The VTX Detector in the ILC Software Framework

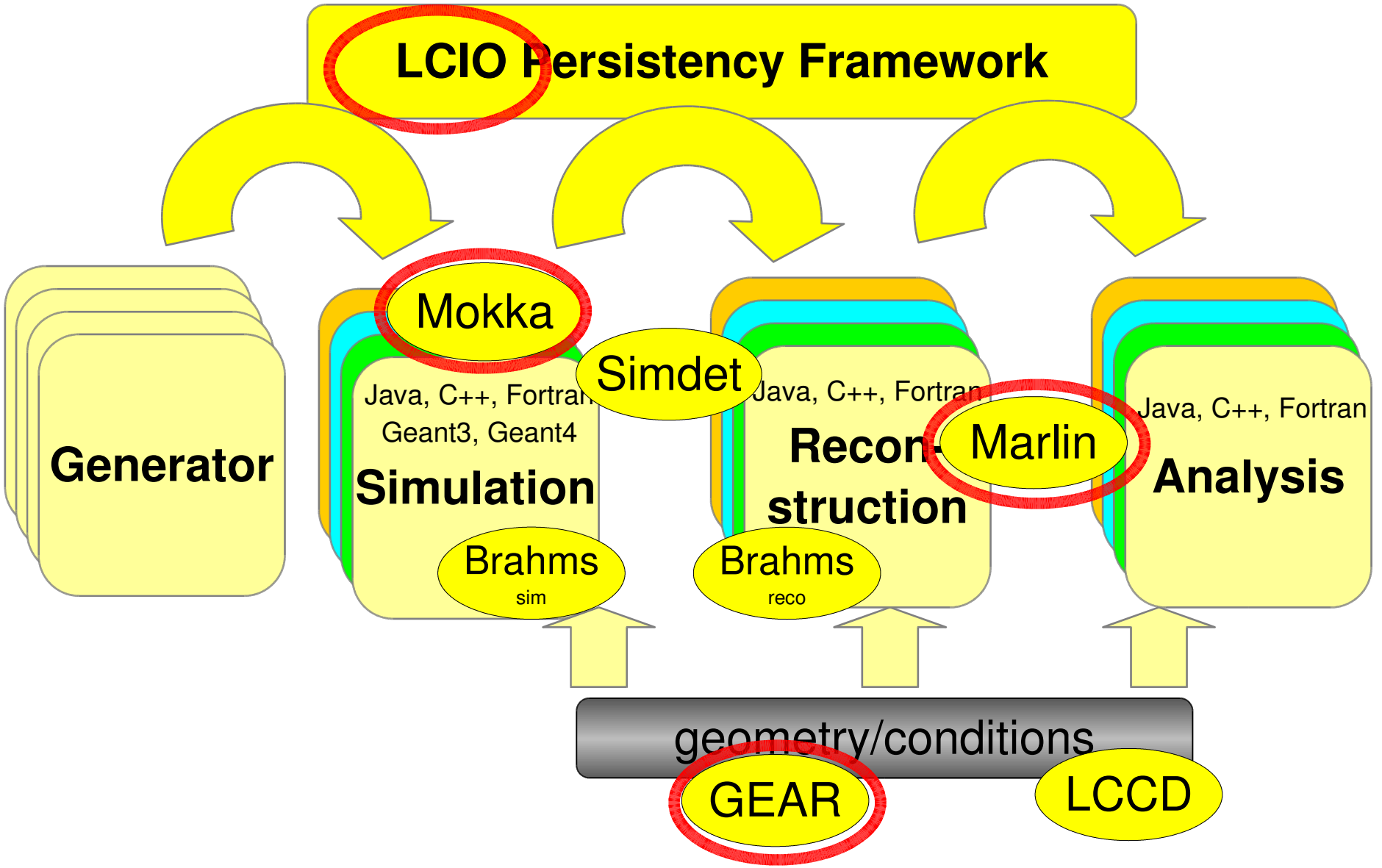
Frank Gaede
DESY

ILC-VTX Workshop, Ringberg Castle
May 28-31, 2006

Outline

- Introduction to ILC core software tools:
  - LCIO – ILC persistency and event data model
  - Marlin/MarlinReco – C++ reconstruction framework
  - GEAR – geometry description
  - (Mokka – geant4 full simulation)
- LCIO classes relevant for VTX detectors:
  - raw data classes
  - a vertex class in LCIO
- VTX geometry description in GEAR
  - detector description
  - material properties

Overview of LDC software tools



LCIO overview

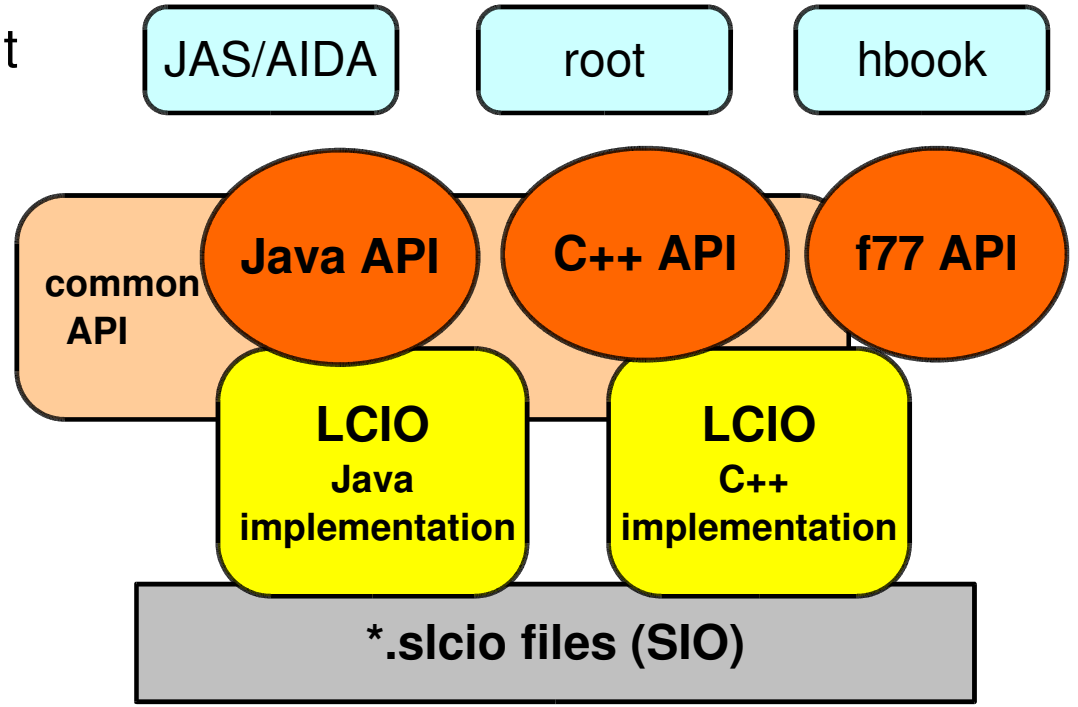
- DESY and SLAC joined project:
  - provide common basis for ILC software
- Features:
  - Java, C++ and f77 (!) API
  - extensible data model for current and future simulation and testbeam studies
  - user code separated from concrete data format
  - no dependency on other frameworks

simple & lightweight

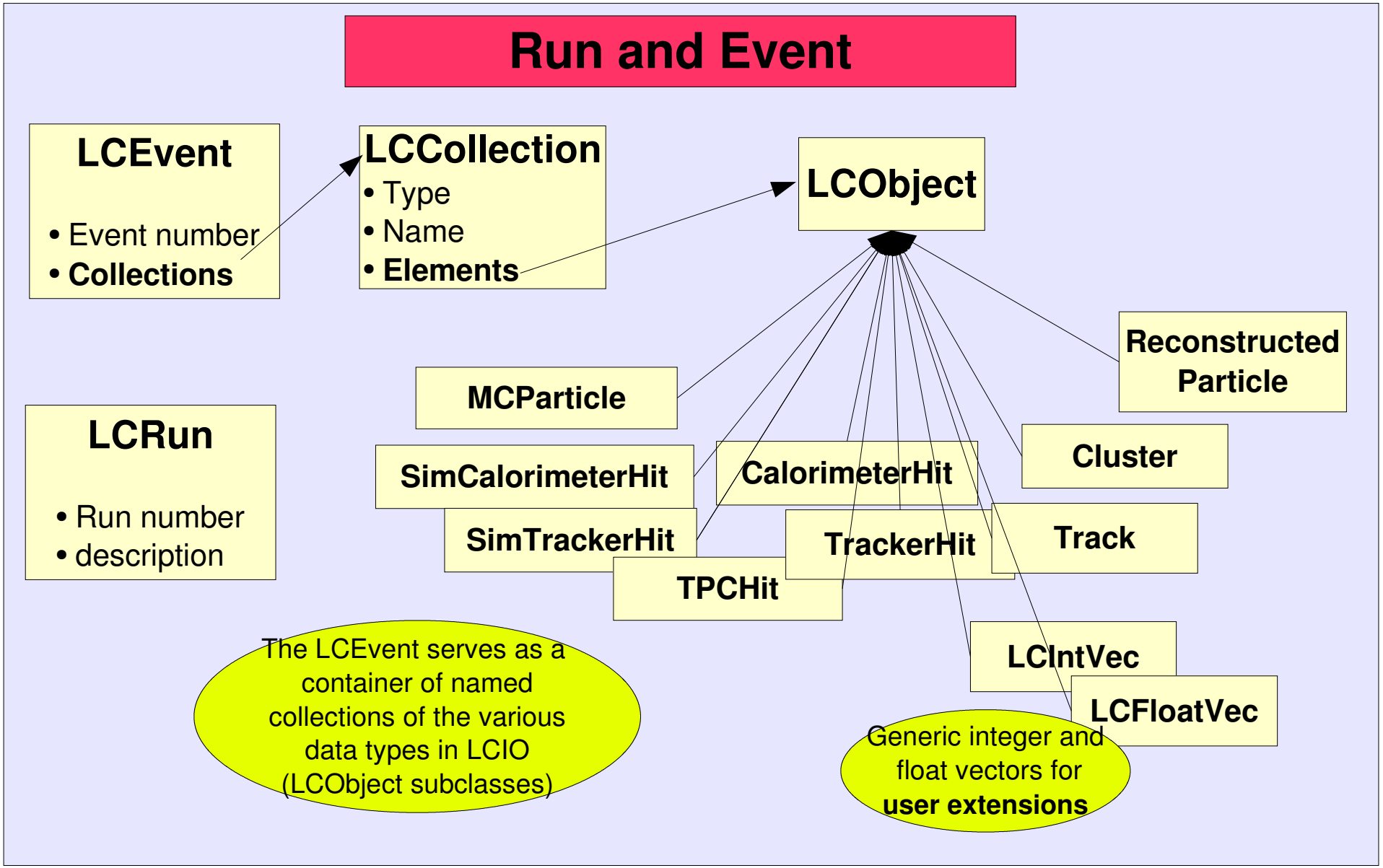
current release: **v01-07**

now de facto standard persistency & datamodel for ILC software

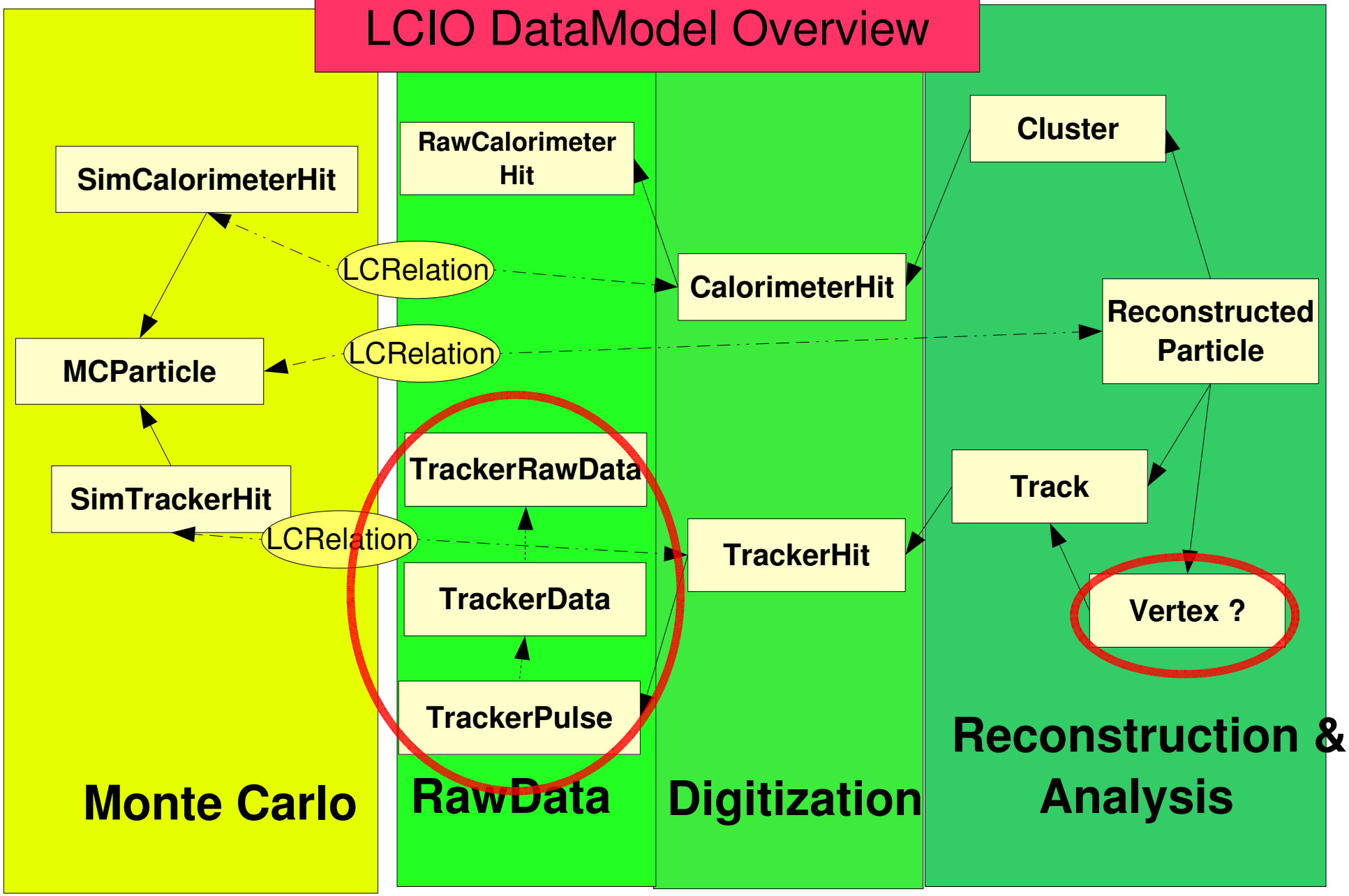
SW-Architecture



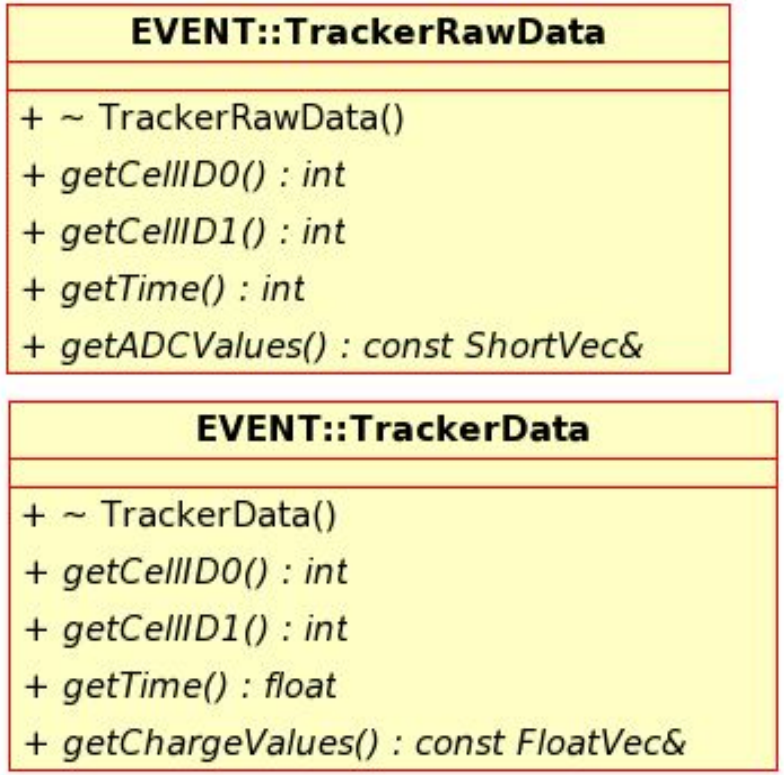
Run and Event

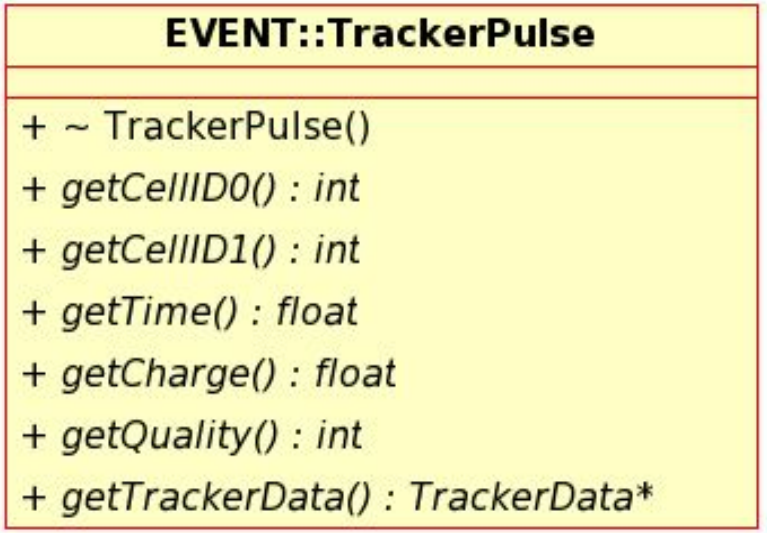


LCIO DataModel Overview



LCIO raw data classes for tracker





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 should look into using these classes !
- provide feedback/feature requests

Proposal for an LCIO vertex class

EVENT::Vertex

+ ~ Vertex()
+ *getMomentum() : const float**
+ *getMass() : float*
+ *getCharge() : float*
+ *getPosition() : const float**
+ *getCovMatrix() : const FloatVec&*
+ *getChi2() : float*
+ *getProbability() : float*
+ *getPreviousVertex() : Vertex**
+ *getParameters() : const FloatVec&*
+ *getTracks() : const TrackVec&*
+ *addTrack(track : Track*) : void*

EVENT::ReconstructedParticle

+ ~ ReconstructedParticle()
+ *getType() : int*
+ *isCompound() : bool*
+ *getMomentum() : const double**
+ *getEnergy() : double*
+ *getCovMatrix() : const FloatVec&*
+ *getMass() : double*
+ *getCharge() : float*
+ *getReferencePoint() : const float**
+ *getParticleIDs() : const ParticleIDVec&*
+ *getParticleIDUsed() : ParticleID**
+ *getGoodnessOfPID() : float*
+ *getParticles() : const ReconstructedParticleVec&*
+ *getClusters() : const ClusterVec&*
+ *getTracks() : const TrackVec&*
+ *addParticleID(pid : ParticleID*) : void*
+ *addParticle(particle : ReconstructedParticle*) : void*
+ *addCluster(cluster : Cluster*) : void*
+ *addTrack(track : Track*) : void*

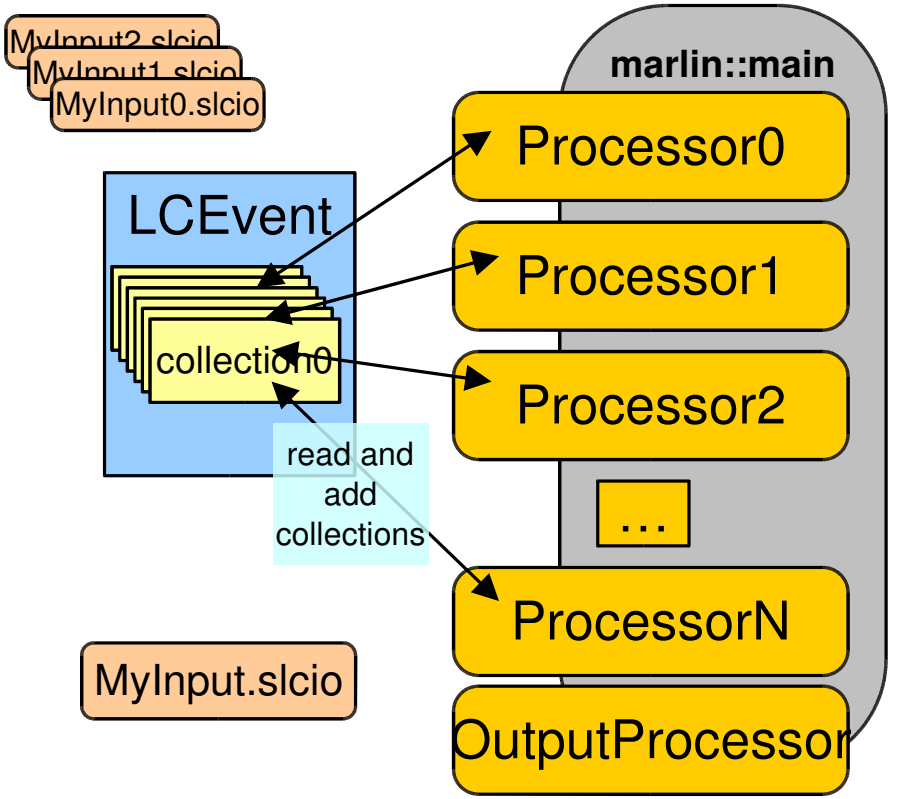
- original LCIO idea:
  use ReconstructedParticle also for compound objects like jets and vertices
- LCFI proposes dedicated vertex class
- need to optimize to avoid overlap and
- redundancy with ReconstructedParticle

http://forum.linearcollider.org

Marlin

Modular**A**nalysis & **R**econstruction for the **L I N**ear Collider

- modular C++ **application framework** for the analysis and reconstruction of LCIO data
- uses LCIO as transient data model
- software modules called Processors
- provides main program !
- provides simple user steering:
  - program flow (active processors)
  - user defined variables
    - per processor and global
  - input/output files



MarlinReco

- **TrackDigi**
  - TPCDigi
  - new VTXDigi — talk: A.Raspereza
- **CaloDigi**
  - LDCCaloDigi
- **Tracking**
  - LEPTracking
  - new VTXTracking
  - TrackCheater
- **Clustering**
  - TrackwiseClustering
  - ClusterCheater
- **Pflow**
  - Wolf
- **Analysis**
  - EventShapes
  - SatoruJetFinder

- MarlinReco is a Marlin based full reconstruction toolkit
- it can be integrated/combined with **any other Marlin processor**
- it uses **GEAR** for the geometry description

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" type
        <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 – under development
- **abstract interface** (a la LCIO)
  - concrete implementation based on XML files
  - and Mokka-CGA – under development

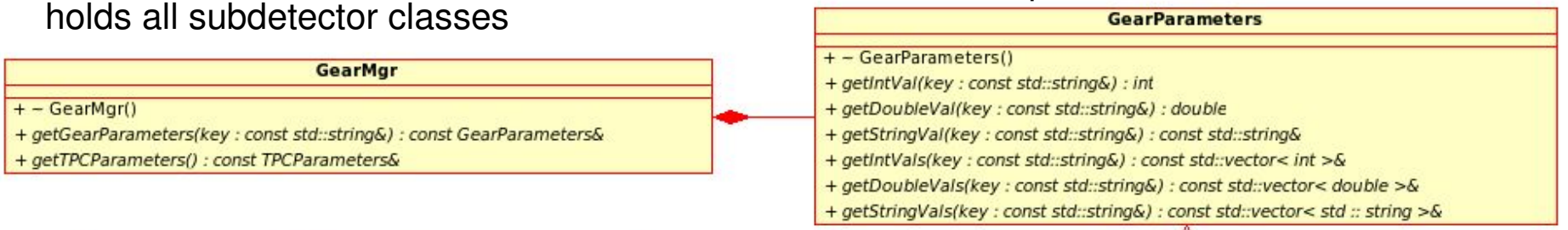
GEAR – Classes

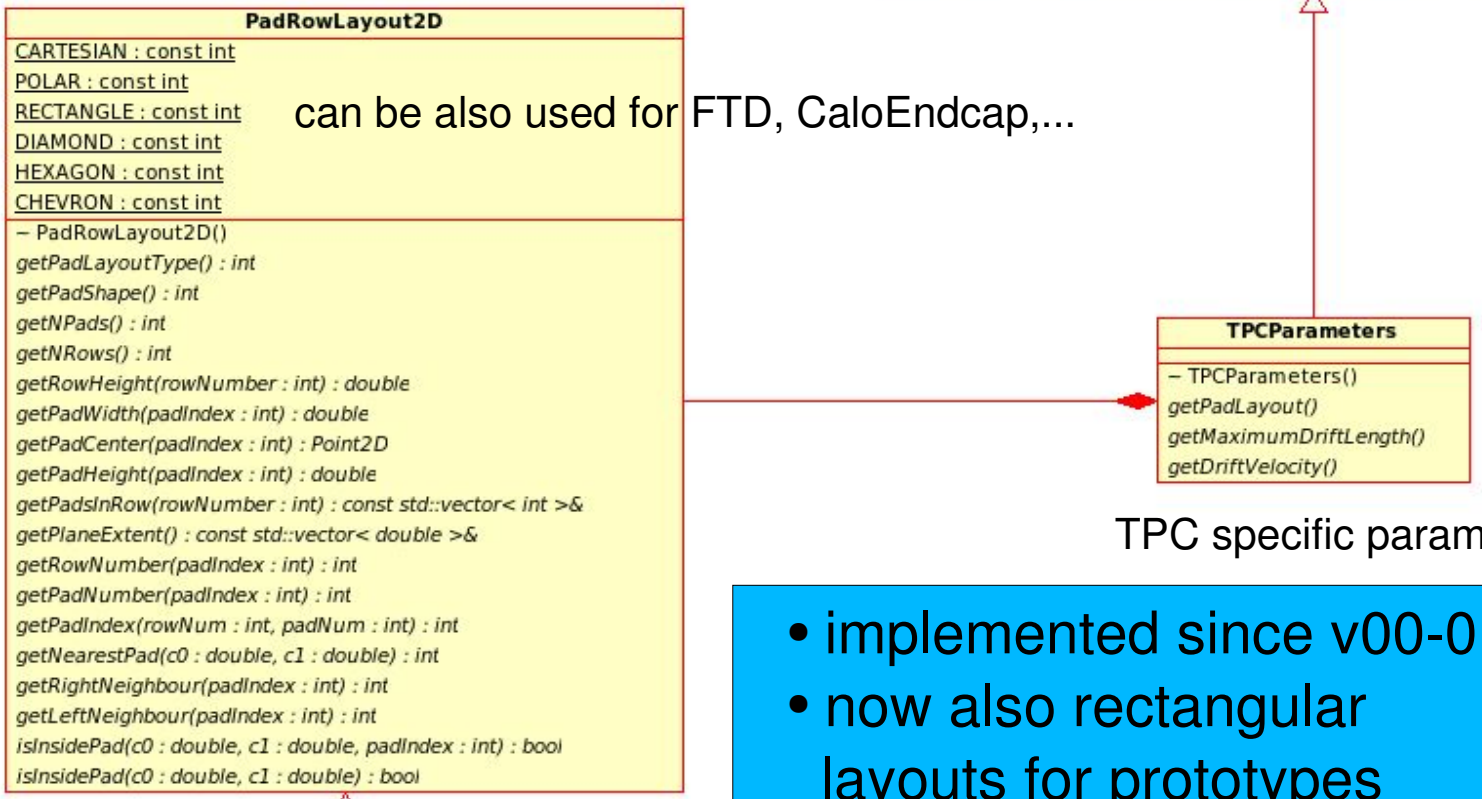
- Subdetector description
  - high level description of detector shape and readout geometry – one class for every subdetector type, e.g.
    - TPC, Ecal, Hcal (MainCalorimeter), FTD, VTX, SIT, ...
    - defines required attributes - as detailed as necessary but as abstract as possible
    - allows to add additional named attributes
  - use XML files
- Material properties
  - point properties (density, material, radlen,...)
  - distance properties integrated along (straight!?) path
  - use Mokka-CGA interface to geant4 geometry !?

GEAR example:TPC description

named parameters for additional attributes

holds all subdetector classes

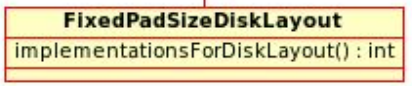




can be also used for FTD, CaloEndcap,...

TPC specific parameters

- implemented since v00-01
- now also rectangular layouts for prototypes



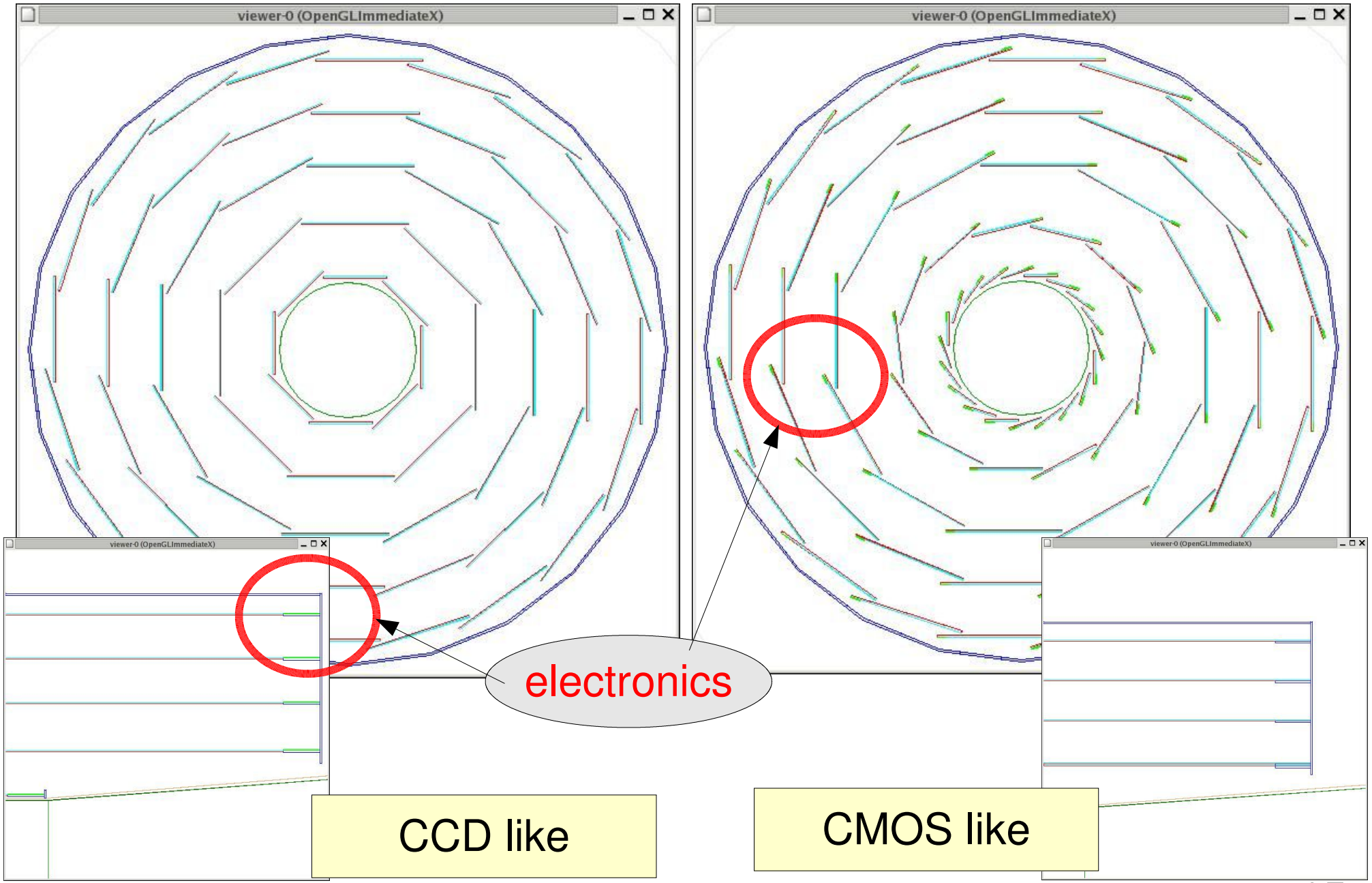
implementation for disk with pad rings

describing the VTX geometry

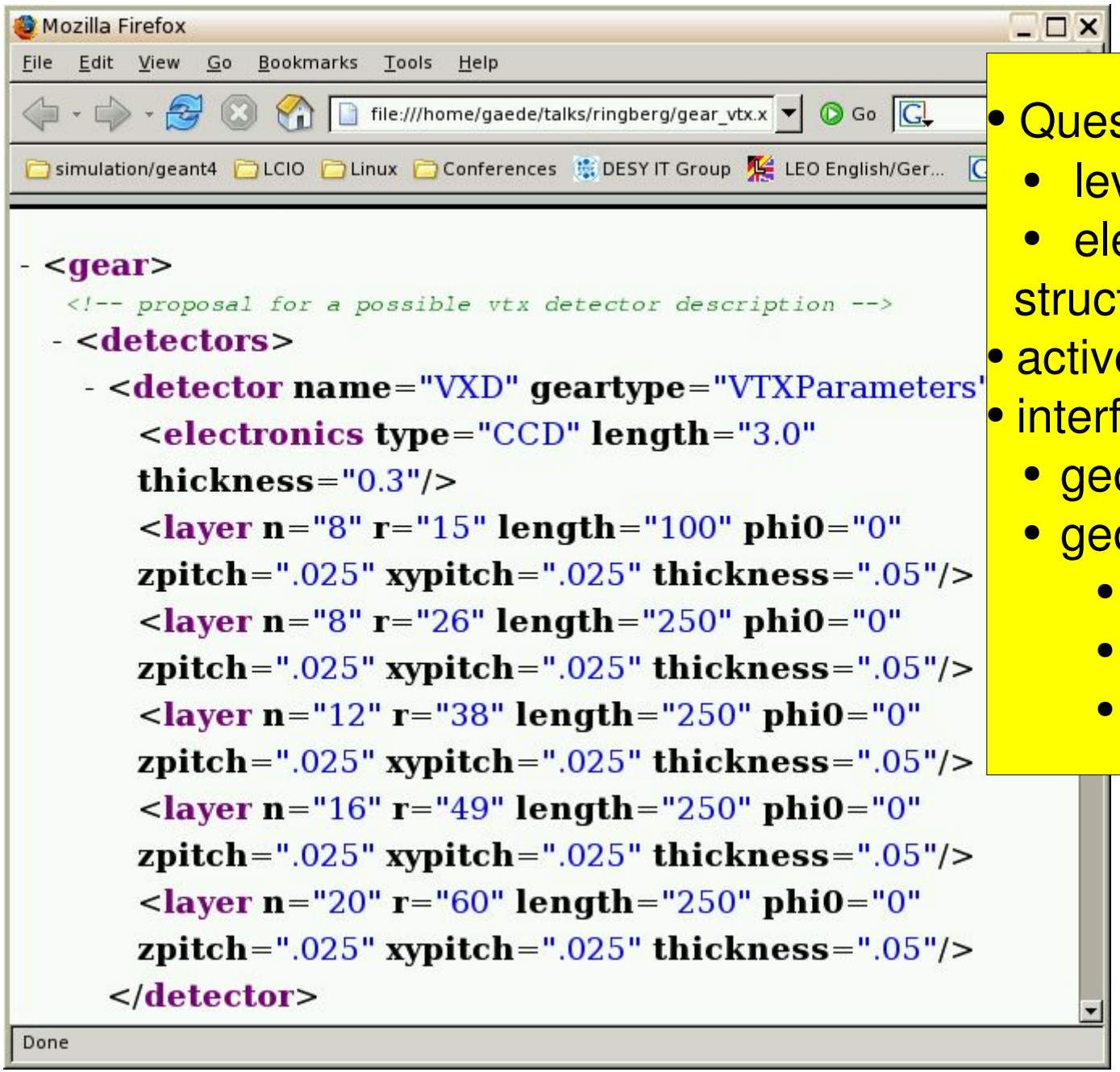
- Mokka full simulation VTX:
  - until recently only simple cylinders used
  - improved now (see next talk by D.Grandjean)
- need to find abstract description of VTX for GEAR
  - could use Mokka parameterized drivers as starting point
  - what level of detail is needed for VTX
    - digitization ?
    - track reconstruction ?
  - need input from people working on VTX tracking !

use this workshop to settle coarse scheme

VTX geometry types

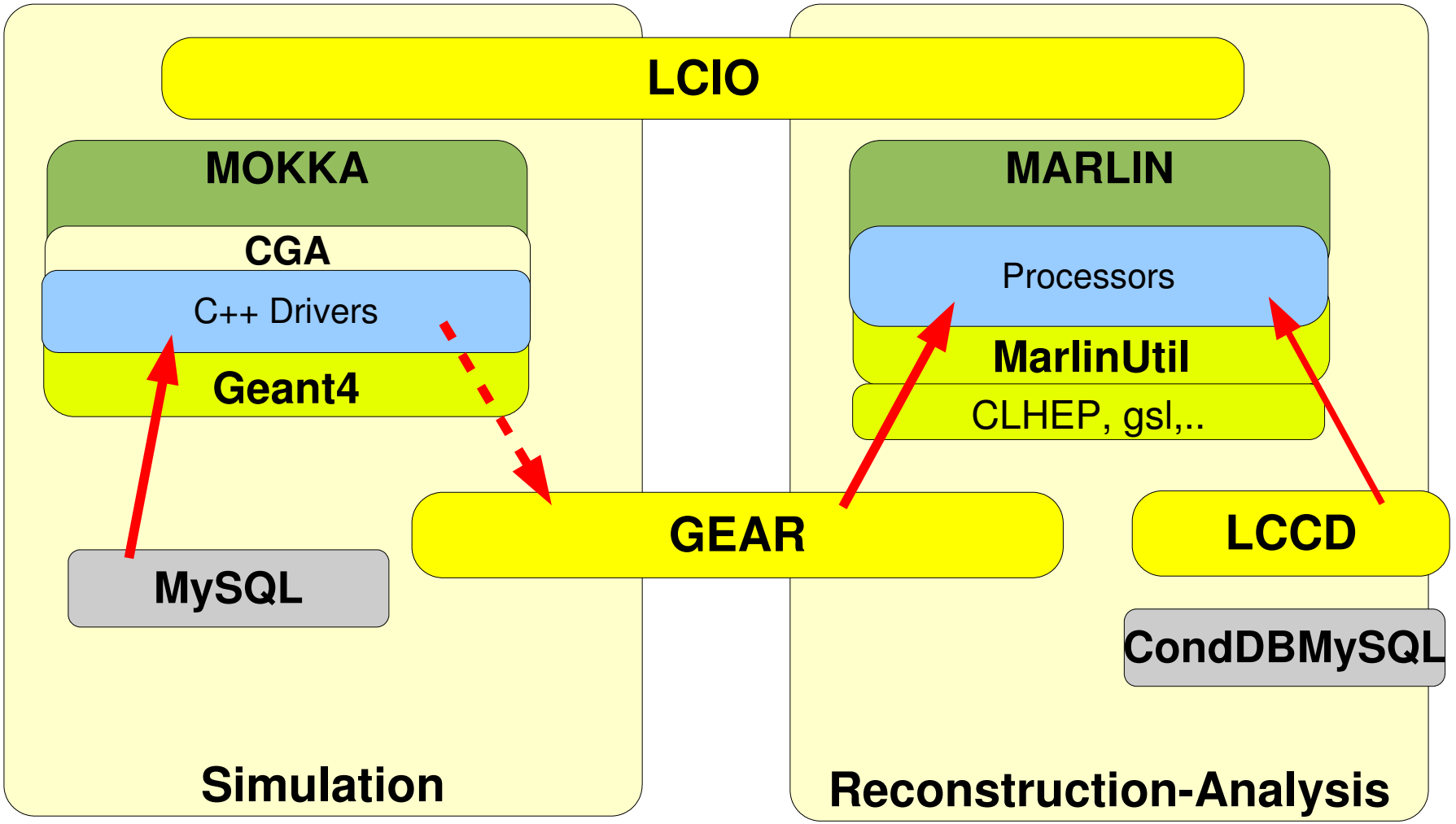


possible gear VTX xml description

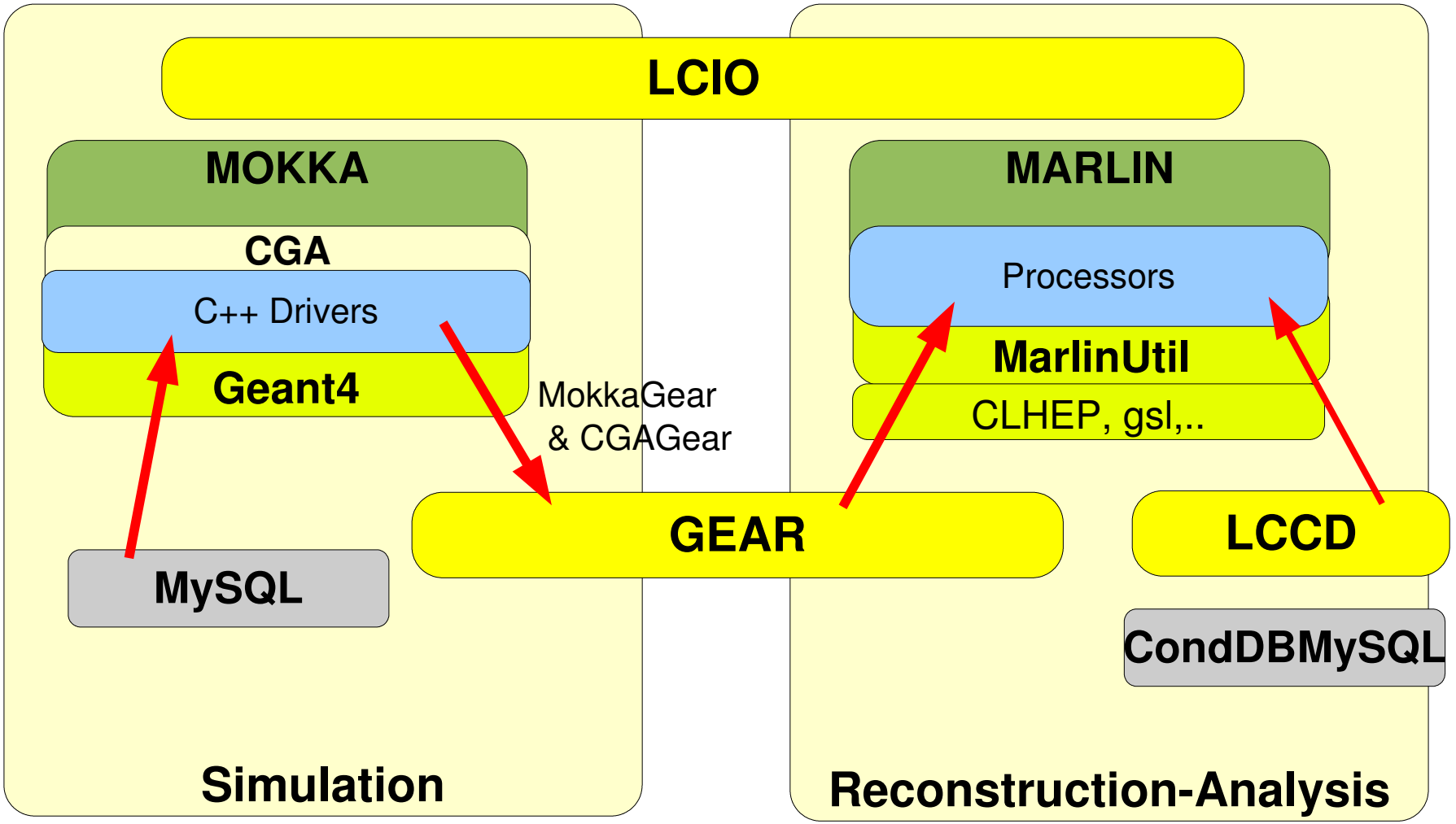


- Questions:
  - level of detail:
  - electronics, support structure
- active/inactive silicon
- interface:
  - geometry properties
  - geometry helpers, e.g.
    - distanceToNextLayer()
    - isPointInLayer()
    - ...

LDC simulation framework



LDC simulation framework



MokkaGear

- extension to Mokka (R.Lippe – Diploma Thesis)
- extract geometry information in drivers when detector is built
- use Gear to create XML files for reconstruction
- currently implemented:
  - TPC (tpc04), Ecal (ecal02) and Hcal (hcal04)
- to be released soon (next Mokka release 6.1)
- optional feature
  - only if Gear is installed and included

aim: have only one source of of information
for describing the detector geometry !

GEAR – material properties

GearDistanceProperties

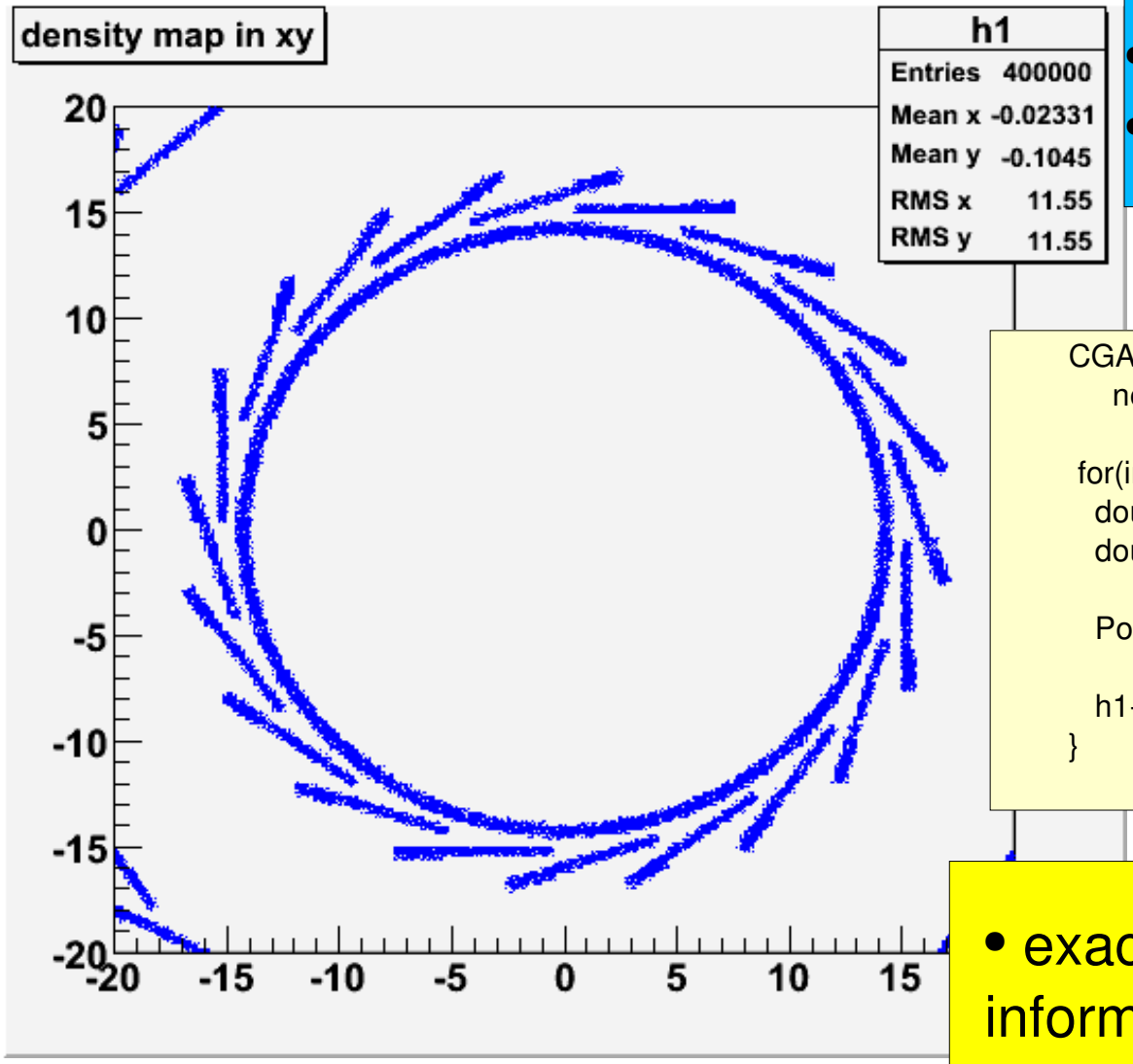
- GearDistanceProperties()
getMaterialNames(p0 : const Point3D&, p1 : const Point3D&) : const std::vector< std :: string >&
getMaterialThicknesses(p0 : const Point3D&, p1 : const Point3D&) : const std::vector< double >&
getNRadlen(p0 : const Point3D&, p1 : const Point3D&) : double
getNIntlen(p0 : const Point3D&, p1 : const Point3D&) : double
getBdL(pos : const Point3D&) : double
getEdL(pos : const Point3D&) : double

GearPointProperties

- GearPointProperties()
getCellID(pos : const Point3D&) : int
getMaterialName(pos : const Point3D&) : const std::string&
getDensity(pos : const Point3D&) : double
getTemperature(pos : const Point3D&) : double
getPressure(pos : const Point3D&) : double
getRadlen(pos : const Point3D&) : double
getIntlen(pos : const Point3D&) : double
getLocalPosition(pos : const Point3D&) : Point3D
getB(pos : const Point3D&) : double
getE(pos : const Point3D&) : double
getListOfLogicalVolumes(pos : const Point3D&) : std::vector< std :: string >
getListOfPhysicalVolumes(pos : const Point3D&) : std::vector< std :: string >
getRegion(pos : const Point3D&) : std::string
isTracker(pos : const Point3D&) : bool
isCalorimeter(pos : const Point3D&) : bool

- proposal since Argonne Simulation Meeting 2004
- implementation with Mokka CGA under development

CGAGear



- implemented by G.Musat, LLR
- to be released soon

```
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 information at runtime !
- performance ?
- practical issues (linking g4) ?

Summary

- the VTX detector is now becoming to be treated properly in the ILC software framework:
  - improved full simulation in Mokka
  - vertex detector raw data classes in LCIO
  - a vertex class for LCIO under development
  - proper digitization and reconstruction code in Marlin (talks: M. Bataglia, A. Raspereza)
- still lacking: GEAR API and xml description of VTX

your input is needed !

- Outlook:
  - have implementation of MokkaGear for VTX to automatically create xml description

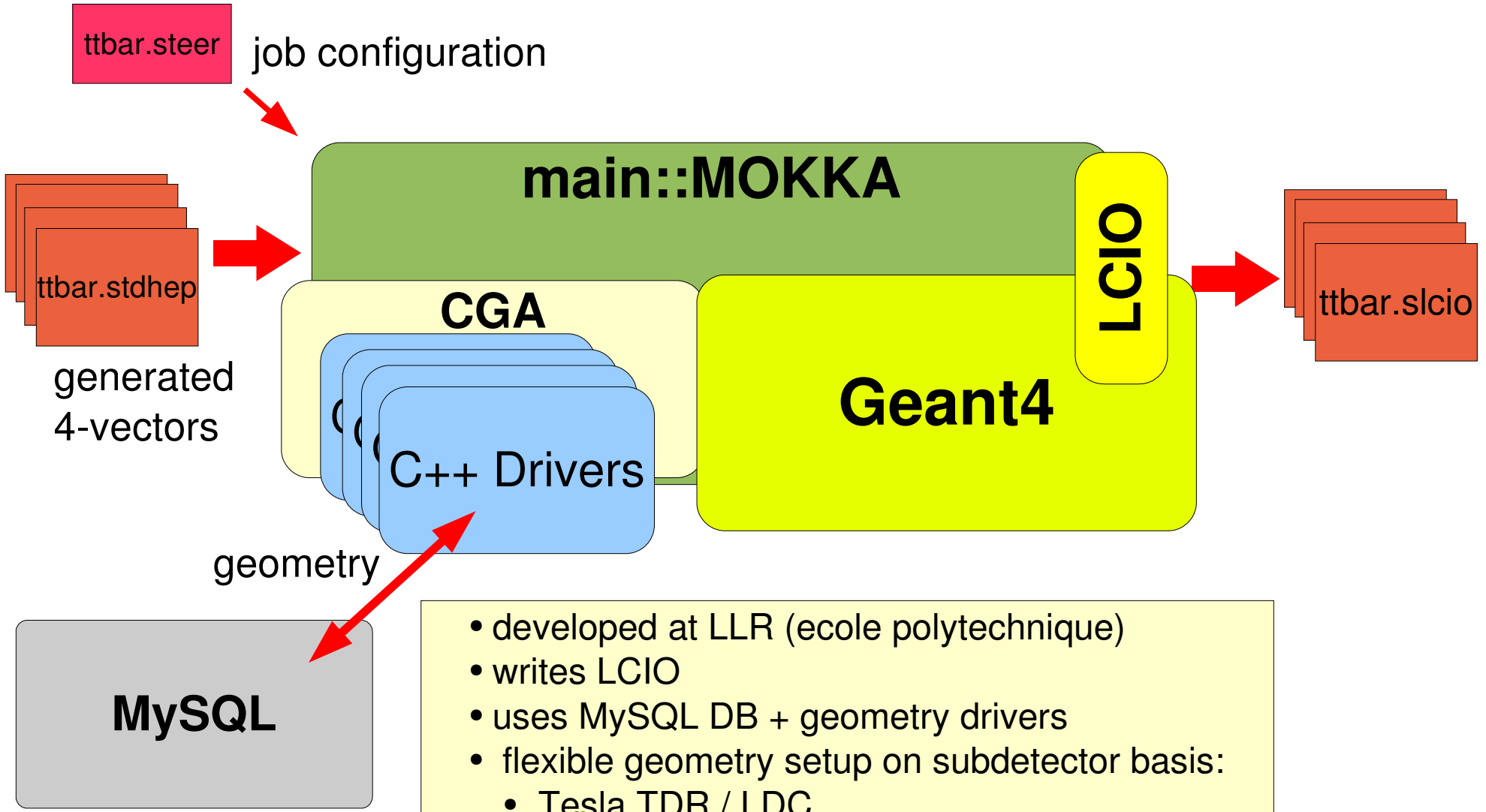
go to the software portal for more information:
http://ilcsoft.desy.de

backup slides ...

Gear status

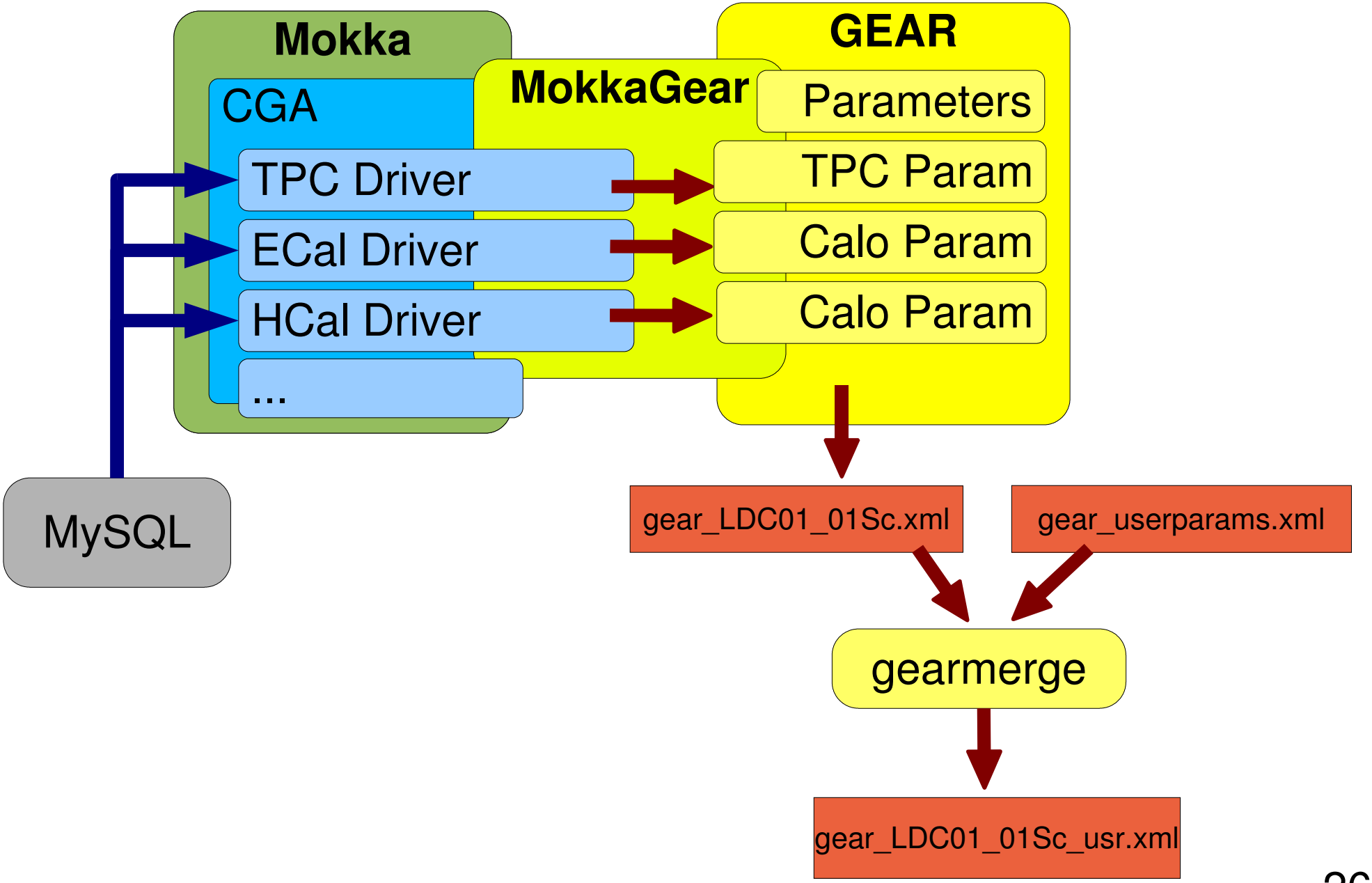
- Gear proposed at DESY simws 2005
  - first version v00-01 for snowmass
- version v00-02 (since vienna 2005):
  - TPC, Hcal, Ecal interfaces defined and implemented
  - user parameters
- current 'release' v00-03-beta:
  - also write xml files from parameters in memory
  - tool to merge files: gearmerge
  - description of TPC prototypes (rectangular pad plane)
- release v00-03
  - possiby first version of VTX description !?

Mokka overview



- developed at LLR (ecole polytechnique)
- writes LCIO
- uses MySQL DB + geometry drivers
- flexible geometry setup on subdetector basis:
  - Tesla TDR / LDC
  - SiD
  - Calice testbeam prototypes
- used in European/ LDC study

MokkaGear


Mokka
CGA
TPC Driver
ECal Driver
HCal Driver
...
MokkaGear
GEAR
Parameters
TPC Param
Calo Param
Calo Param
MySQL
gear_LDC01_01Sc.xml
gear_userparams.xml
gearmerge
gear_LDC01_01Sc_usr.xml