	**getCovMatrix** () const =0 *Covariance matrix of the track parameters.*
virtual const float *	**getReferencePoint** () const =0 *Reference point of the track parameters.*
~~virtual bool~~	~~**isReferencePointPCA** () const =0~~ *True if the reference point is the point of closest approach.*

original functions getX()
of Track will return:

trk.getTrackStates()[0].getX()

Track – multiple track states

- isReferecePointPCA() will be dropped !
  - all track states should have the reference point chosen such that this is where the track is defined, i.e. at the closest hit, the IP or the face of the calorimeter (in this case, d0 and z0 are typically 0)
- introduce:
  - int TrackState::getLocation()
  - with defined constants/enums:

    AtIP, AtfFirstHit, AtLasthit, AtCalorimeter, AtVertex, Other
- convention that TrackStates are ordered wrt. path length s – as seen from the IP !
- add convenient method:
  - TrackState* getClosestTrackState(float x, float y, float z)
  - TrackState* getTrackState(int location)
    - returns track state for location with given enum – or NULL
- first and last hit position available through
  - trk->getTrackState(AtFirstHit / AtLastHit)->getReferencePoint()

Tracker-and CalorimeterHit

- canonical way of accessing layer number:
  - local to sub detector (inside-out, starting from 0)
  - getLayerNumber(), setLayerNumber()
    - filled from cellIDs after reading, write to cellID
  - need convention: string "layer" in CellIDEncoding
  - if "layer" not present - layerNum = -1 (deal with this in Marlin/org.lcsim)
  - will update SLIC and Mokka accordingly
- add cellIDs to TrackerHit:
  - getCellID0(), getCellID1() (-> same as in CalorimeterHit)
  - use cellID for consistency w/ CaloHit - even though there are no cells
  - drop old 'type' word and replace getType() with access to cellID["type"]
- question: convention for subdetectorIDs in cellIDs ?
  - -> this will probably have to be done on a per concept (detector) basis
  - -> need convention for ILD for DBD reconstruction

additional extensions

- to Cluster add
- float getEnergyError()

- to SimCalorimeterHit optionally add the position where the energy deposition (step) occurred:
- float[3] getStepPosition(int i)
  - only if flag LCIO.CHBIT_STEP==1
    - useful for detailed simulation studies of edge effects in calorimeter cells or MAPS digitization

- any other requests ?

1d and 2d hits

- agreed to introduce six new TrackerHit classes
  - PlanarDisk1D
  - Planar1D
  - Cylindrical1D
  - PlanarDisk2D
  - Planar2D
  - Cylindrical2D
  - have u, du, pos1, pos2 (strip begin end) for 1D
  - have u, v + cov(u,v) + cylinder/plane parameters for 2D
- details currently defined (N.G.)
- probably these will also implement TrackerHit interface (x,y,z, cov) for backward compatibility
- this will make it possible to properly take Si-strip detectors into account in the tracking (if manpower allows)

discussed common DST format

- ILD DSTs:
  - ReconstrucedParticles (PFOs)
  - Tracks
  - Cluster
  - MCParticlesSkimmed
  - LCRelation PFOs <-> MCTruth
    - via Tracks and Clusters or directly
  - 2-6 JetCollections with Flavour tag
- SID interested in having a common definition of the DST format for the DBD
- -> could simplify collaboration on physics analyses

Summary

- made quite some progress towards LCIO 2.0
- already in v01-51:
  - direct access to events and ROOT dictionary
- now defined all the planned extensions/improvements to the EDM:
  - MCParticle: spin and color information
  - Tracks (multiple track states), TrackerHits 1D/2D
  - Tracker/CaloHits: getLayer()
  - ...
- hope to have beta release in a few weeks
- now is the time to make additional requests if needed for the DBD