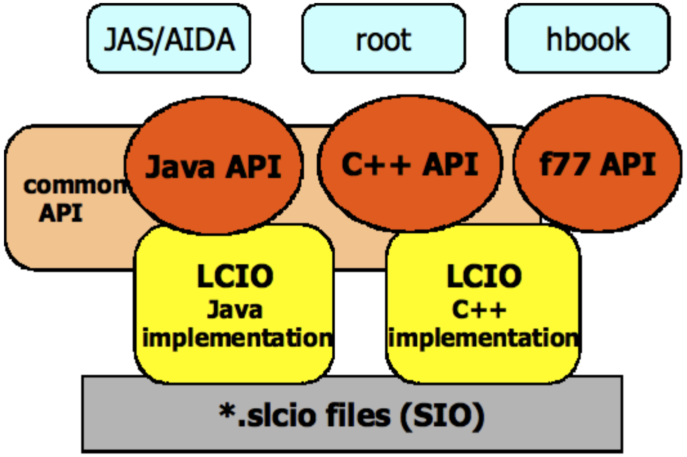
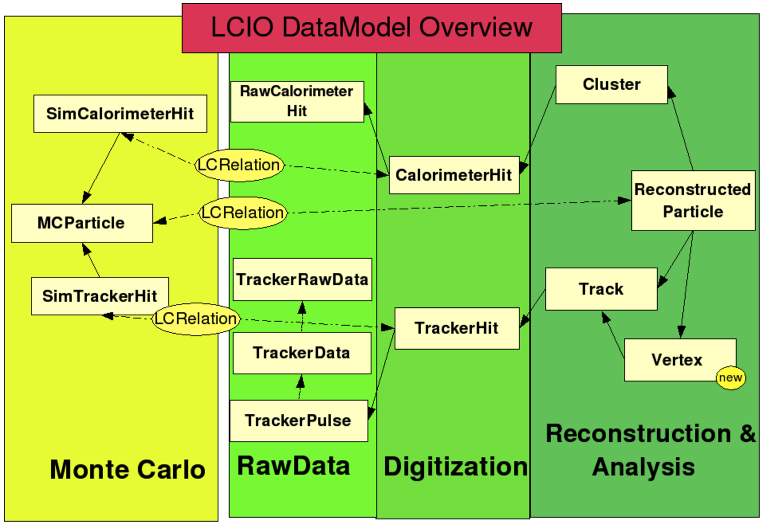
Frank Gaede, LC Software Meeting, Feb 8, 2012





LCIO 2.0

- LCIO provides a **hierarchical event data model** and a **persistency solution** for LC software
  - DESY/SLAC project since 2002
- C++ and Java API
  - also f77 (obsolete !?) and Python (experimental)
- used in ILD and SID SW frameworks and in many ILC **testbeam** experiments
- several 100TByte of simulated and real data stored in LCIO
- recently released LCIO v2 with many improvements

LCIO v02-00

- after LOI decided to have major new LCIO release "2.0"
- goal: improve usability of LCIO and address some short comings while being fully backward compatible
- planned/requested features:
- **simplify using LCIO with ROOT -> Done** (v01-12-01)
- **direct access to events -> Done** (v01-51)
- **improving the event data model -> Done** (v01-60, v02-00)
- partial reading of events -> postponed
- splitting of events over files -> postponed

could be addressed post DBD – depending on user requests

- LCIO v02-00 has been released:
- svn co svn://svn.freehep.org/lcio/tags/v02-00-03

LCIO v02-00 - new features/extensions

- moved to SVN code repository

  http://java.freehep.org/

  svn/repos/lcio/list
- browse code changes online
- added method to count events
  - LCReader::getNumberOfEvents()
  - tool: $LCIO/bin/lcio_event_counter
- added definitions specific to ILD to UTIL/ILDConf.h
  - -> allows to encode: subdetector, side, layer, module, sensor

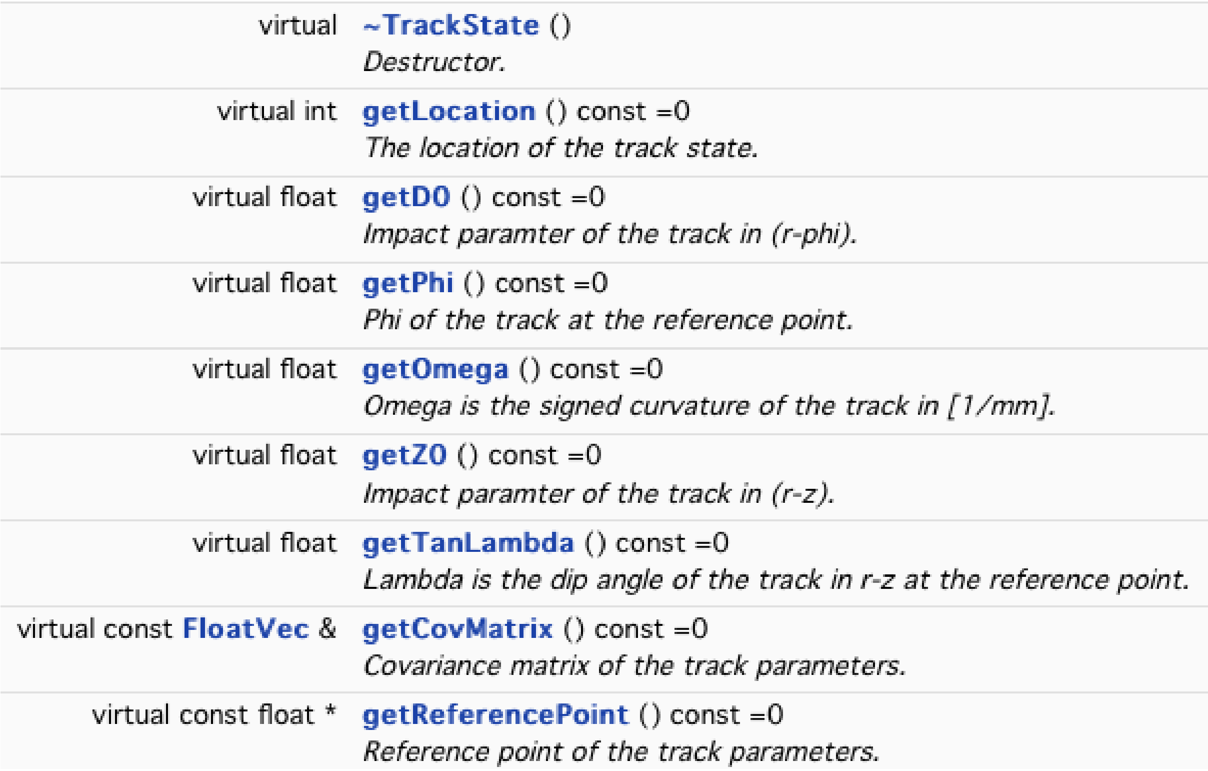
    in cellID0

- EDM extensions:
  - float[3] MCParticle::getSpin()
  - int[2] MCParticle::getColorFlow()
    - also written by Whizzard now
  - SimCalorimeterHit::getStepPosition(int i)
    - needed for SDHCAL digitization
  - Cluster::getEnergyError()
  - int (Sim)TrackerHit::getCellID0()
  - int (Sim)TrackerHit::getCellID1()
    - allows to encode details of the measurement module in the hits

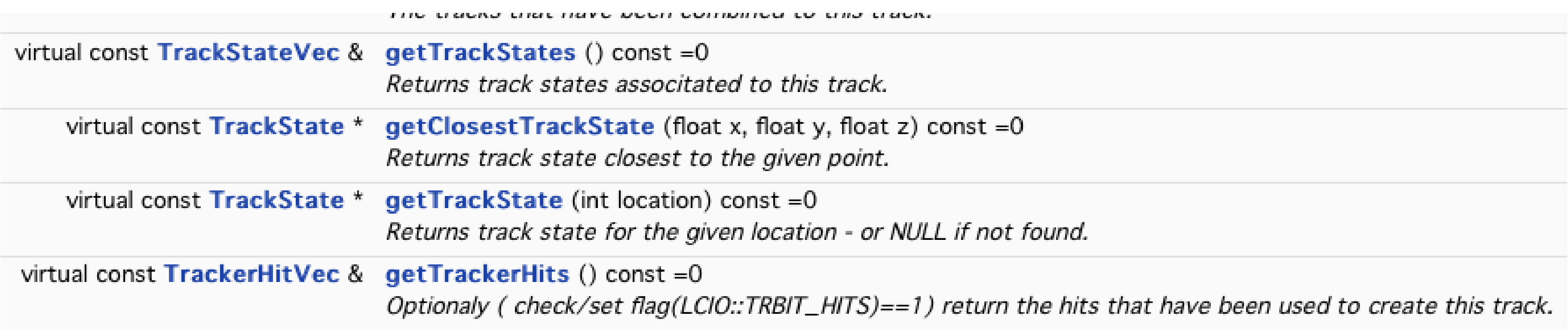
      -> needed for tracking package

LCIO v2 Track & Trackstates

- lcio Track now has **multiple TrackStates**
- will store four canonical TSs:
  - AtIP, AtFirstHit, AtLastHit, AtCalo
- TS returned either by
  - identifier
  - or closest to given point
- mostly backward compatible



virtual	**~TrackState** () *Destructor.*
virtual int	**getLocation** () const =0 *The location of the track state.*
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.*



The tracks that have been combined to this track.

virtual const **TrackStateVec** &	**getTrackStates** () const =0 *Returns track states assoicated to this track.*
virtual const **TrackState** *	**getClosestTrackState** (float x, float y, float z) const =0 *Returns track state closest to the given point.*
virtual const **TrackState** *	**getTrackState** (int location) const =0 *Returns track state for the given location - or NULL if not found.*
virtual const **TrackerHitVec** &	**getTrackerHits** () const =0 *Optionaly (check/set flag(LCIO::TRBIT_HITS)==1) return the hits that have been used to create this track.*

LCIOv2: 1d and 2d TrackerHits

- need new tracker hit classes to properly describe 1d and 2d measurements (pixels/TPC and **strips**)
- **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 (dv is strip length in 1d case)
- **TrackerHitCylindrical**
  - x, y, z - 'space point'
  - Xc, Yc – center of cylinder (parallel to z)
    - (cylinder radius: R = sqrt((x−x_c)^2 + (y-y_c)^2))
  - dphi, dz - measurement errors
  - -> to be used for 1d and 2d
- these also implement the TrackerHit interface (x,y,z, cov) for backward compatibility and code reusability (eg in event display)

a ROOT dictionary for LCIO

- LCIO comes with a ROOT dictionary for all LCIO classes - with this one can: (since v01-12-01)
  - 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 is a reasonable way to work with ROOT & LCIO
- other option: implement ROOT I/O for LCIO (.rlcio) !?