LCIOv2

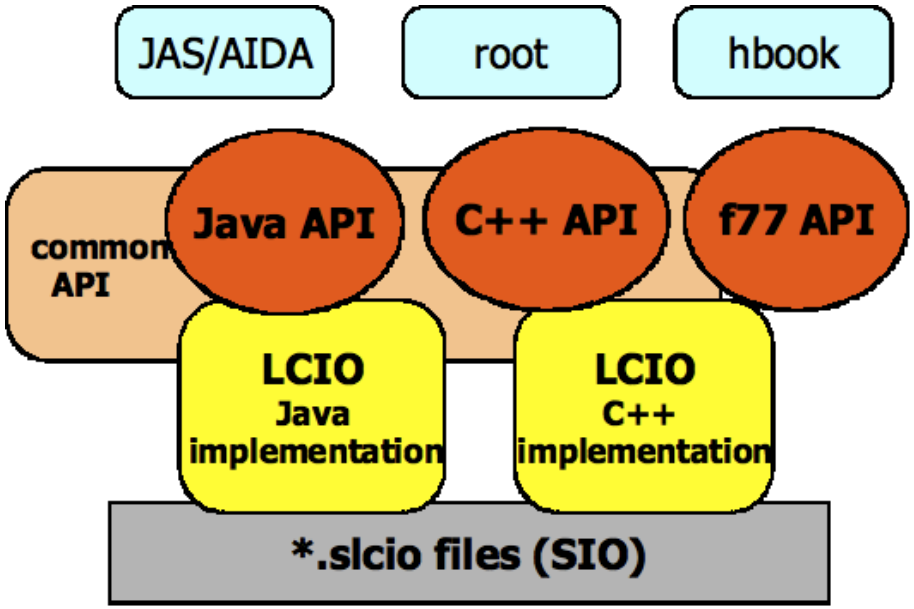
Improving the I/O and the Event Data Model

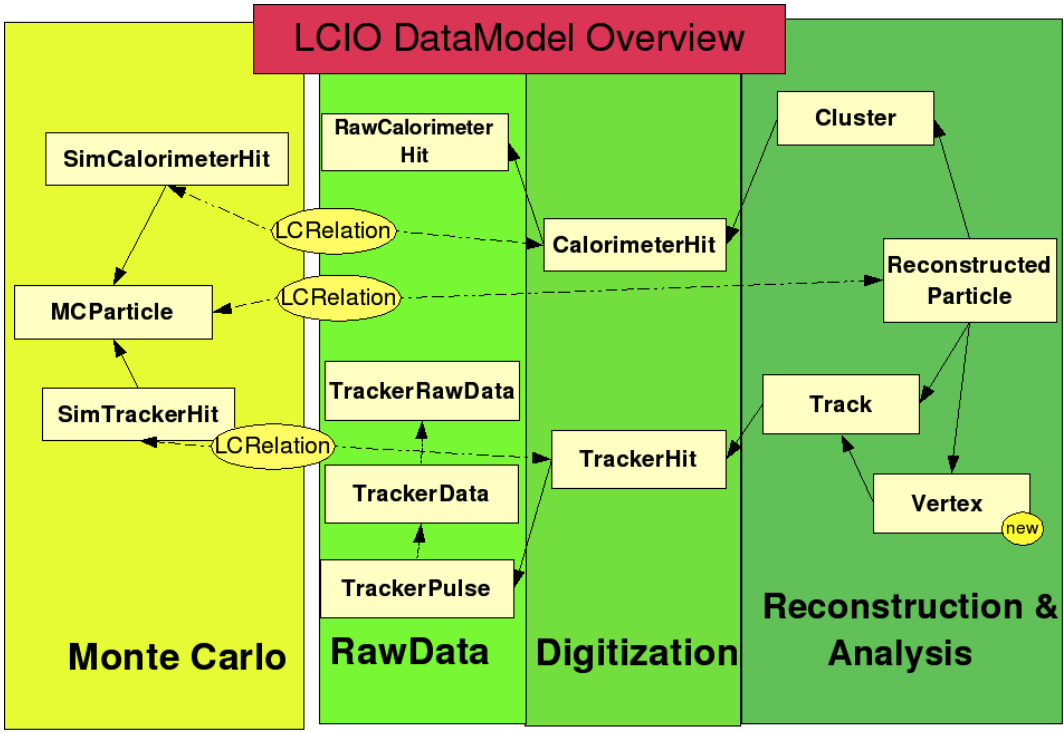
Frank Gaede
DESY

ILD Software Meeting
Paris, Jan 27, 2009

LCIO overview

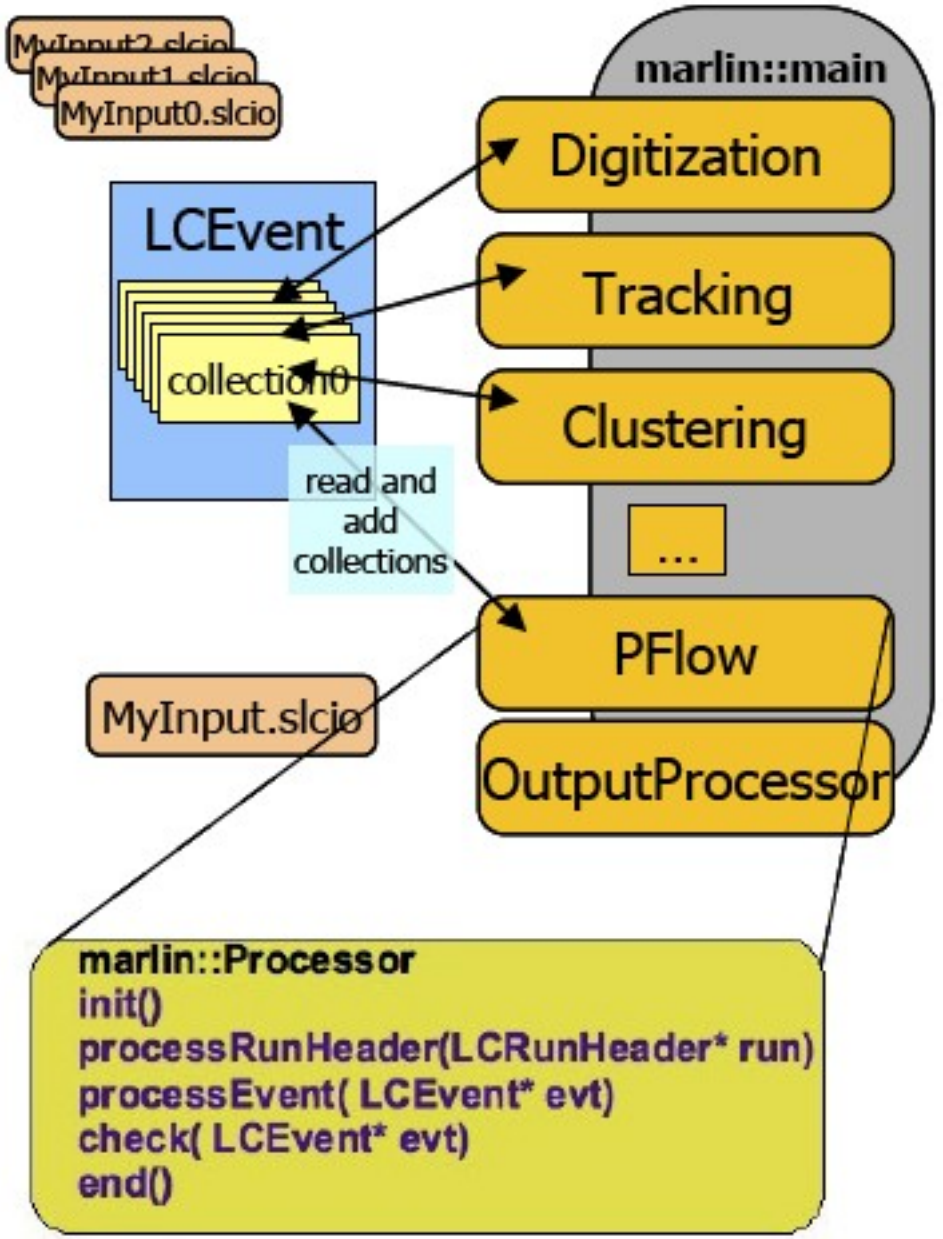
- LCIO provides a hierarchical **event data model** and a **persistency solution (I/O)** for LC software
  - DESY/SLAC project since 2002
- C++ and Java API
  - also f77 (obsolete) and Python (experimental)
- used in ILD and SID SW frameworks and in many ILC **testbeam** experiments





LCIO in Marlin

- LCIO provides the event data model in Marlin:
- transient==persistent event data
- software bus model
  - Marlin processors, eg PandoraPFAProcessor, LCFIVertexProcessor programmed against LCIO



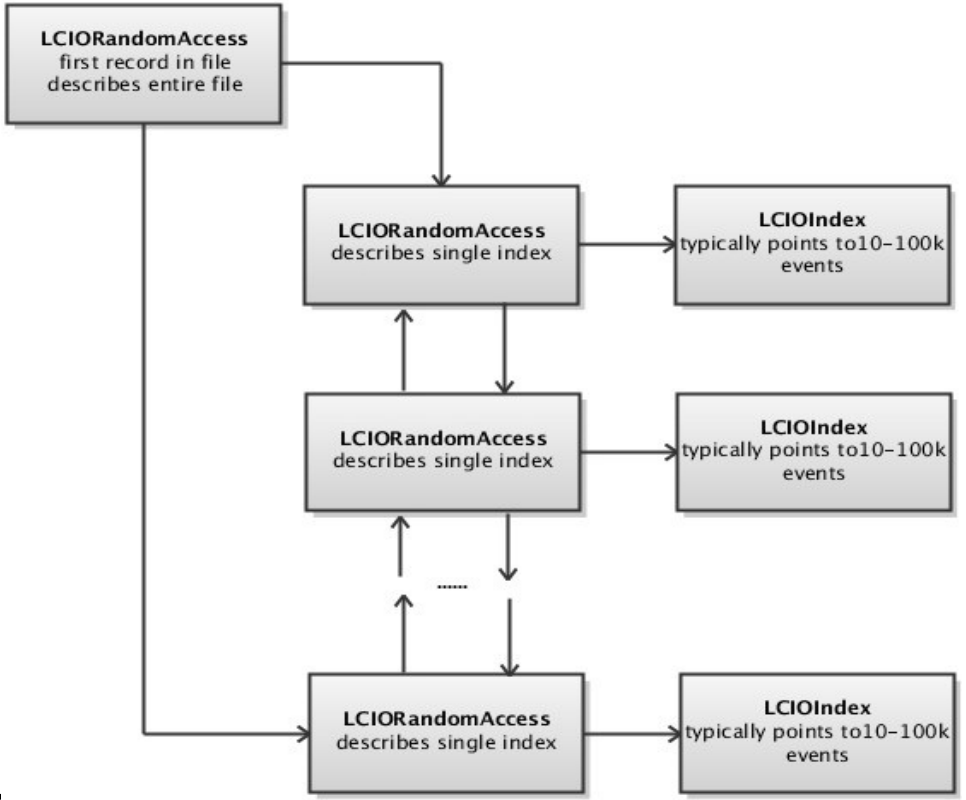
- effectively all existing LC software tools are programmed against LCIO
- need to evolve LCIO in a backward compatible way

LCIO features and possible improvements

- LCIO provides a rather complete event data model and has been used successfully in SID and ILD LOI mass production and in various R&D testbeam programs
- user extensions through use of LCGenericObject
  - no dictionary needed, i.e. anyone can read any LCIO file - but small performance penalty
- runtime extensions: attach arbitrary C++ types to any LCObject and N-to-M relationships (not used frequently !?)
- current I/O: SIO has compression one event per record
- possible (and requested) improvements:
  - **direct access to events (now only via fast skip ot TOC creation)**
  - **partial reading of events (e.g. only PandoraPFOs)**
  - **splitting of events over files (sim, rec, DST w/o dublication)**
  - **storing of (arbitrary) user classes**
  - **use LCIO with ROOT (ROOT macros, TTreeViewer, I/O,....)**
  - **improving the event data model (1d,2d hits, tracks/trajectories)**

direct access to LCIO events

- direct access to LCIO events needed:
  - overlay of random background events
  - physics analysis - reading of pre-selection
  - debugging
- now available through fast skip or creation of TOC (slow)



- proposed extension of LCIO/SIO (T. Johnson):
  - add two additional records LCIORandomAccess/LCIOIndex to SIO
  - allows to create index of LCIO events over arbitrarily large sets of files
  - direct access to events - possibly w/ pre-selection criteria **(E_t>50GeV)**
- first implementation for Java exists in exp. cvs branch - need to test and implement in C++
- Note: ROOT I/O would 'automatically' provide direct access

partial reading & splitting of events

- needed for performance and cost (disk space) issues:
  - 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
  - not available in SIO now and can't use TRefs in ROOT
- need index based pointers independent of I/O, e.g.:
  - long64 index = HASH(collName) << 32 | collIndex
- experimental C++ version exists in ROOT I/O branch for partial reading of events (not yet file splitting)
  - need further testing & implementation in SIO (also Java)
  - need extension of LCIO::Reader interface

storing of arbitrary user classes

- LCIO event data model rather complete - but also clear need for storing user defined information
  - LCGenericObjects can store almost arbitrary data structures based on ints, floats and doubles
    - files can be read w/o any additional code (dictionary)
    - small performance penalty
    - extensively used in LCCD (conditions data) by testbeams
- occasional user request for 'natively' storing arbitrary user classes in LCIO
  - possible in principle with LCIO/SIO (not documented and somewhat 'discouraged') - would come 'for free' w/ ROOT I/O
- IMHO: success of LCIO is to a large extend due to the slightly restrictive definition of the event data model i.e. the interfaces between modules/processors

ROOT I/O for LCIO

- user request to have closer link of LCIO to ROOT
  - use LCIO classes in ROOT macros (former GLD groups)
  - have fast interactive analysis with ROOT tree
- investigate the optional use of ROOT I/O for LCIO
  - would provide 'missing features': direct access, partial reading and splitting of events (and streaming of user classes)
- created experimental branch in cvs (rio_v00-00)
  - create ROOT dictionary w/ help from ROOT team
  - implemented index based pointers for C++
  - needed some changes to LCIO classes: LCTCollection<T>, std::vector as members,...
  - can create almost complete copies of LCIO DST in ROOT
    - no subcollections (pointers only) yet
  - streaming mode for Marlin under development
- see: talks at ILD software meetings for details
- still some issues to resolve (interface to Java !!)

a ROOT dictionary for LCIO

- the latest patch version of LCIO v01-12-01 allows to optionally create a ROOT dictionary for all LCIO classes - 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

Improving the LCIO event data model

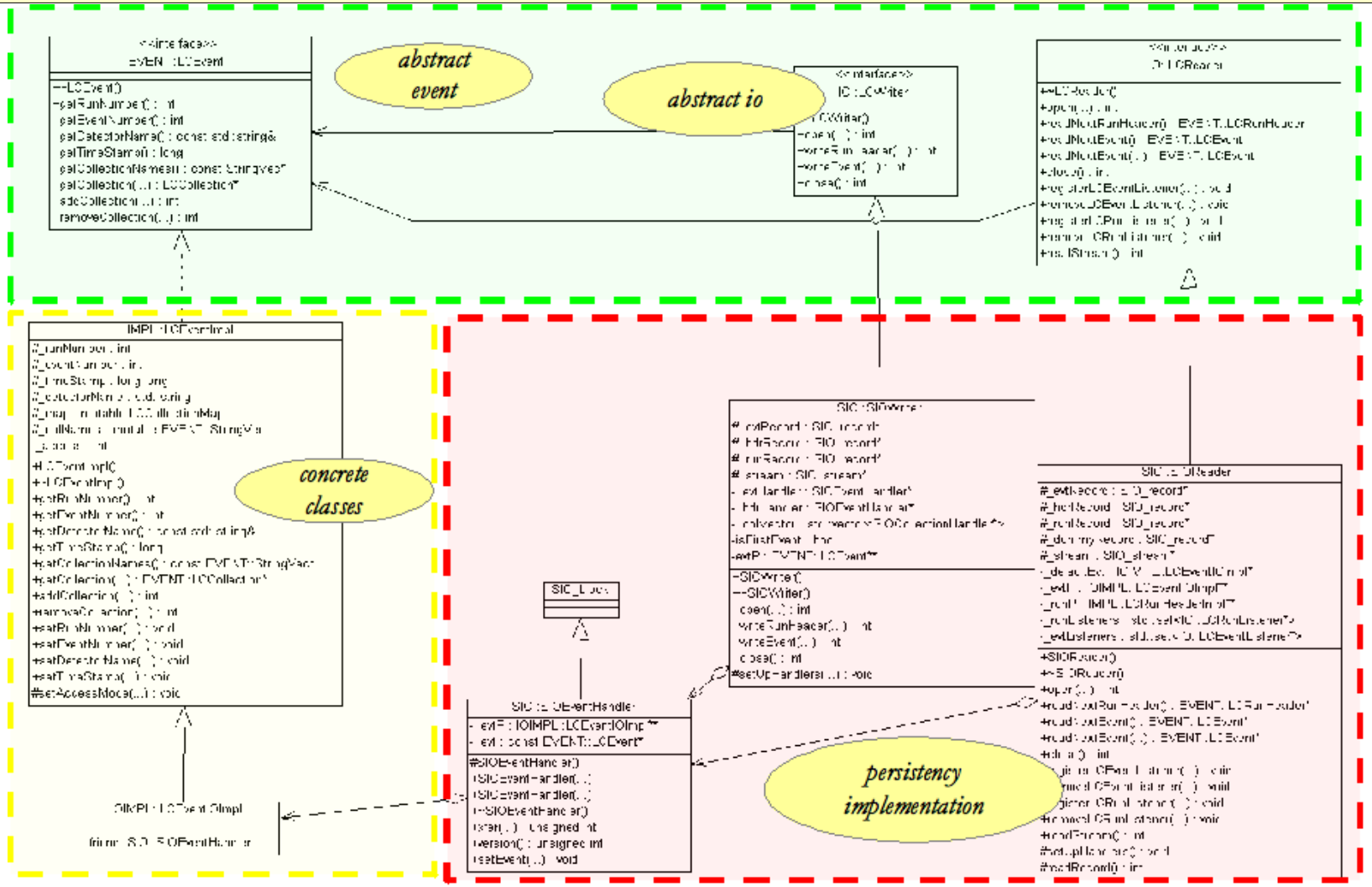
- suggested improvements to the event data model:
- 1D, 2D tracker hits
  - LCIO (Sim)TrackerHit is a 3D space point - whereas actual measurements are either 1D (strip) or 2D (TPC) where the detector surface (line) provides the additional geometry information
- Track
  - the current LCIO Track class consists of pointers to all TrackerHits and one set of (Helix) parameters to these hits
  - generally one wants to have multiple fits for one set of hits, e.g. at the IP or at the face of the calorimeter - could store list of parameterizations per Track
  - Trajectory could be introduced as high level convenient view to these fits
  - currently not straight forward (though possible) to store kinks in LCIO
- detailed proposal under development (N.Graf)
- user feedback needed - also for other improvements

Summary & Outlook

- currently the LCIO team is working on 'LCIOv2' to further improve LCIO and address the following feature requests:
  - direct access
  - partial reading and splitting over files
  - linking LCIO closer to ROOT for analysis
  - improving the event data model
- other improvements not shown today are also discussed:
  - make interfaces more consistent and more convenient to use
  - better define the suggested use of meta-data
  - provide LCIO file browser (JAS)
  - ...
- make use of the 'spare time' in 2010 until preparation for the DBD reports in 2012 will have to start
- continuos user feedback is important and welcome !

additional material

LCIO software design

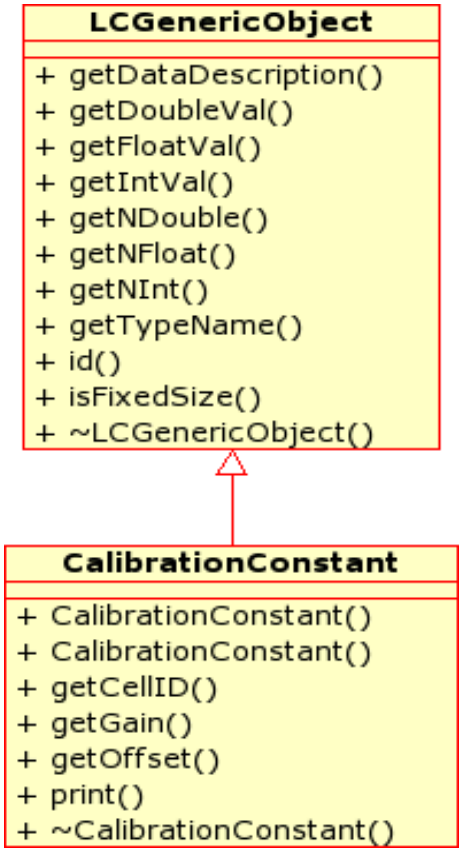


- event data model is strictly decoupled from persistency package – currently SIO, but can be changed
- user code only sees pure abstract interface (Reading) or LCIO implementation classes (Writing)

LCIO – user extensions

- LCIO defines the event data model and provides the persistency for it
- however users want to extend existing classes and persist their own classes
- LCGenericObject provided by LCIO:
  - users can store 'arbitrary' data structures in LCGenericObject w/o writing streamer code
  - performance not great
- LCIO runtime extensions (C++)
- extension of the object with arbitrary (even non-LCObject) classes
- bidirectional relations between LCObjects
  - one to one
  - one to many
  - many to many

no persistency yet



SIO persistency

- simple C++ persistency tool developed at SLAC
- provides some OO-features like pointer chasing
- user needs to write streamer code (done in LCIO)

- missing so far:
  - splitting of events over files
  - direct access
  - user streamer code
- could be implemented rather easily, if needed