LCIO 2.0

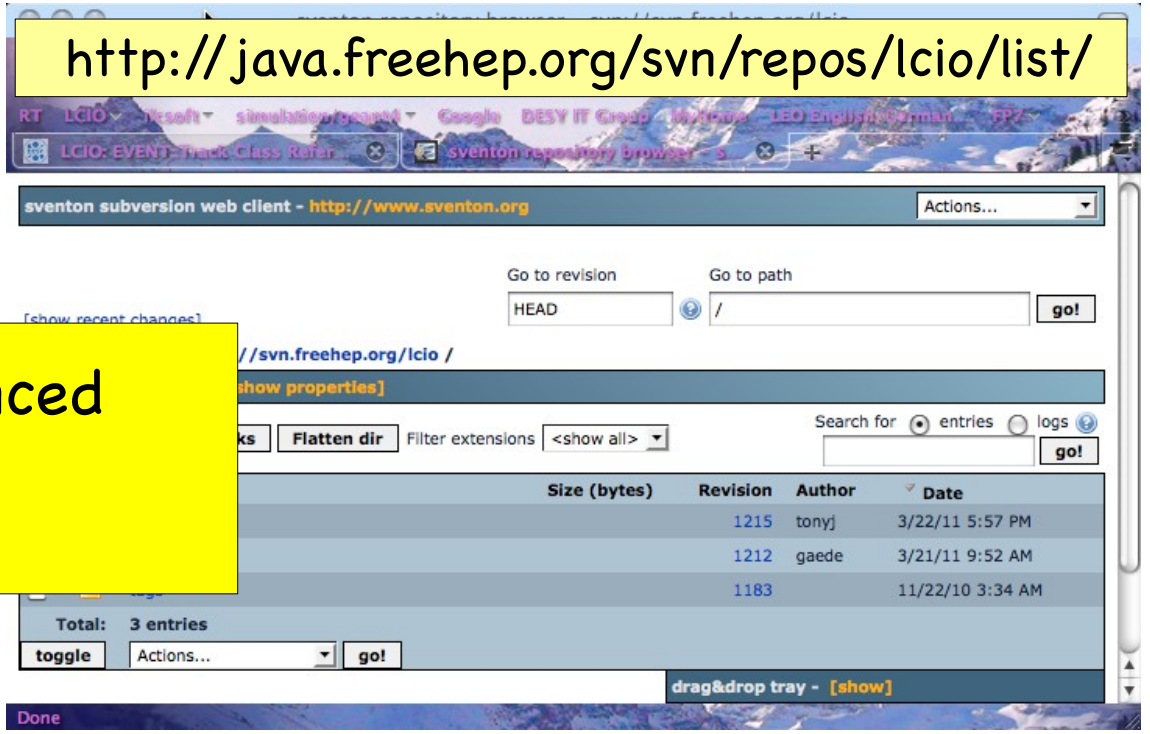
Status and open issues

Frank Gaede, DESY

Software Common Task Group Meeting, June 6, 2011

LCIO repository moved to SVN

svn webinterface:



need to be announced
- manual
- web page

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 -> postponed**
- **splitting of events over files -> postponed**
- **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 meeting**
  - (1d,2d hits, tracks/trajectories)

cleanup of build systems

- C++
  - remove old Makefiles – have CMake only

to be done

- 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)

to be done (?)

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

to be done

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 refere*
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 get**X**()
of Track will return:

trk.getTrackStates()[0].get**X**()

- first version done
branch: trackstate_dev

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

- first version done
branch: trackstate_dev
- convention that TrackStates are ordered wrt. path length s – as seen from the IP – to be done by the user !
- 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

to be done

additional extensions

- to Cluster add

to be done

- float getEnergyError()
- float getTime() <- new request from ILD/CLIC

- 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

needed asap for DHCAL !!

1d and 2d TrackerHits

- agreed to introduce new TrackerHit classes in Eugene
- we need this now for new ILD tracking code – propose to the following:
- **TrackerHitPlanar**
  - x, y, z – 'space point'
  - u(theta, phi) , v(theta, phi) – measurement directions (spanning vectors in the plane)
  - du, dv – measurement errors
  - -> to be used for 1d and 2d
- **TrackerHitCylindrical**
  - x, y, z – 'space point'
  - R, Xc, Yc – cylinder parameters (parallel to z)
  - dphi, dz – measurement errors
  - -> to be used for 1d and 2d
- these also need to implement the TrackerHit interface (x,y,z, cov) for backward compatibility and code reusability
- need TrackerHit::Type – either as attribute or convention to have it in CellID