Core software tools for JRA1

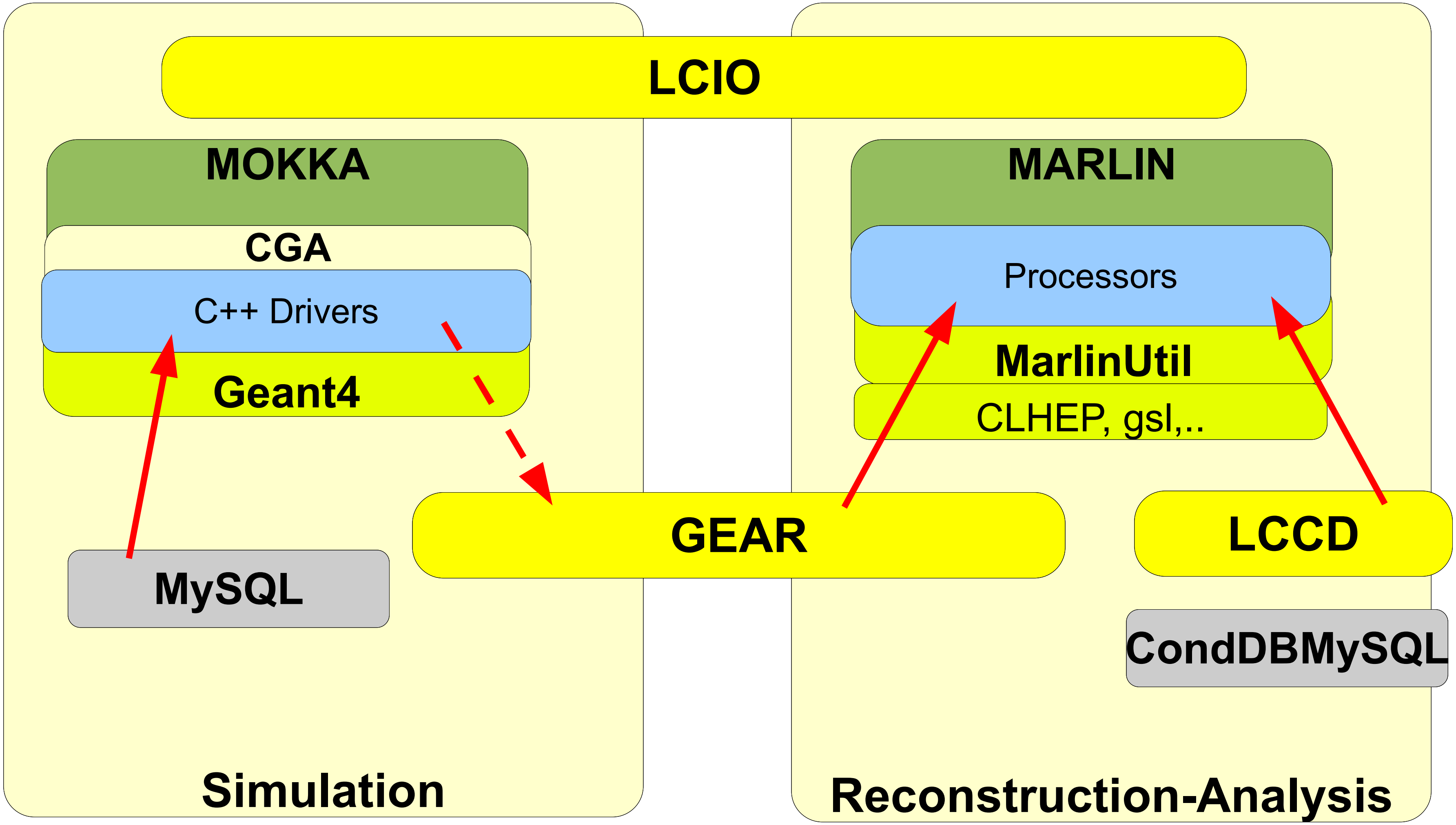
Frank Gaede
DESY

EUDET – JRA1 Software Meeting,
Geneva
March 30, 2007

Outline

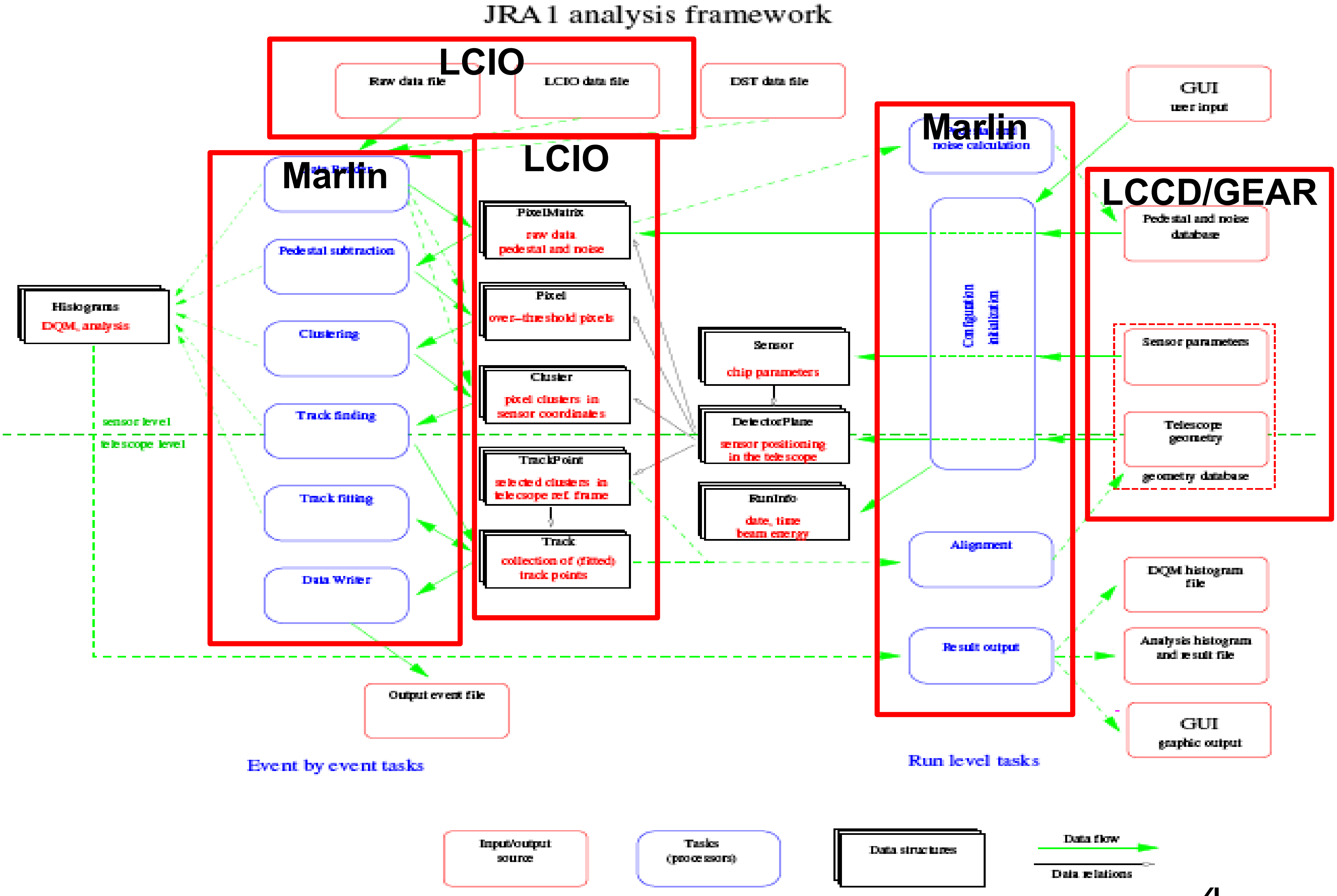
- intro - overview core tools
- DAQ data format
  - LCIO
- geometry information
  - GEAR
- conditions data
  - LCCD
- event display
  - CED
- histograms
  - (R)AIDA, root
- Summary

ILC-LDC software framework



all tools are also used in testbeam programs

JRA1 SW-Framework



JRA1 analysis framework

LCIO

Raw data file

LCIO data file

DST data file

GUI
user input

Marlin

Pedestal and noise calculation

LCCD/GEAR

Marlin

Data Reader

LCIO

PixelMatrix
raw data
pedestal and noise

Pedestal subtraction

Pedestal and noise database

Pixel
over–threshold pixels

Histograms
DQM, analysis

Clustering

Configuration initialization

Sensor
chip parameters

Sensor parameters

Cluster
pixel clusters in sensor coordinates

Track finding

sensor level

telescope level

DetectorPlane
sensor positioning in the telescope

Telescope geometry

geometry database

TrackPoint
selected clusters in telescope ref. frame

Track fitting

RunInfo
date, time
beam energy

Track
collection of (fitted) track points

Alignment

Data Writer

DQM histogram file

Analysis histogram and result file

Result output

Output event file

GUI
graphic output

Event by event tasks

Run level tasks

Input/output source

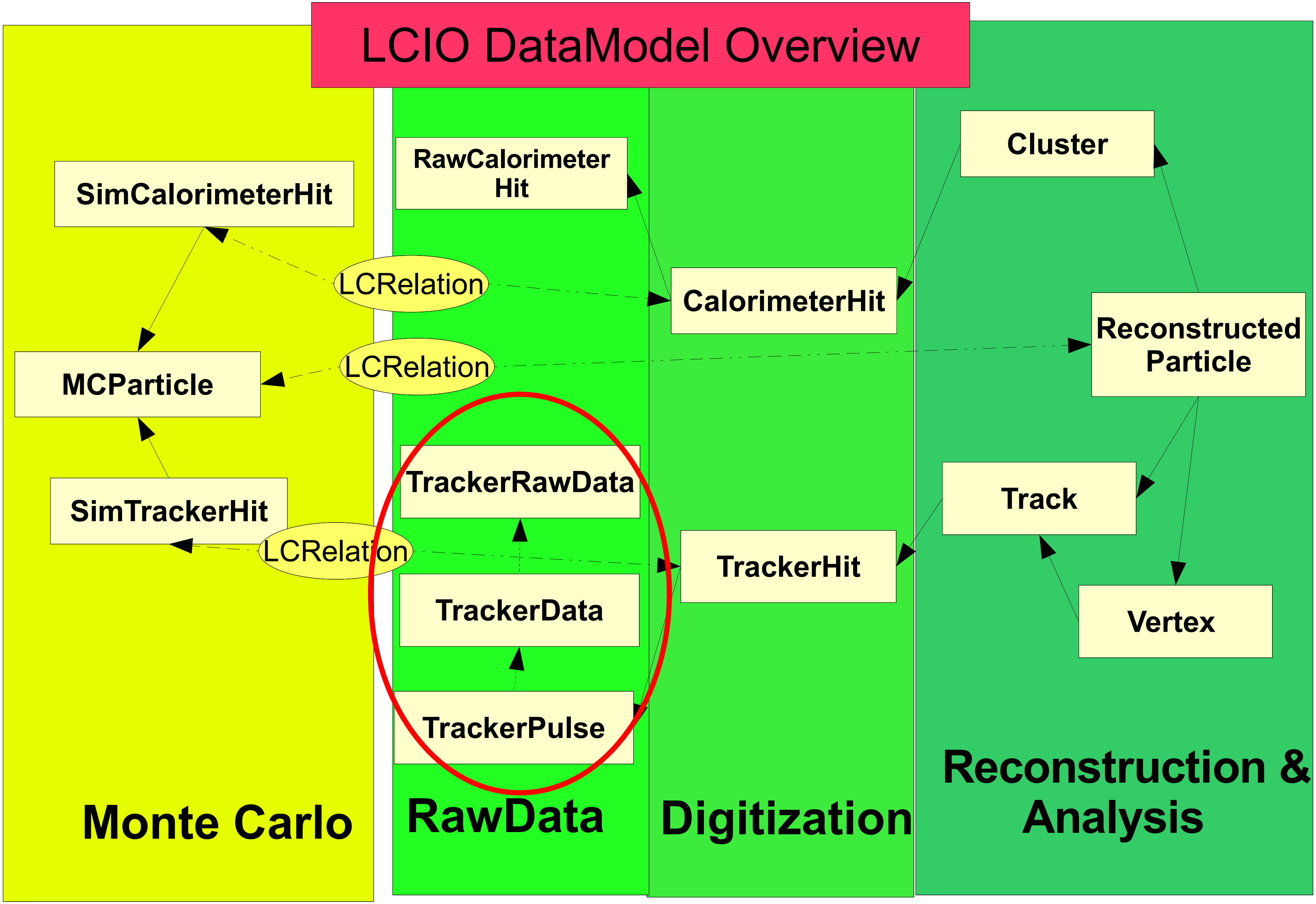
Tasks (processors)

Data structures

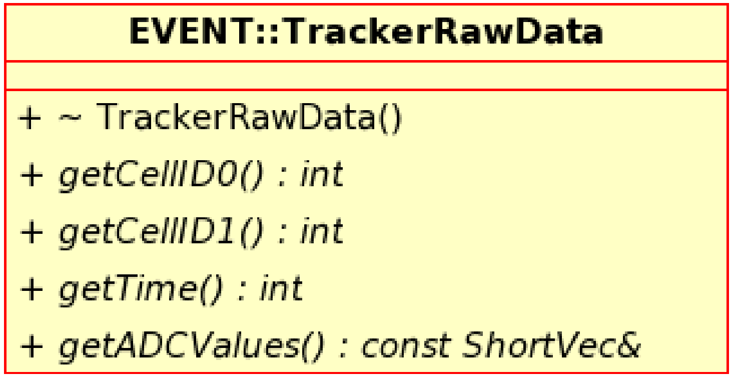
Data flow

Data relations

LCIO Event Data Model

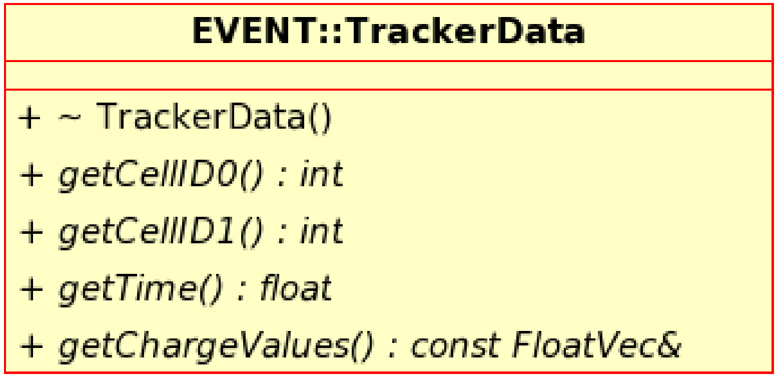


LCIO raw data classes for tracker



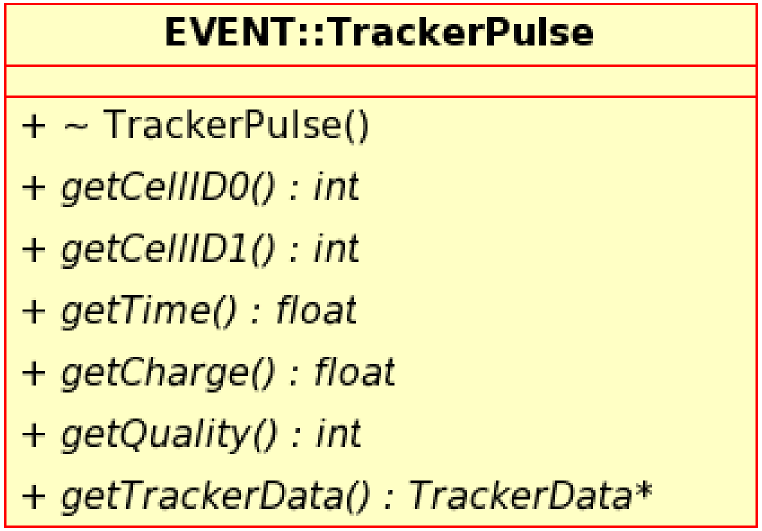
EVENT::TrackerRawData

+ ~ TrackerRawData()
+ *getCellID0() : int*
+ *getCellID1() : int*
+ *getTime() : int*
+ *getADCValues() : const ShortVec&*



EVENT::TrackerData

+ ~ TrackerData()
+ *getCellID0() : int*
+ *getCellID1() : int*
+ *getTime() : float*
+ *getChargeValues() : const FloatVec&*



EVENT::TrackerPulse

+ ~ TrackerPulse()
+ *getCellID0() : int*
+ *getCellID1() : int*
+ *getTime() : float*
+ *getCharge() : float*
+ *getQuality() : int*
+ *getTrackerData() : TrackerData**

feature extracted signal
for one cell

raw readout classes
with n measurements per cell
uncalibrated (*short*) & calibrated (*float*)

- used by TPC prototypes
VTX prototypes/testbeams
- **used in Pixeltelescope code**
(A.Bulgheroni)

LCIO as DAQ format

- LCIO can be used as native DAQ format
  - data classes for raw data exist (RawTrackerData,...)
  - LCGenericObject provides way for arbitrary user defined classes (performance issue!)
- if needed, LCIO could be extended to serve as a container for raw data records
  - -> use the same persistency package that is also used in analysis and simulation
- need agreement between all DAQ groups within EUDET
  - -> dedicated DAQ software meeting May 2nd in Orsay – prior to software workshop
  - work/discussion already started within LCIO developers group

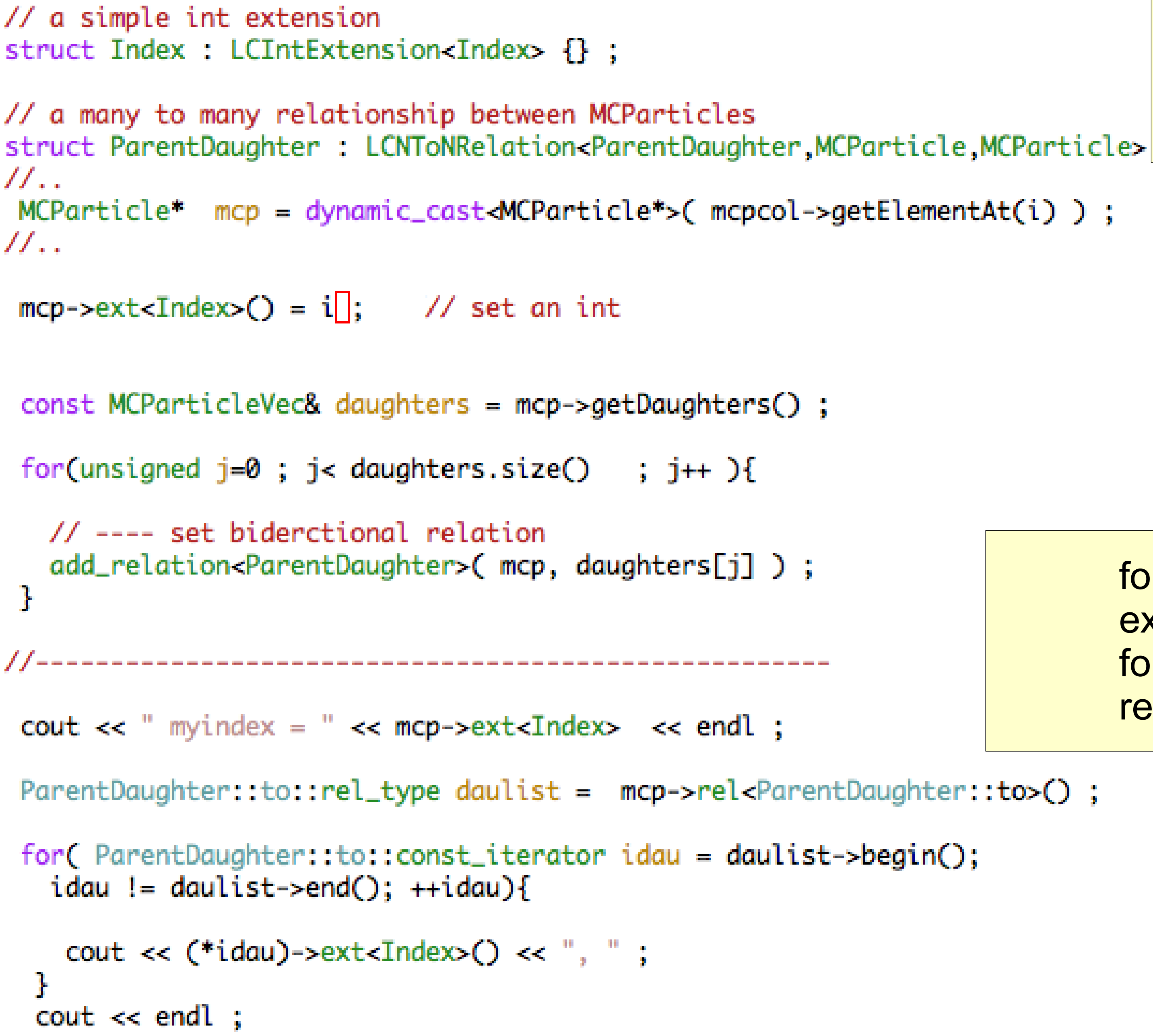
your requirements are needed !

LCIO runtime extensions

- long pending user request:
  - attach arbitrary user objects to LCObjects
  - fast and easy creation of links (relations) between various LCObject subtypes, eg. TrackerHits and Track
- features
  - extension of the object with arbitrary (even non-LCObject) classes
  - extension of single objects or vectors, lists of objects
  - optionally ownership is taken for extension objects (memory management)
  - bidirectional relations between LCObjects
    - one to one
    - one to many
    - many to many

LCIO runtime extensions

extensions and relations identified through a tagging class T

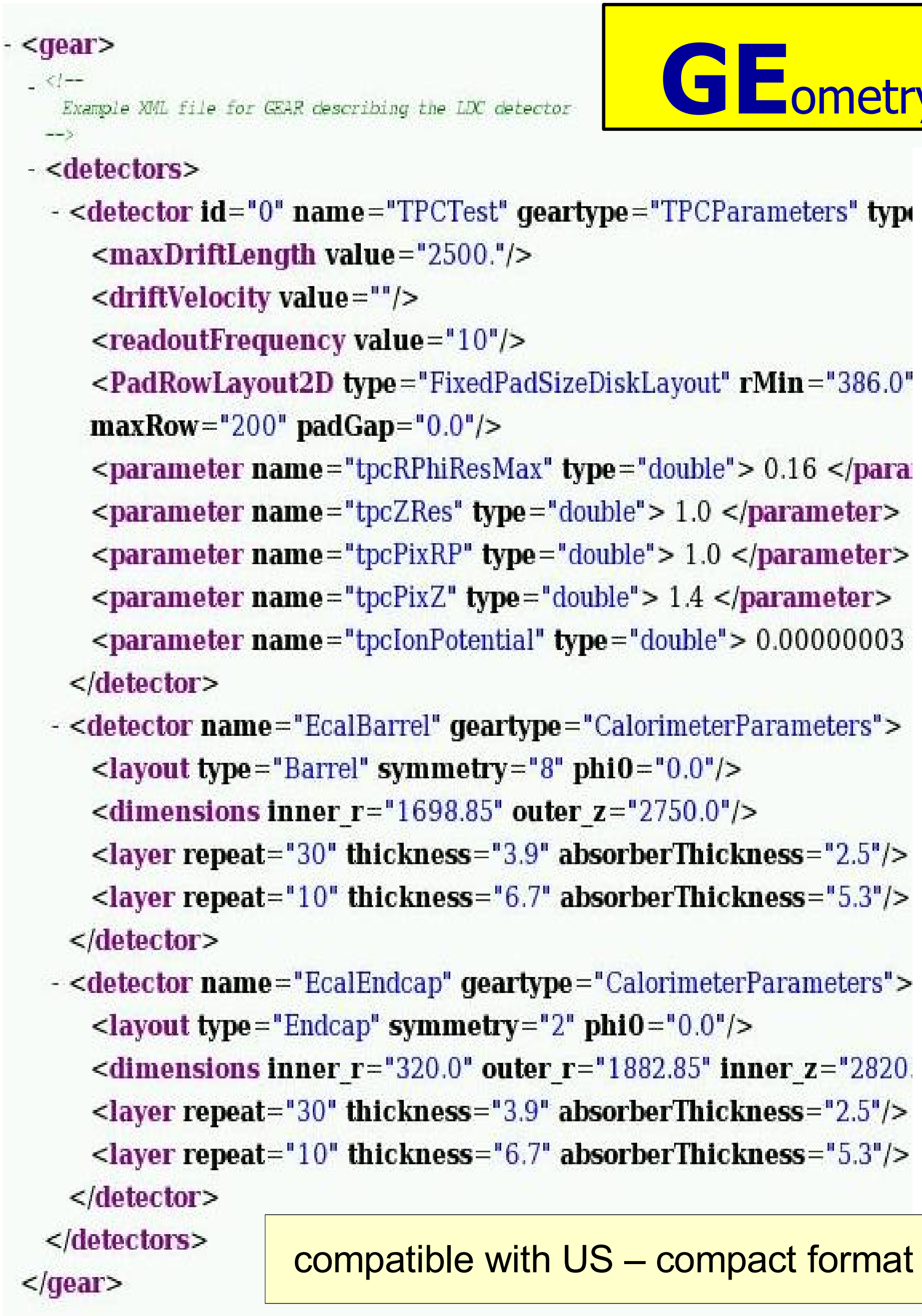


```
// a simple int extension
struct Index : LCIntExtension<Index> {} ;

// a many to many relationship between MCParticles
struct ParentDaughter : LCNToNRelation<ParentDaughter,MCParticle,MCParticle>
//..
 MCParticle*  mcp = dynamic_cast<MCParticle*>( mcpcol->getElementAt(i) ) ;
//..

 mcp->ext<Index>() = i;     // set an int


 const MCParticleVec& daughters = mcp->getDaughters() ;

 for(unsigned j=0 ; j< daughters.size()   ; j++ ){

   // ---- set biderctional relation
   add_relation<ParentDaughter>( mcp, daughters[j] ) ;
 }

//-------------------------------------------------------

 cout << " myindex = " << mcp->ext<Index>  << endl ;

 ParentDaughter::to::rel_type daulist =  mcp->rel<ParentDaughter::to>() ;

 for( ParentDaughter::to::const_iterator idau = daulist->begin();
   idau != daulist->end(); ++idau){

    cout << (*idau)->ext<Index>() << ", " ;
  }
  cout << endl ;
```

for extensions use
ext<T>()
for relations use
rel<T>

Gear

GEometry **A**PI for **R**econstruction

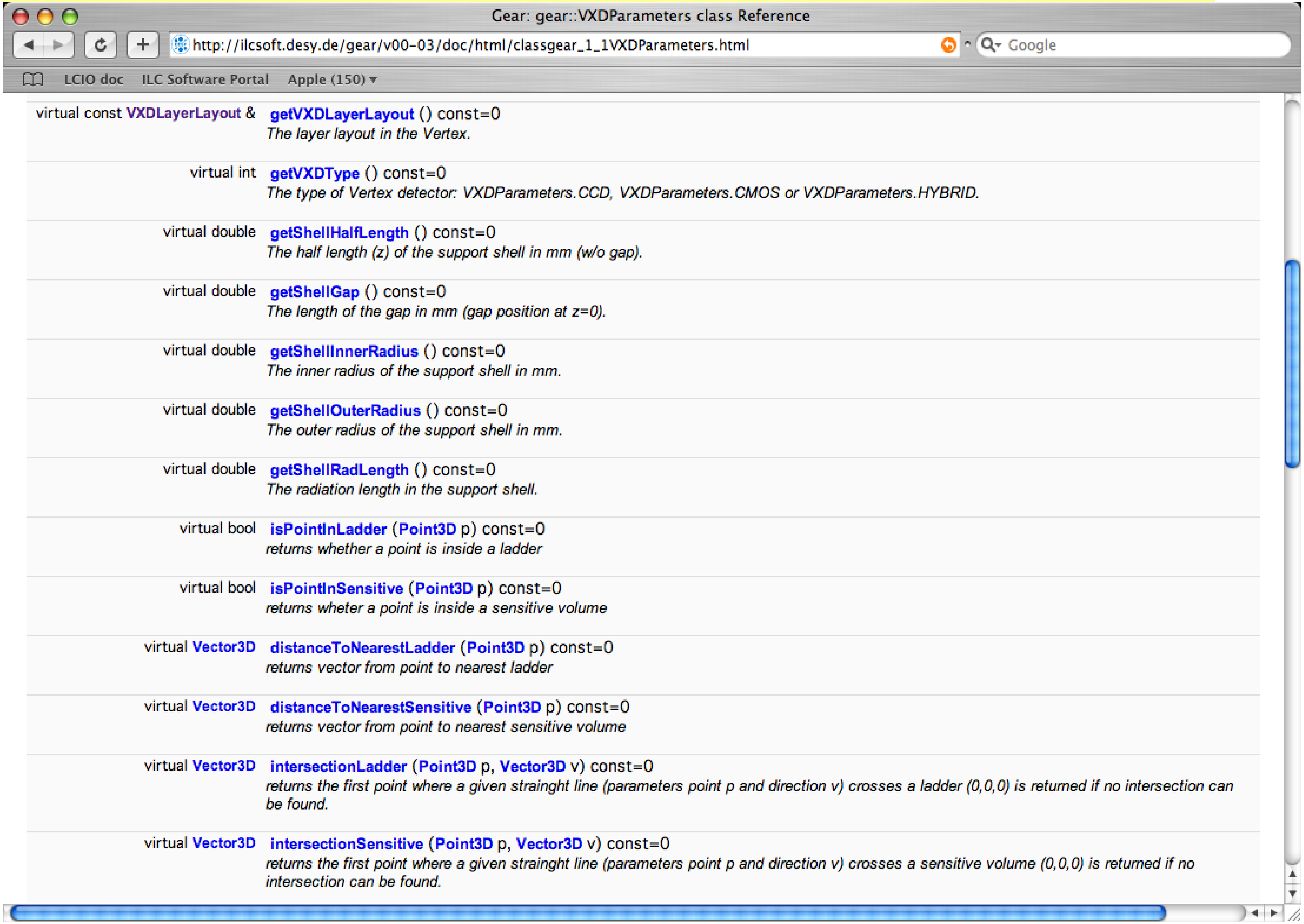


```
- <gear>
  - <!--
      Example XML file for GEAR describing the LDC detector
    -->
  - <detectors>
    - <detector id="0" name="TPCTest" geartype="TPCParameters" typ
        <maxDriftLength value="2500."/>
        <driftVelocity value=""/>
        <readoutFrequency value="10"/>
        <PadRowLayout2D type="FixedPadSizeDiskLayout" rMin="386.0"
        maxRow="200" padGap="0.0"/>
        <parameter name="tpcRPhiResMax" type="double"> 0.16 </para
        <parameter name="tpcZRes" type="double"> 1.0 </parameter>
        <parameter name="tpcPixRP" type="double"> 1.0 </parameter>
        <parameter name="tpcPixZ" type="double"> 1.4 </parameter>
        <parameter name="tpcIonPotential" type="double"> 0.00000003
      </detector>
    - <detector name="EcalBarrel" geartype="CalorimeterParameters">
        <layout type="Barrel" symmetry="8" phi0="0.0"/>
        <dimensions inner_r="1698.85" outer_z="2750.0"/>
        <layer repeat="30" thickness="3.9" absorberThickness="2.5"/>
        <layer repeat="10" thickness="6.7" absorberThickness="5.3"/>
      </detector>
    - <detector name="EcalEndcap" geartype="CalorimeterParameters">
        <layout type="Endcap" symmetry="2" phi0="0.0"/>
        <dimensions inner_r="320.0" outer_r="1882.85" inner_z="2820.
        <layer repeat="30" thickness="3.9" absorberThickness="2.5"/>
        <layer repeat="10" thickness="6.7" absorberThickness="5.3"/>
      </detector>
    </detectors>
</gear>
```

compatible with US – compact format

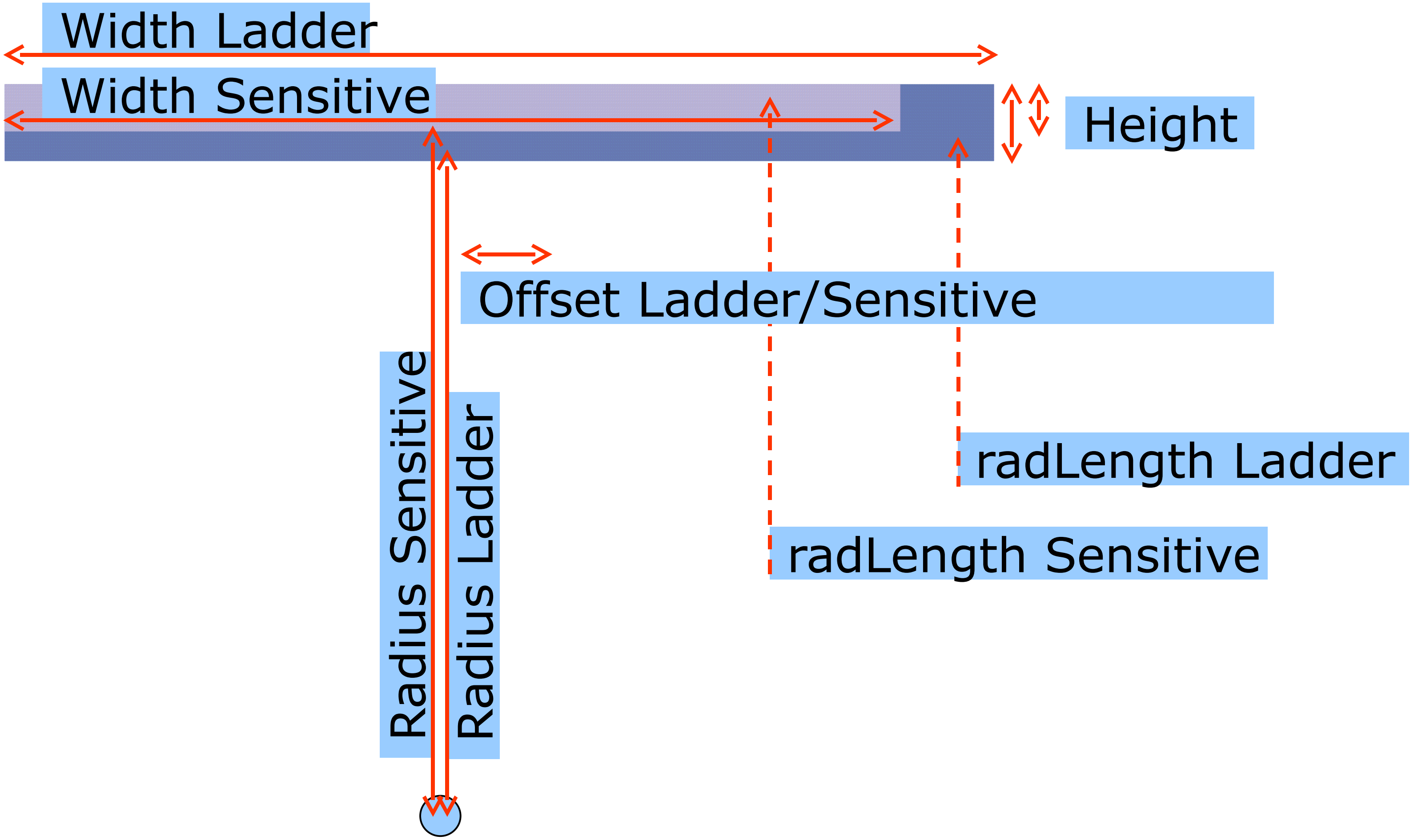
- well defined geometry definition for reconstruction that
  - is flexible w.r.t different detector concepts
  - has high level information needed for reconstruction
  - provides access to material properties
- **abstract interface (a la LCIO)**
  - concrete implementation based on XML files
  - and Mokka-CGA

Gear API for VXD/Pixel-detectors I


Gear: gear::VXDParameters class Reference
http://ilcsoft.desy.de/gear/v00-03/doc/html/classgear_1_1VXDParameters.html
Google
LCIO doc
ILC Software Portal
Apple (150)

virtual const VXDLayerLayout &	getVXDLayerLayout () const=0 *The layer layout in the Vertex.*
virtual int	getVXDType () const=0 *The type of Vertex detector: VXDParameters.CCD, VXDParameters.CMOS or VXDParameters.HYBRID.*
virtual double	getShellHalfLength () const=0 *The half length (z) of the support shell in mm (w/o gap).*
virtual double	getShellGap () const=0 *The length of the gap in mm (gap position at z=0).*
virtual double	getShellInnerRadius () const=0 *The inner radius of the support shell in mm.*
virtual double	getShellOuterRadius () const=0 *The outer radius of the support shell in mm.*
virtual double	getShellRadLength () const=0 *The radiation length in the support shell.*
virtual bool	isPointInLadder (Point3D p) const=0 *returns whether a point is inside a ladder*
virtual bool	isPointInSensitive (Point3D p) const=0 *returns wheter a point is inside a sensitive volume*
virtual Vector3D	distanceToNearestLadder (Point3D p) const=0 *returns vector from point to nearest ladder*
virtual Vector3D	distanceToNearestSensitive (Point3D p) const=0 *returns vector from point to nearest sensitive volume*
virtual Vector3D	intersectionLadder (Point3D p, Vector3D v) const=0 *returns the first point where a given strainght line (parameters point p and direction v) crosses a ladder (0,0,0) is returned if no intersection can be found.*
virtual Vector3D	intersectionSensitive (Point3D p, Vector3D v) const=0 *returns the first point where a given strainght line (parameters point p and direction v) crosses a sensitive volume (0,0,0) is returned if no intersection can be found.*

VTX ladder



detailed description of the ladder position allows to describe all ILC vertex detectors

- not yet covered: misalignment

MokkaGear

- extension to Mokka
  - have only one source of of information for describing the detector geometry (however have to start with simulation also for testbeam)
- extract geometry information in drivers when detector is built
- use Gear to create XML files for reconstruction
- currently implemented:
  - TPC, Ecal, Hcal, **Lcal**, VTX
- **released with Mokka 6.3**
- NOTE: in planned new system LCGO there will be an independent and standalone geometry descriptionthat is fed into all other tools

Gear for pixel telescope

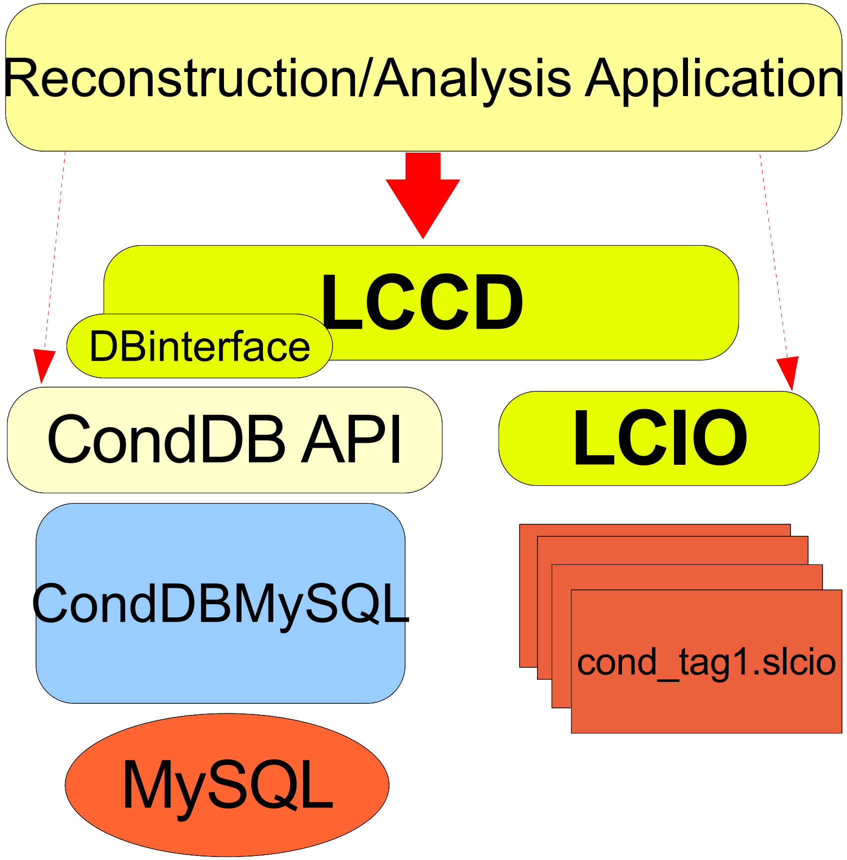
start today !

- need to define API
  - similar to VTX detector
  - should be designed to be suitable for FTD as well
    - planar r-phi detectors in general
- then need implementation
  - in Gear
    - geometrical functionality
    - xml description, parser
  - Mokka:
    - fill appropriate Gear object to write out xml file for reconstruction and

LCCD

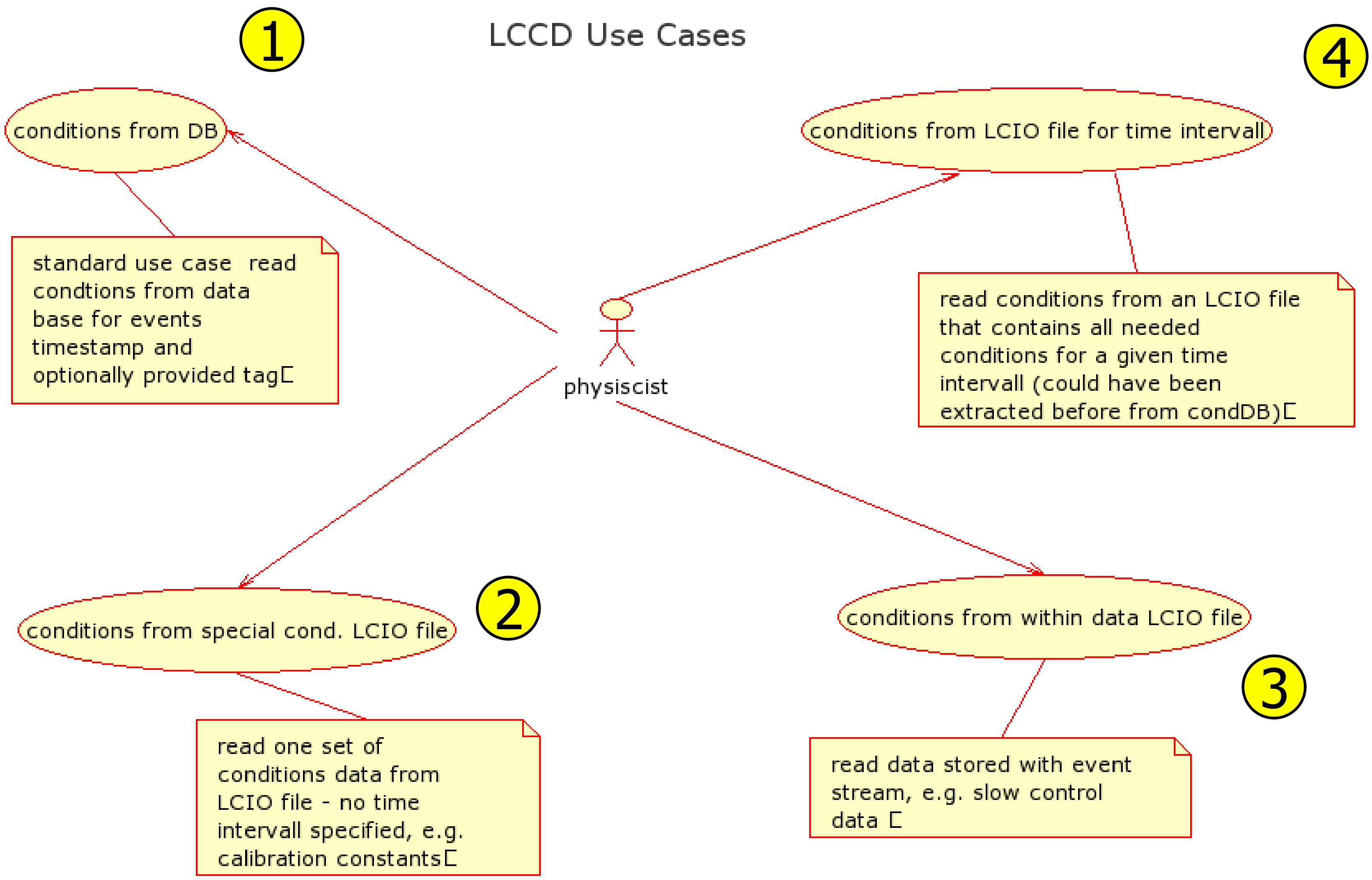
Linear **C**ollider **C**onditions **D**ata Toolkit

- Reading conditions data
  - from conditions database
  - from simple LCIO file
  - from LCIO data stream
  - from dedicated LCIO-DB file
- Writing conditions data
- tag conditions data
- Browse the conditions database
  - through creation of LCIO files
    - vertically (all versions for timestamp)
    - horizontally (all versions for tag)



LCCD is used by Calice and TPC groups for the conditions data of the ongoing testbeam studies

LCCD Use Cases



LCCD Use Cases

physiscist

1

conditions from DB

standard use case read condtions from data base for events timestamp and optionally provided tag[

2

conditions from special cond. LCIO file

read one set of conditions data from LCIO file - no time intervall specified, e.g. calibration constants[

3

conditions from within data LCIO file

read data stored with event stream, e.g. slow control data [

4

conditions from LCIO file for time intervall

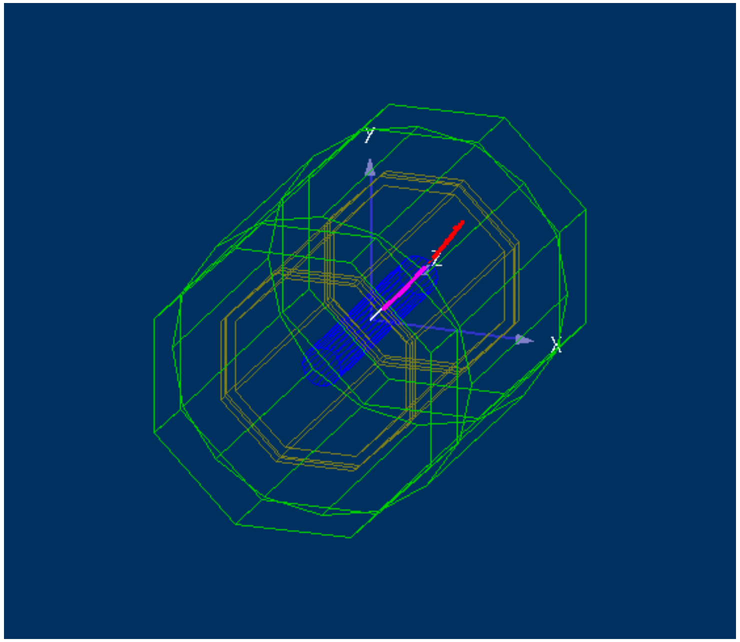
read conditions from an LCIO file that contains all needed conditions for a given time intervall (could have been extracted before from condDB)[

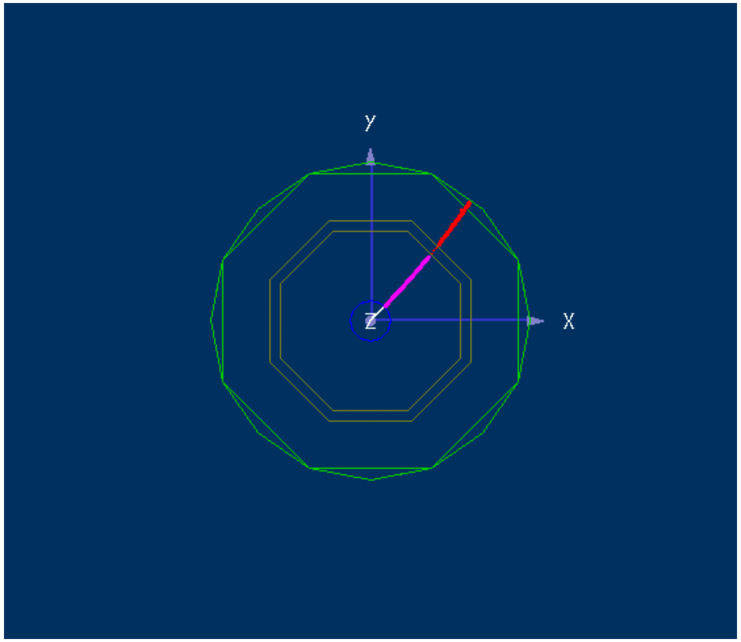
LCCD for Pixeltelescope

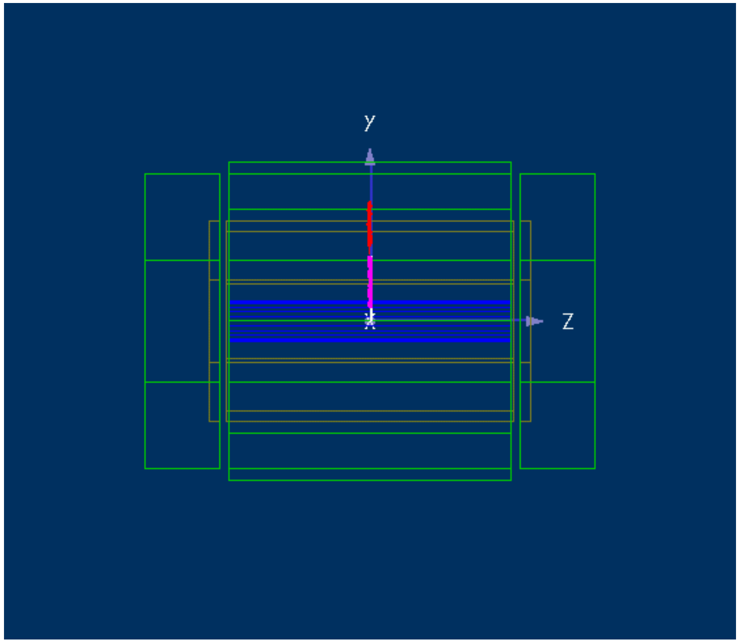
- the advantage of LCCD is transparent access to conditions data (as LCIO collections, e.g. in Marlin) regardless of the actual source
- possible scenario:
  - start with simple LCIO files
  - eventually install a centrally managed conditions data base and import existing data form LCIO files
  - code will be unchanged (only steering)
- NB: maintaining a (conditions) data base is not an entirely trivial task – needs dedicated manpower !
  - not covered by LCCD
  - should learn from calice experience !

CED Event display

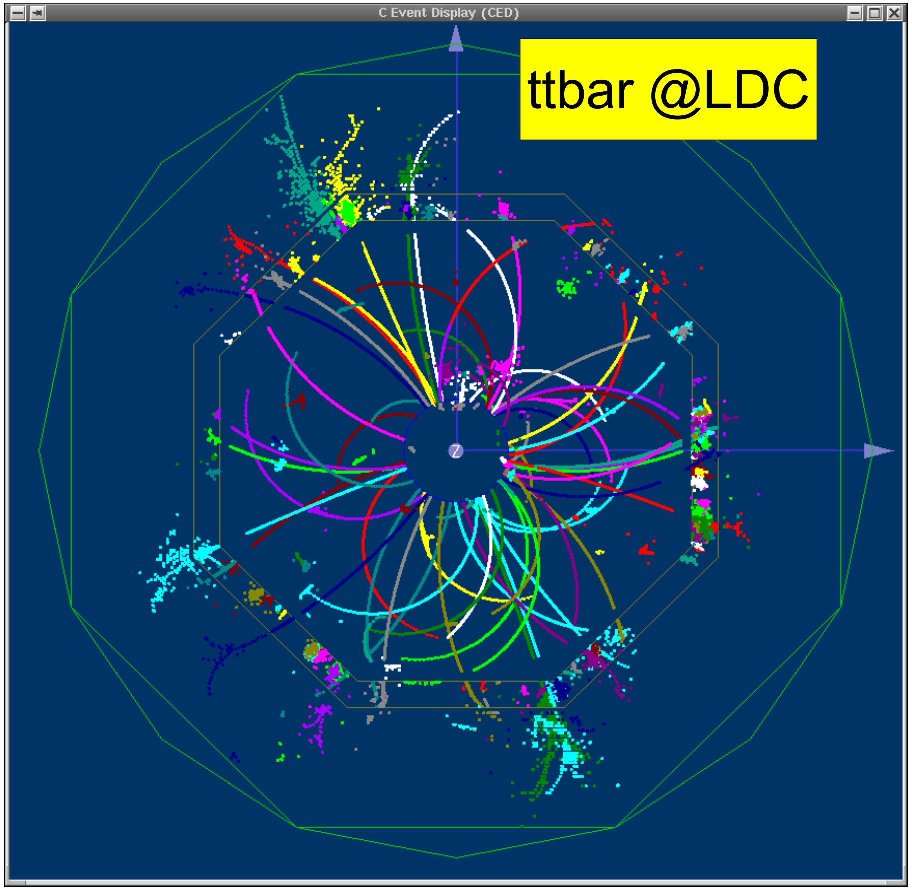
- OpenGL (glut) based event display
- 3D view, rotate, zoom, shift,...
- supports various layers
- integrated with MARLIN and GEAR

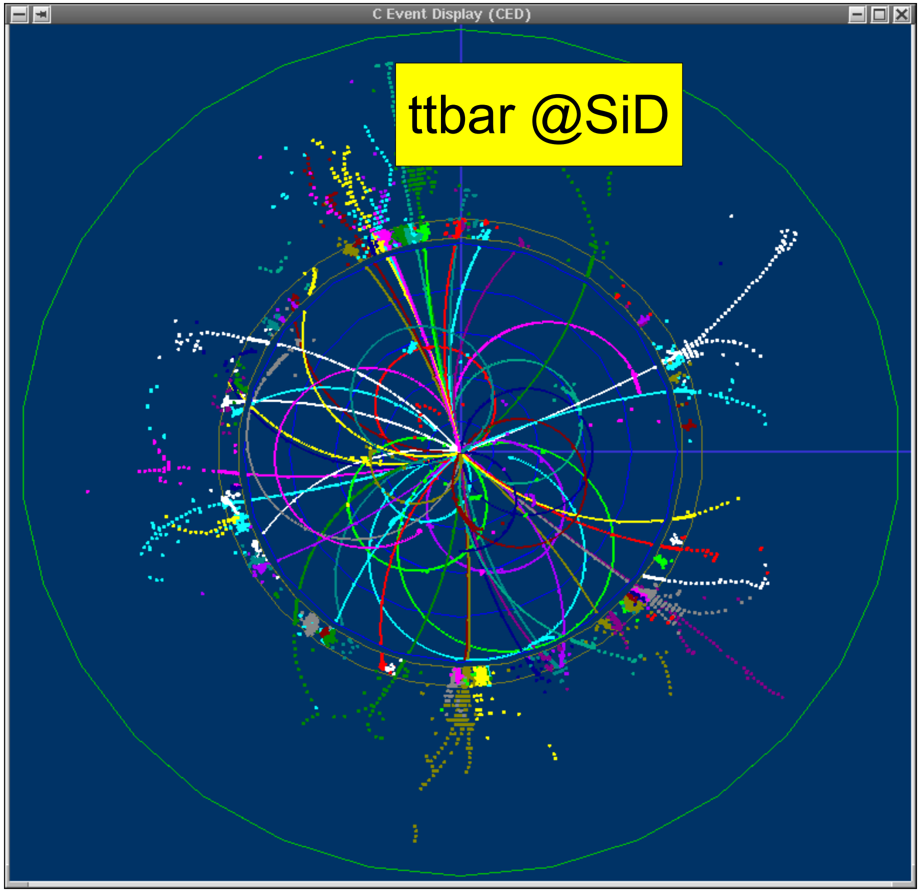






CED Event display II





can easily be adapted for testbeams

other Event displays

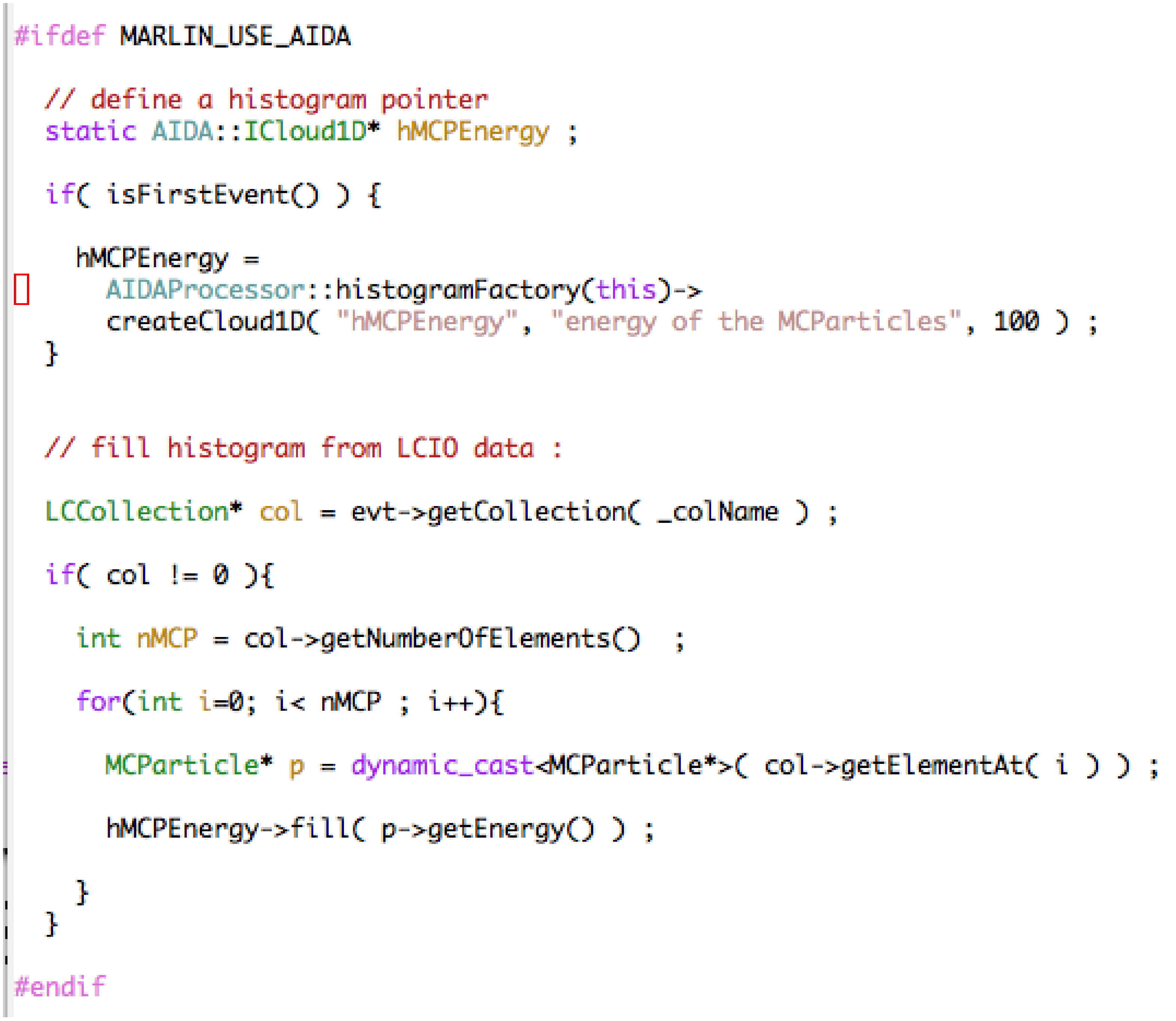
- CED is fast and easy to adapt for any displaying purpose, however it is not a full event display yet:
  - no picking
  - somewhat cumbersome handling through keystrokes
  - -> needs further development (volunteers ?)
- WIRED/JAS3 – full blown event display (knows LCIO)
  - written in Java (rather slow on Linux)
- Calice and TPC testbeams have their own online event displays
  - -> somewhat unfortunate situation
  - -> needs discussion and agreement, maybe at Orsay workshop !

your input is needed
please do not write yet another event display !

Histograming

- general agreement not to (explicitly) use root in ILC core software !
- core software (MarlinReco) uses AIDA for histogramming (http://www.freehep.org/AIDA)
  - allows decision about histograming tool to be made at link time, e.g.
  - RAIDA – implementation based on root
    - not fully implemented but standard histograms and ntuples work fine
- others: JAIDA, OpenScientist, Pi, PAIDA,...
- AIDA in Marlin is very simple to use, see $MARLIN/example/mymarlin/src/MyProcsessor.cc

example: AIDA histogram in Marlin



```
#ifdef MARLIN_USE_AIDA

  // define a histogram pointer
  static AIDA::ICloud1D* hMCPEnergy ;

  if( isFirstEvent() ) {

    hMCPEnergy =
      AIDAProcessor::histogramFactory(this)->
      createCloud1D( "hMCPEnergy", "energy of the MCParticles", 100 ) ;
  }


  // fill histogram from LCIO data :

  LCCollection* col = evt->getCollection( _colName ) ;

  if( col != 0 ){

    int nMCP = col->getNumberOfElements()  ;

    for(int i=0; i< nMCP ; i++){

      MCParticle* p = dynamic_cast<MCParticle*>( col->getElementAt( i ) ) ;

      hMCPEnergy->fill( p->getEnergy() ) ;

    }
  }

#endif
```

using root directly

- of course one can also use root directly in any Marlin processor
  - however:
    - you have to manage the file(s)
      - one global file – one file for every processor ?
    - you create an explicit dependency on root
    - users can not switch to other histograming tools
    - your processors (Kalman-Filtering,...) can not be (easily) integrated in core software tools like MarlinReco
    - ...

core software ongoing work

- ilcinstall build tool
  - easy to configure python script that allows complete installation of ILC core software tools
  - downloads and builds QT, CLHEP, LCIO, cernlib, Marlin, MarlinRec, MarlinUtil, LCCD, CondDBMySQL,...
  - beta version already available (in marlin cvs repository)
  - will also help developers make code more compatible with current versions of operating systems, gcc versions,...
- introduce cmake makefiles
  - tool to manage complex software systems (a la gnu autotools)
  - platform independent (almost) makefiles
- improving LCIO
  - bug fixing (thanks to A.Bulgheroni)
  - more flexible (user defined persistent data)
  - direct access to events (extra event loops)

Summary

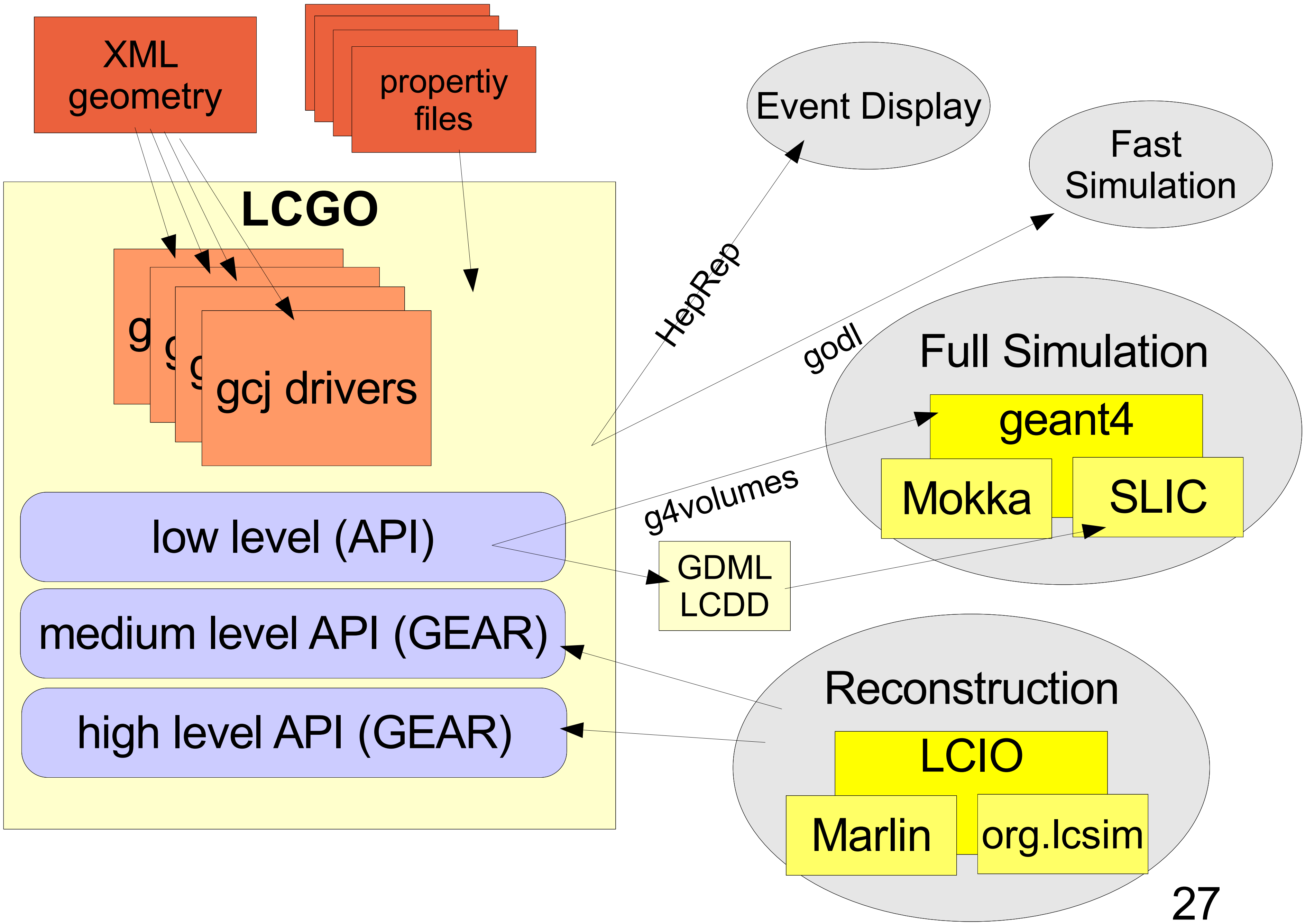
- the pixel telescope software should be written using common ILC software tools: LCIO, Marlin, LCCD, GEAR
- a lot of what is needed for JRA1 is already there
- need to evaluate (already started) the existing software and identify missing features / issues

• your input and requirements are needed to improve the software !
• let's start now

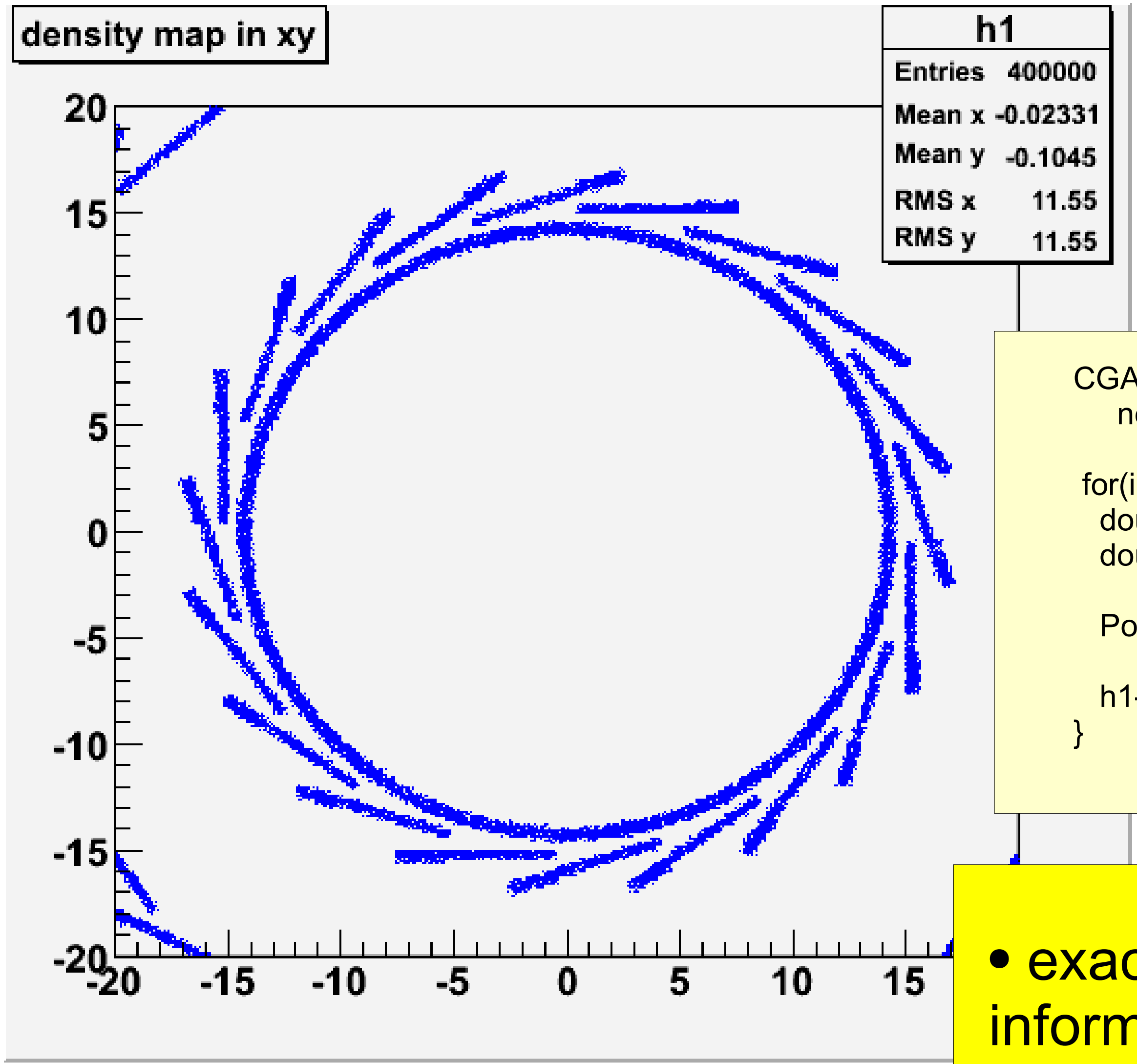
Developing the ILC software framework is an iterative procedure. By using common tools you will
• contribute to improve it
• benefit from other group's improvements

Backup Slides

LCGO implementation prelim.



CGAGear



- implemented by G.Musat, LLR
- released with v00-03

```
CGAGearPointProperties * pointProp =
    new CGAGearPointProperties(steer.str(),...);

 for(int i=0 ; i<nPoint ; ++i){
   double xr =  xmin +  ( xmax - xmin)  * random() ;
   double yr =  ymin +  ( ymax - ymin)  * random() ;

   Point3D p( xr, yr, z0 ) ;

   h1->fill(  xr, yr, pointProp->getDensity( p )  ) ;
}
```

- exact geant4 material & field information at runtime !
- performance ?
- practical issues (linking g4) ?

ILC software used in testbeams

- CALICE PPT-testbeam (now at CERN)
  - usage of LCIO, Marlin, Gear, LCCD
  - specific extensions developed by CALICE
- TPC prototypes
  - usage of LCIO, Marlin, Gear, (LCCD planned)
  - special raw data classes for Tracker hits in LCIO
  - Gear geometry description of TPC prototype
- VTX prototypes
  - usage of LCIO, Marlin, Gear
  - development of VTX geometry definition in Gear

CondDBMySQL

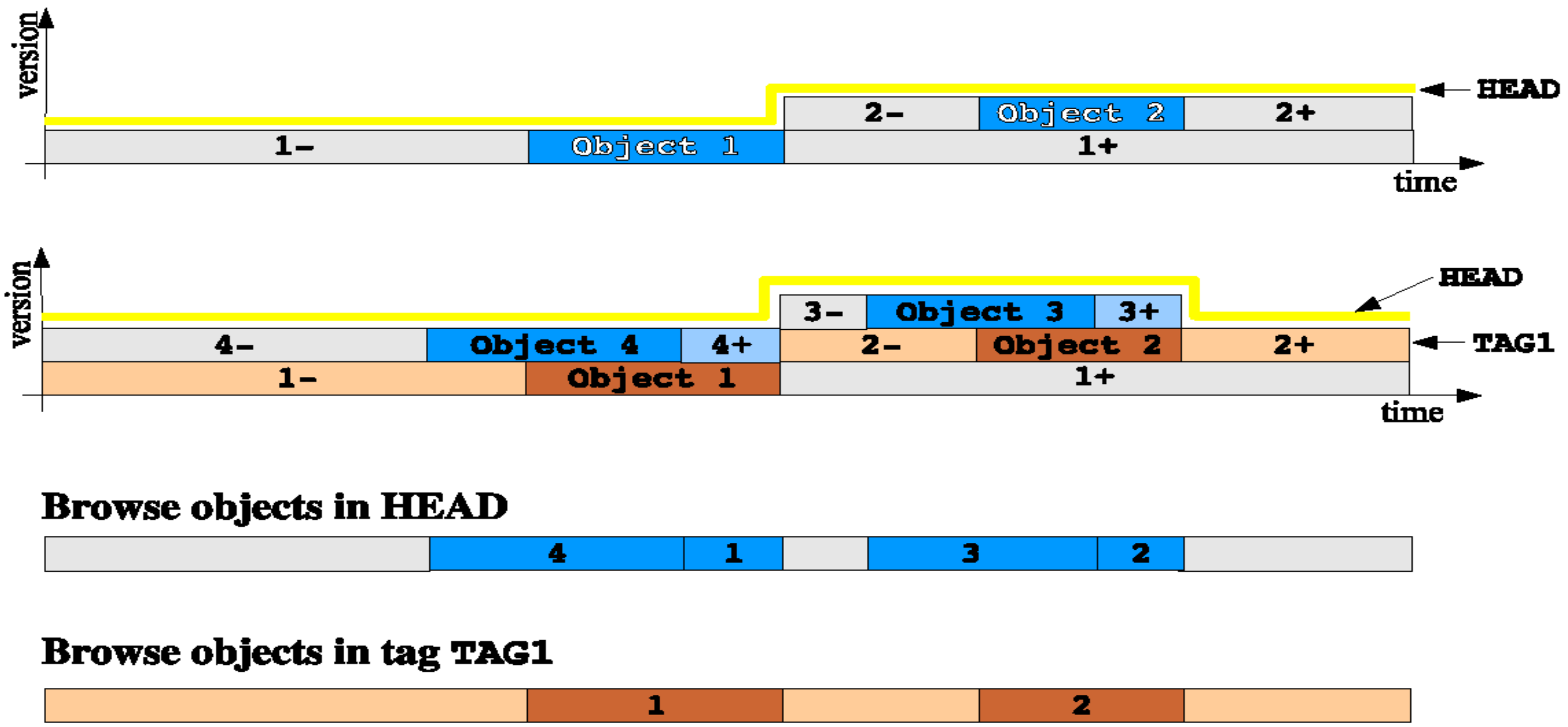


Figure 3: tagging and browsing example in the ConditionsDB mySQL's implementation.

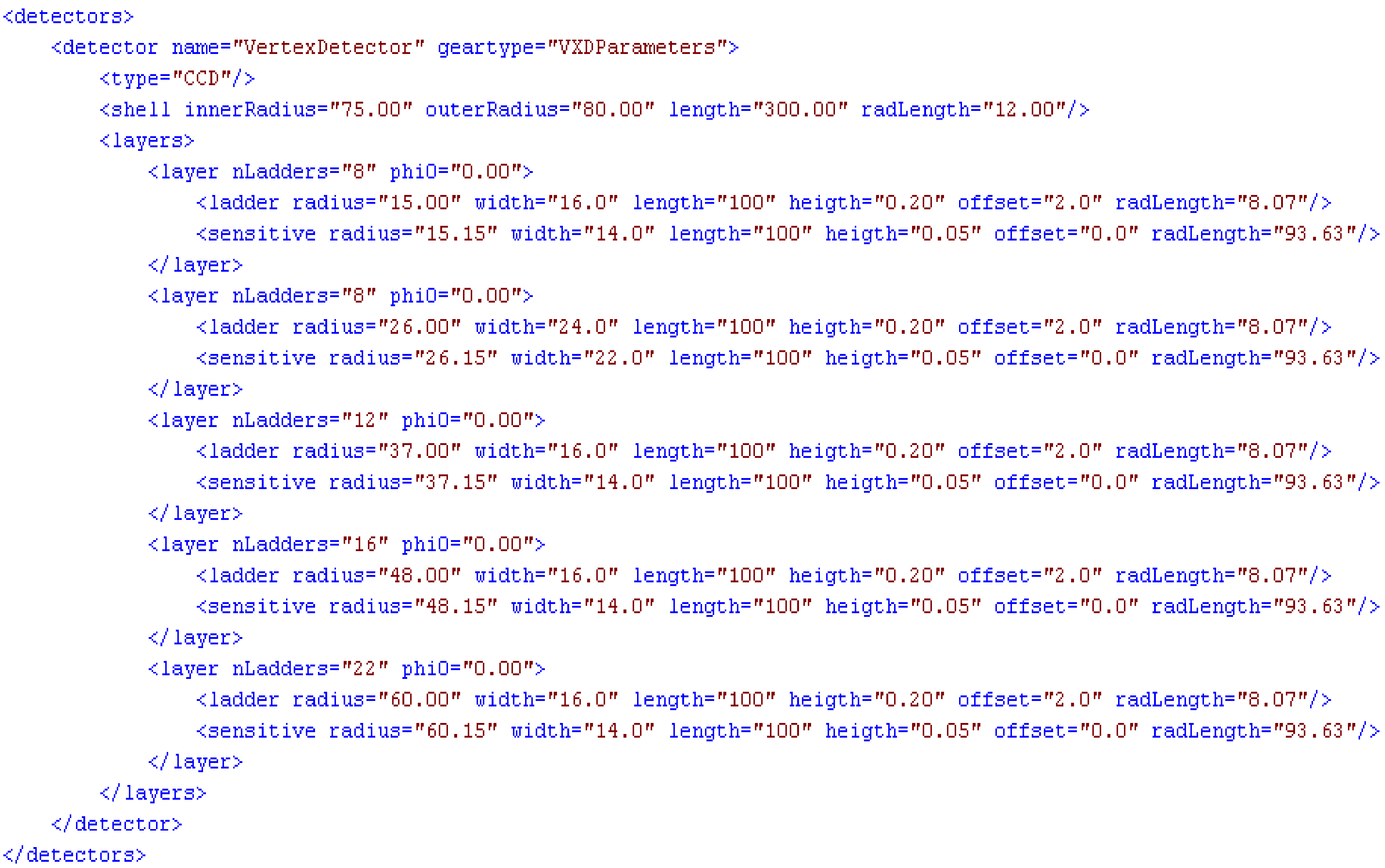
- full tagging and validity time mechanism
- developed for ATLAS by Lisbon group
- will be replaced by COOL (transparent to users of LCCD)

some LCGO planned features

- extended GEAR interface (medium and high level)
  - tracking (and clustering PFA)
    - average material volumes
    - intersection with 'next' volume
    - dE/dx
    - field maps
    - access to volumes
    - extensions of detectors (a la gear)
      - e.g. #layers, thickness, width,..

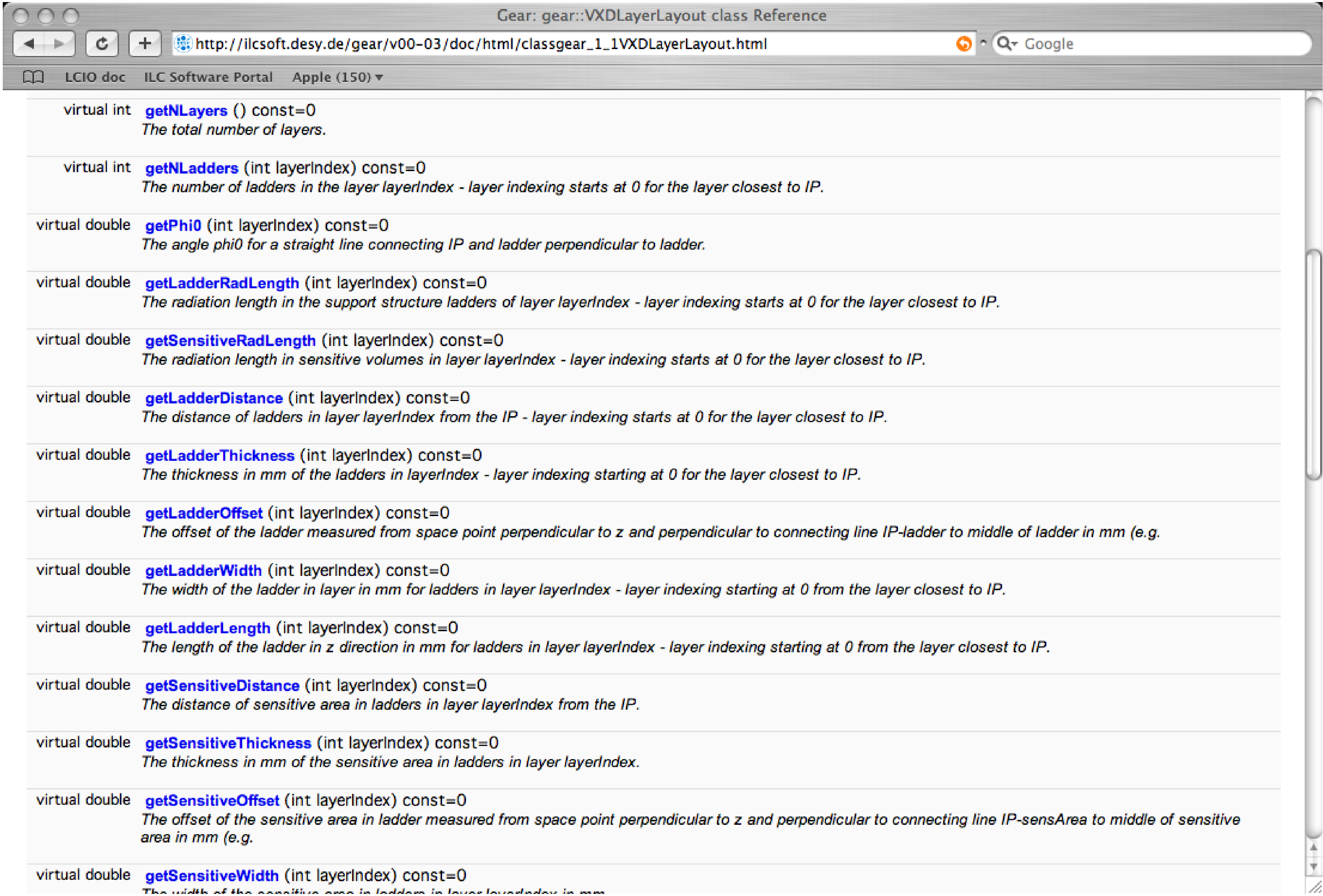
- material database
- field maps
- properties (sampling fractions)
- readout properties
  - cellId <-> position
  - cellid range (noise simulation)
  - cell sizes
  - neighbors
- Vector and Matrix classes ?
  - ThreeVector, Point3D
  - Planes, cylinders, ... ?
  - FourVector
  - SymMatrix (covariances)

VTX XML description



```xml
<detectors>
    <detector name="VertexDetector" geartype="VXDParameters">
        <type="CCD"/>
        <shell innerRadius="75.00" outerRadius="80.00" length="300.00" radLength="12.00"/>
        <layers>
            <layer nLadders="8" phi0="0.00">
                <ladder radius="15.00" width="16.0" length="100" heigth="0.20" offset="2.0" radLength="8.07"/>
                <sensitive radius="15.15" width="14.0" length="100" heigth="0.05" offset="0.0" radLength="93.63"/>
            </layer>
            <layer nLadders="8" phi0="0.00">
                <ladder radius="26.00" width="24.0" length="100" heigth="0.20" offset="2.0" radLength="8.07"/>
                <sensitive radius="26.15" width="22.0" length="100" heigth="0.05" offset="0.0" radLength="93.63"/>
            </layer>
            <layer nLadders="12" phi0="0.00">
                <ladder radius="37.00" width="16.0" length="100" heigth="0.20" offset="2.0" radLength="8.07"/>
                <sensitive radius="37.15" width="14.0" length="100" heigth="0.05" offset="0.0" radLength="93.63"/>
            </layer>
            <layer nLadders="16" phi0="0.00">
                <ladder radius="48.00" width="16.0" length="100" heigth="0.20" offset="2.0" radLength="8.07"/>
                <sensitive radius="48.15" width="14.0" length="100" heigth="0.05" offset="0.0" radLength="93.63"/>
            </layer>
            <layer nLadders="22" phi0="0.00">
                <ladder radius="60.00" width="16.0" length="100" heigth="0.20" offset="2.0" radLength="8.07"/>
                <sensitive radius="60.15" width="14.0" length="100" heigth="0.05" offset="0.0" radLength="93.63"/>
            </layer>
        </layers>
    </detector>
</detectors>
```

Gear API for VXD/Pixel-detectors II


Gear: gear::VXDLayerLayout class Reference
http://ilcsoft.desy.de/gear/v00-03/doc/html/classgear_1_1VXDLayerLayout.html
Google
LCIO doc
ILC Software Portal
Apple (150)

virtual int **getNLayers** () const=0
The total number of layers.

virtual int **getNLadders** (int layerIndex) const=0
The number of ladders in the layer layerIndex - layer indexing starts at 0 for the layer closest to IP.

virtual double **getPhi0** (int layerIndex) const=0
The angle phi0 for a straight line connecting IP and ladder perpendicular to ladder.

virtual double **getLadderRadLength** (int layerIndex) const=0
The radiation length in the support structure ladders of layer layerIndex - layer indexing starts at 0 for the layer closest to IP.

virtual double **getSensitiveRadLength** (int layerIndex) const=0
The radiation length in sensitive volumes in layer layerIndex - layer indexing starts at 0 for the layer closest to IP.

virtual double **getLadderDistance** (int layerIndex) const=0
The distance of ladders in layer layerIndex from the IP - layer indexing starts at 0 for the layer closest to IP.

virtual double **getLadderThickness** (int layerIndex) const=0
The thickness in mm of the ladders in layerIndex - layer indexing starting at 0 for the layer closest to IP.

virtual double **getLadderOffset** (int layerIndex) const=0
The offset of the ladder measured from space point perpendicular to z and perpendicular to connecting line IP-ladder to middle of ladder in mm (e.g.

virtual double **getLadderWidth** (int layerIndex) const=0
The width of the ladder in layer in mm for ladders in layer layerIndex - layer indexing starting at 0 from the layer closest to IP.

virtual double **getLadderLength** (int layerIndex) const=0
The length of the ladder in z direction in mm for ladders in layer layerIndex - layer indexing starting at 0 from the layer closest to IP.

virtual double **getSensitiveDistance** (int layerIndex) const=0
The distance of sensitive area in ladders in layer layerIndex from the IP.

virtual double **getSensitiveThickness** (int layerIndex) const=0
The thickness in mm of the sensitive area in ladders in layer layerIndex.

virtual double **getSensitiveOffset** (int layerIndex) const=0
The offset of the sensitive area in ladder measured from space point perpendicular to z and perpendicular to connecting line IP-sensArea to middle of sensitive area in mm (e.g.

virtual double **getSensitiveWidth** (int layerIndex) const=0
The width of the sensitive area in ladders in layer layerIndex in mm