GEAR

Geometry API for Reconstruction

Frank Gaede

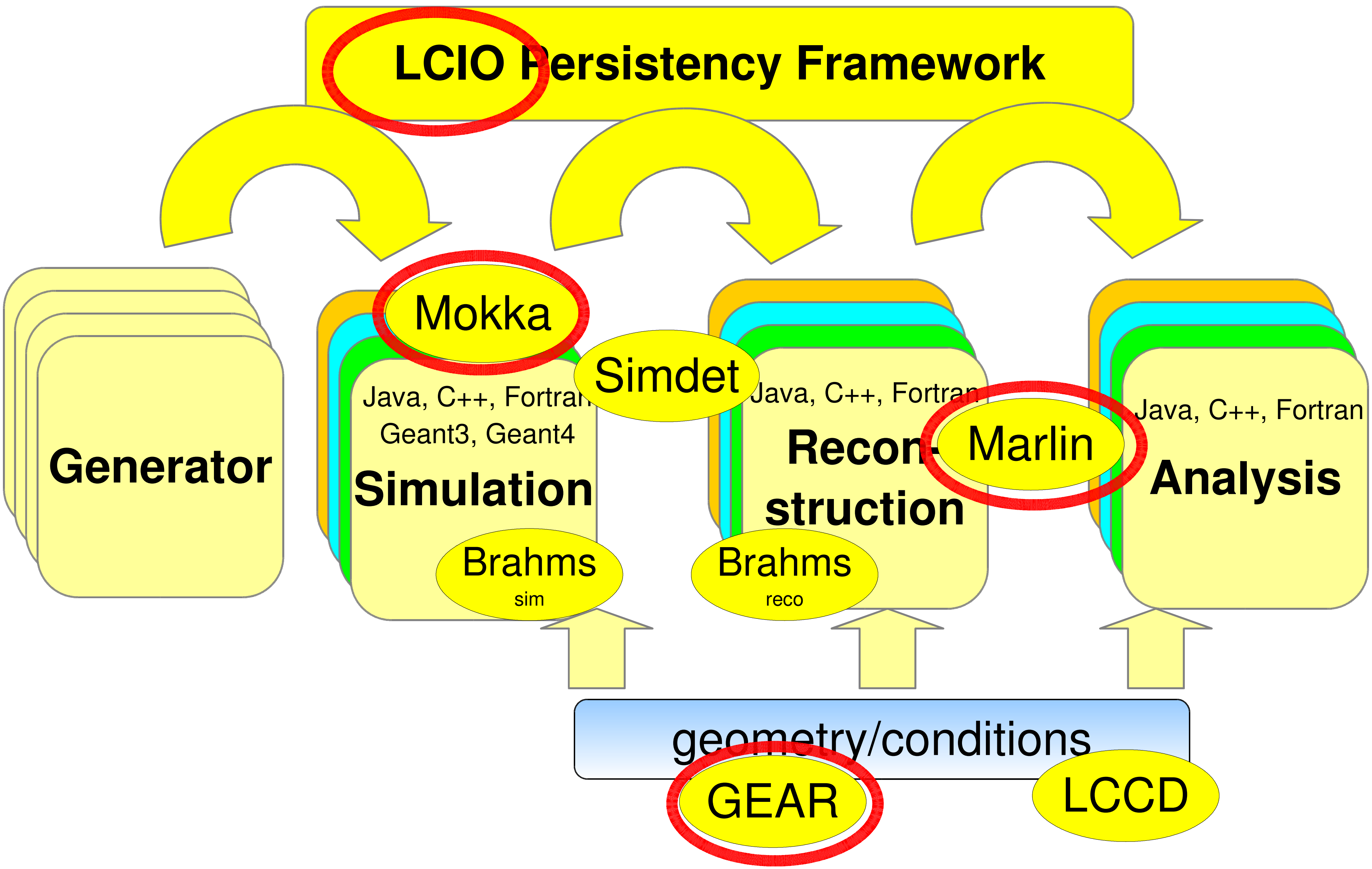
Common Geometry Meeting

SLAC Sep 18-22, 2006

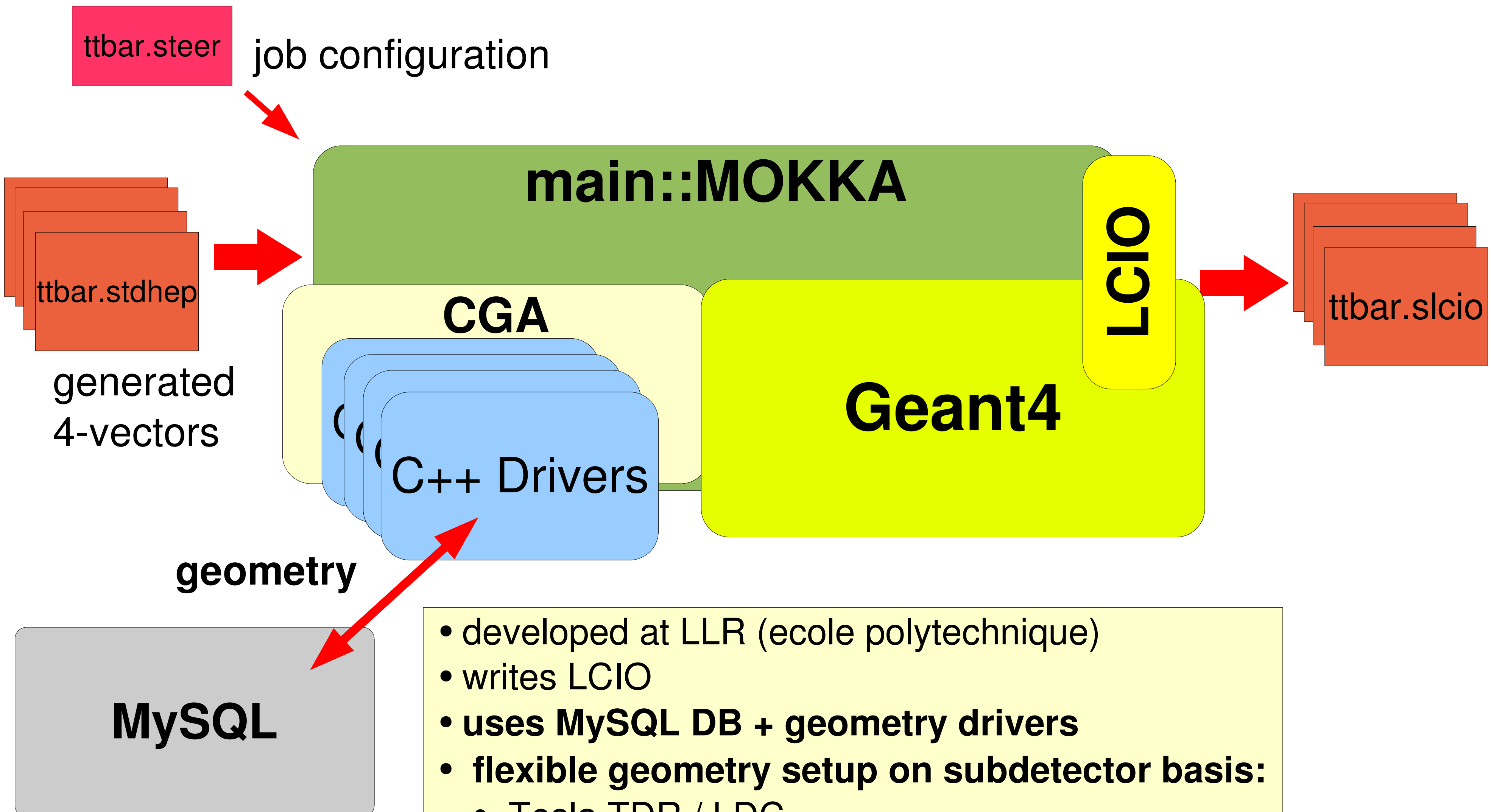
Outline

- Detector geometry description in ILC software
- GEAR introduction
  - detector description
  - material properties
- new developments
  - MokkaGear
  - GearCGA
  - VXD geometry description in GEAR (R.Lippe)

Overview of LDC software tools

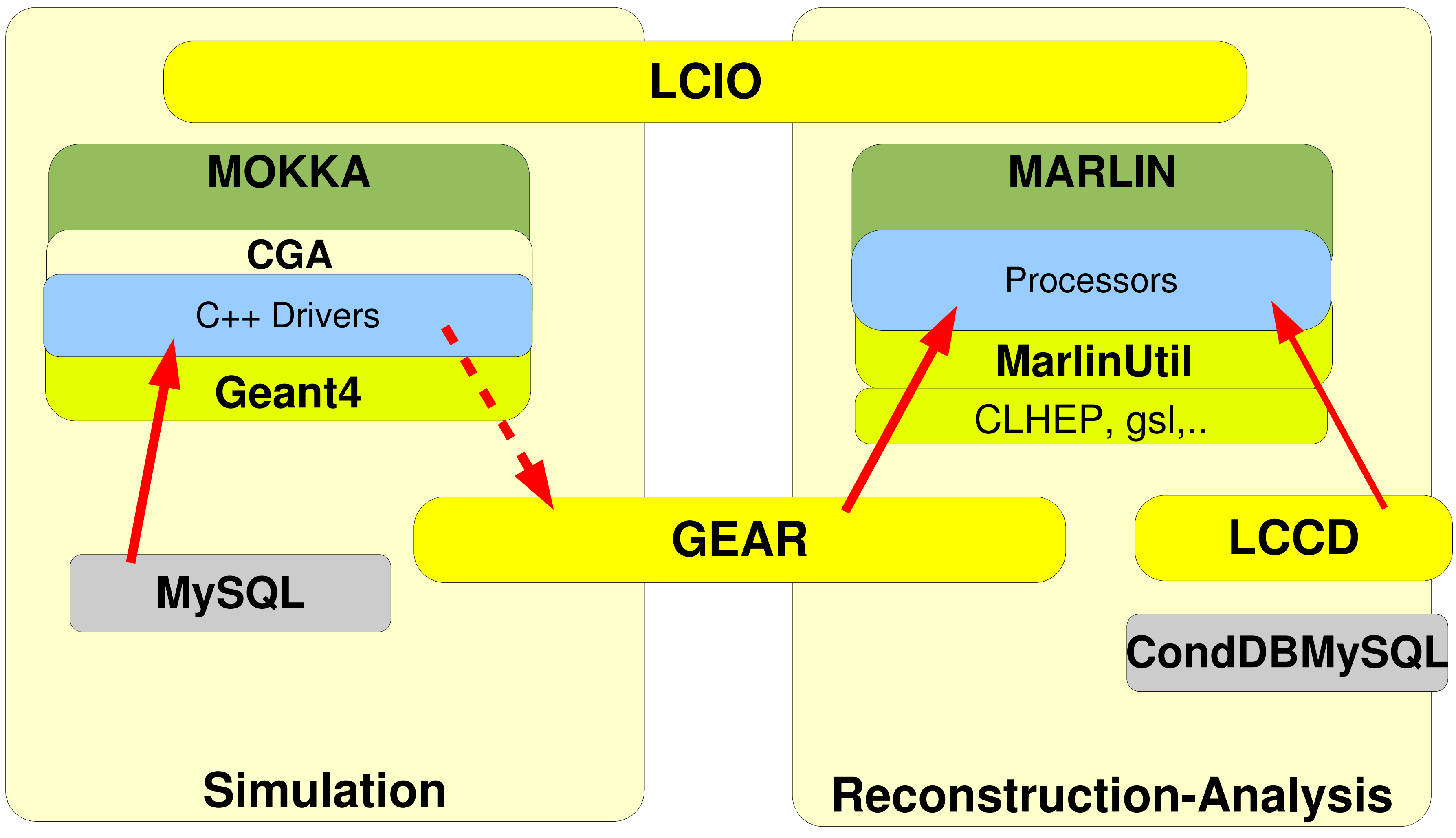


Mokka Geometry definition



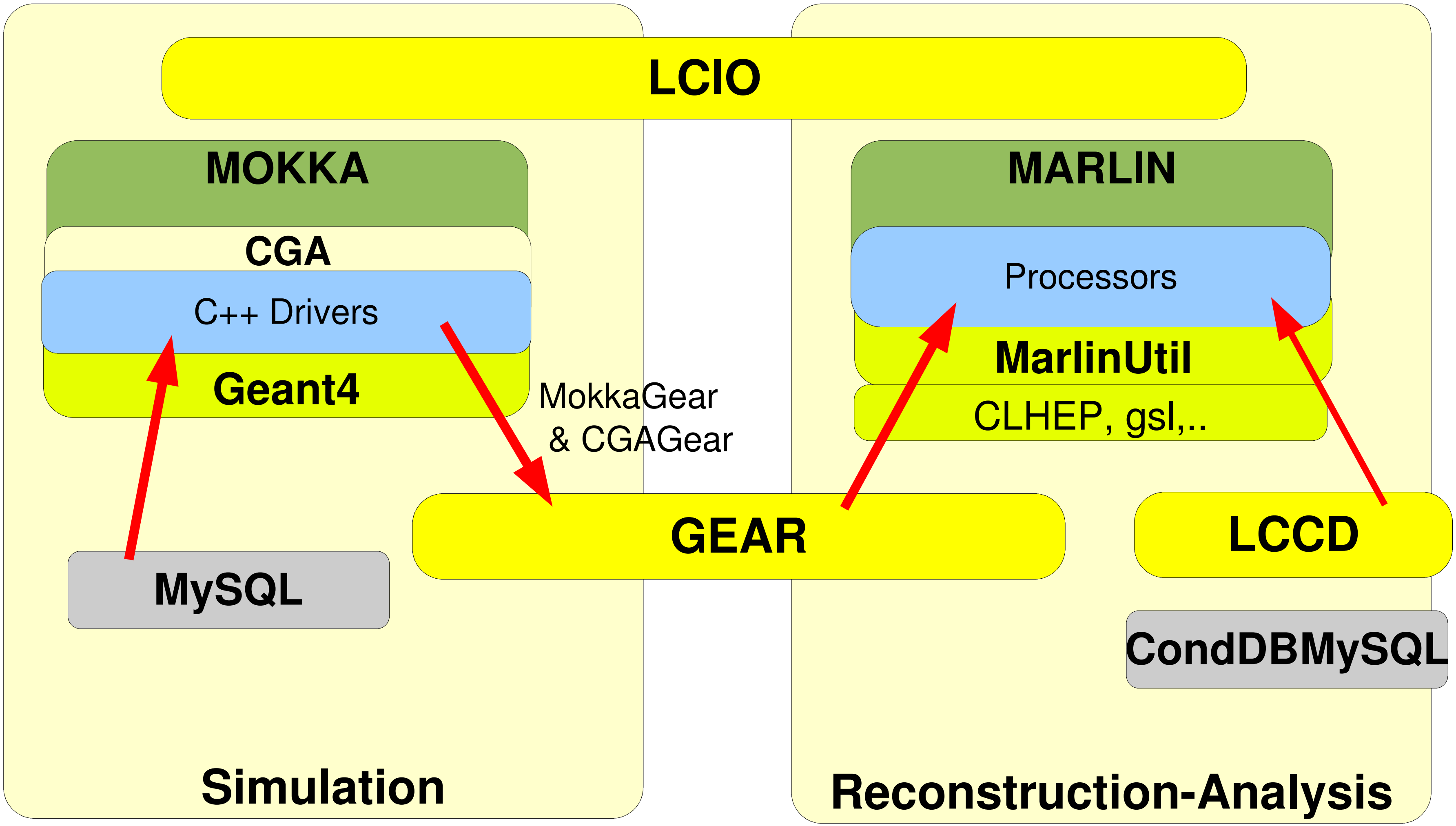
- developed at LLR (ecole polytechnique)
- writes LCIO
- **uses MySQL DB + geometry drivers**
- **flexible geometry setup on subdetector basis:**
  - Tesla TDR / LDC
  - SiD
  - Calice testbeam prototypes
- used in European/ LDC study

LDC simulation framework



Frank Gaede, Common Geometry Meeting, SLAC Sep. 18- 22, 2006

LDC simulation framework



Gear

GEometry API for Reconstruction


```
- <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>
```

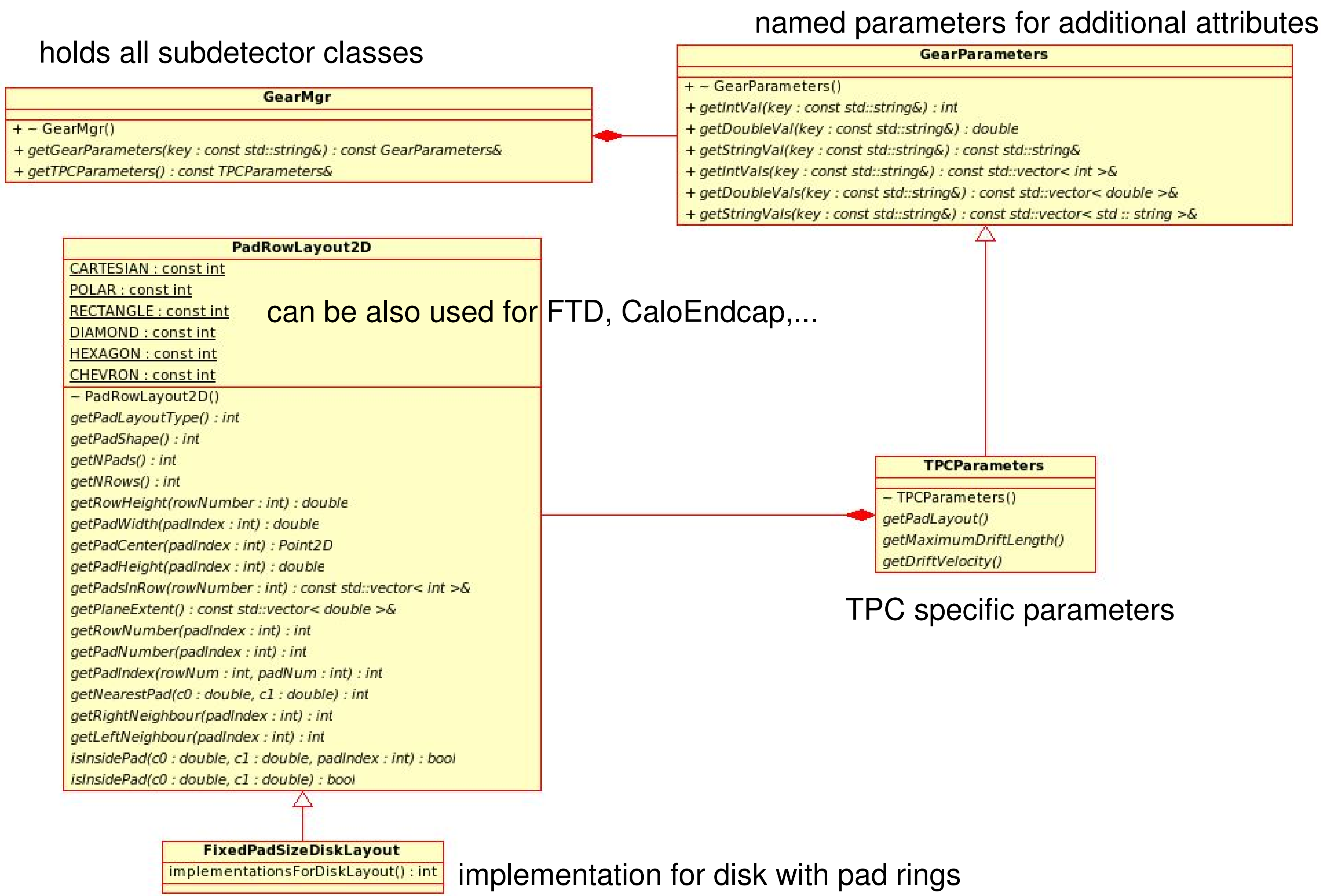

compatible with US – compact format

- 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 – under development
- **abstract interface** (a la LCIO)
  - concrete implementation based on XML files
  - and Mokka-CGA – under development (now first version)

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

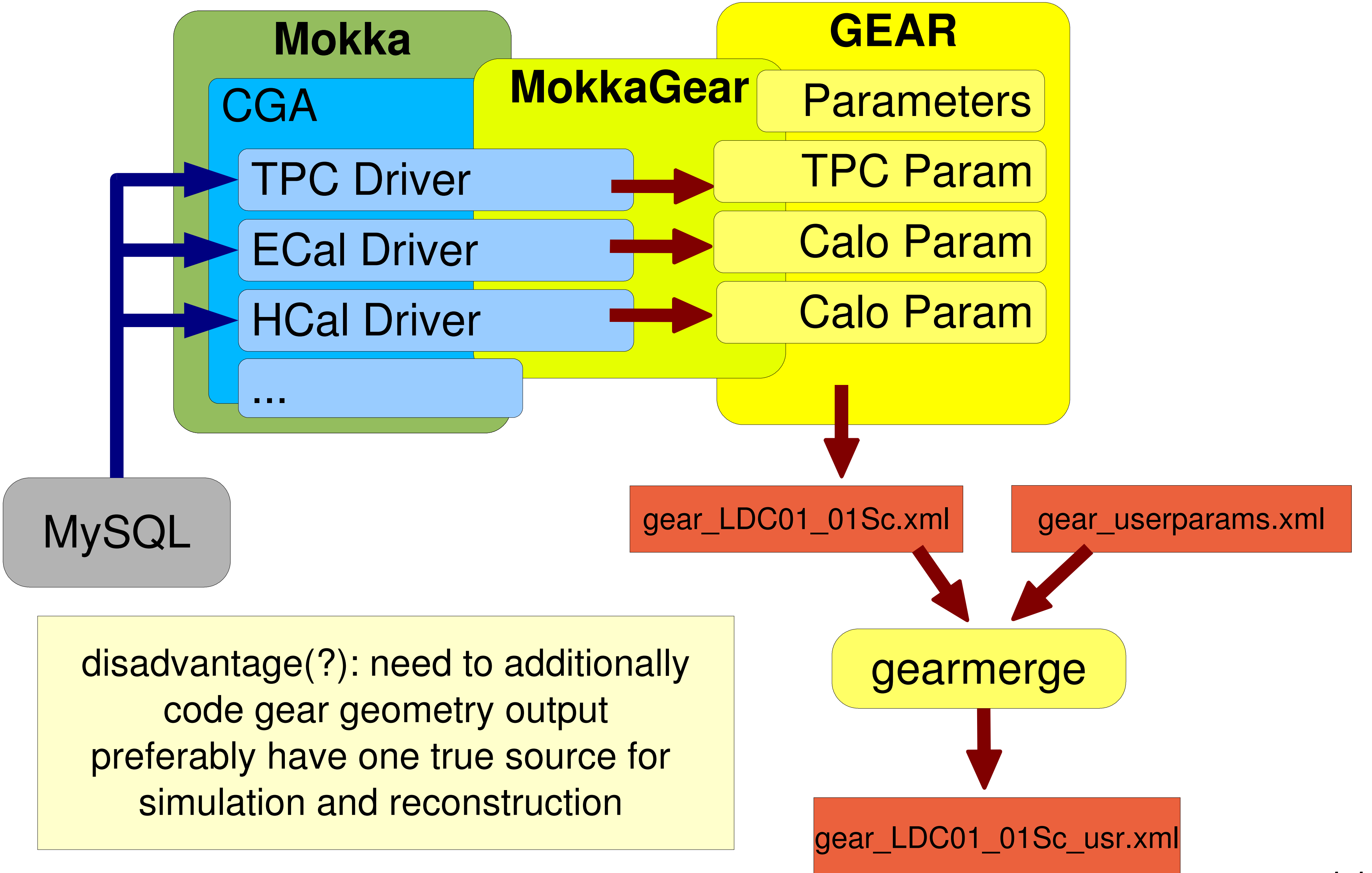


MokkaGear

- extension to Mokka
- 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)
  - VTX (to be released with Mokka 6.2)
- released with Mokka 6.1
  - optional feature
  - only if Gear is installed and included

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

MokkaGear



disadvantage(?): need to additionally code gear geometry output
preferably have one true source for simulation and reconstruction

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
```

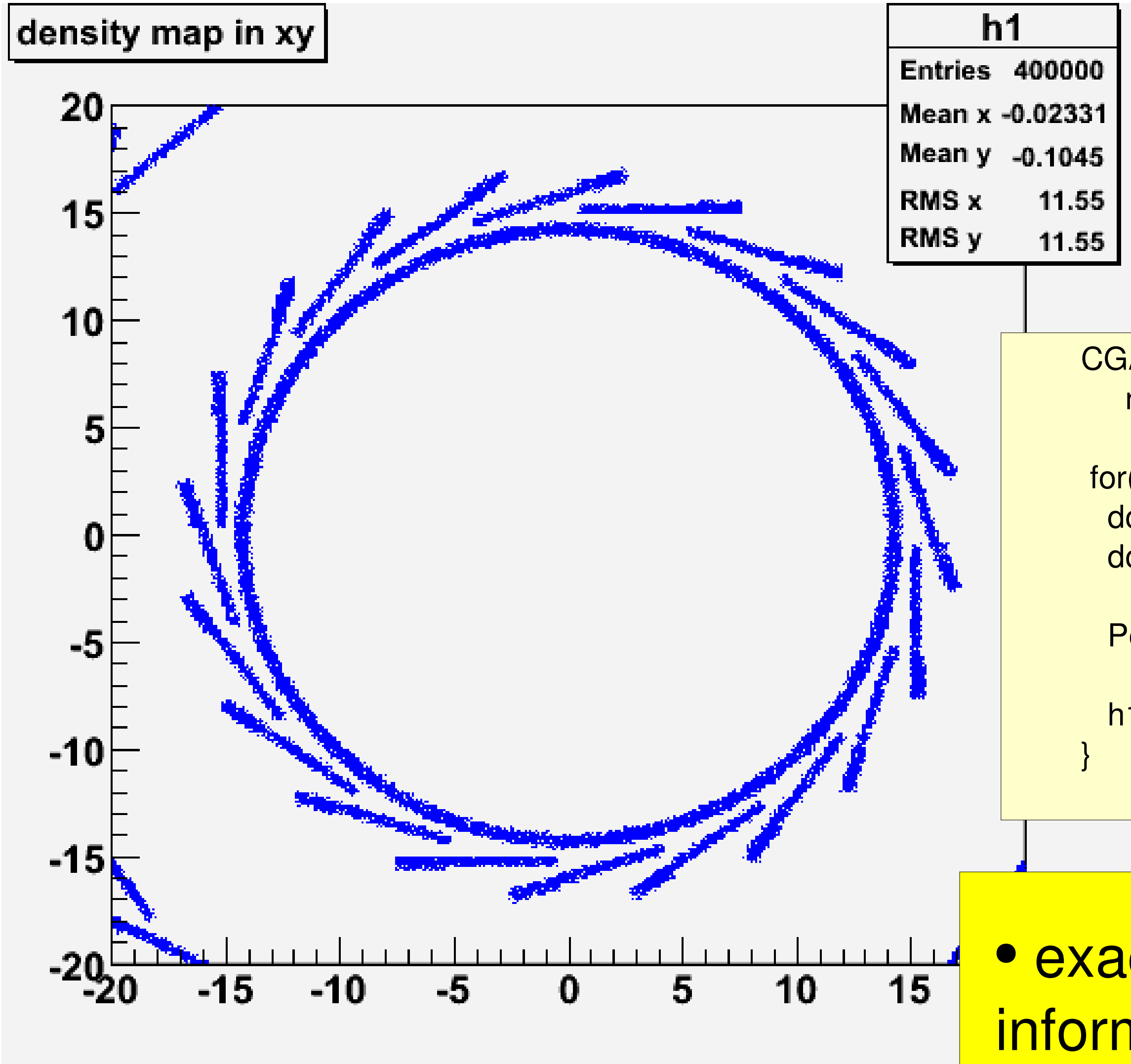
- proposal from Argonne Simulation Meeting 2004
- now implemented with Mokka CGA (geant4)

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
```

question: is this the correct interface?
e.g. **more** needed for tracking to compute dEdx and in more compact form
i.e. the **actual volumes** possibly with **averaged** material (performance)

CGAGear



- implemented by G.Musat, LLR

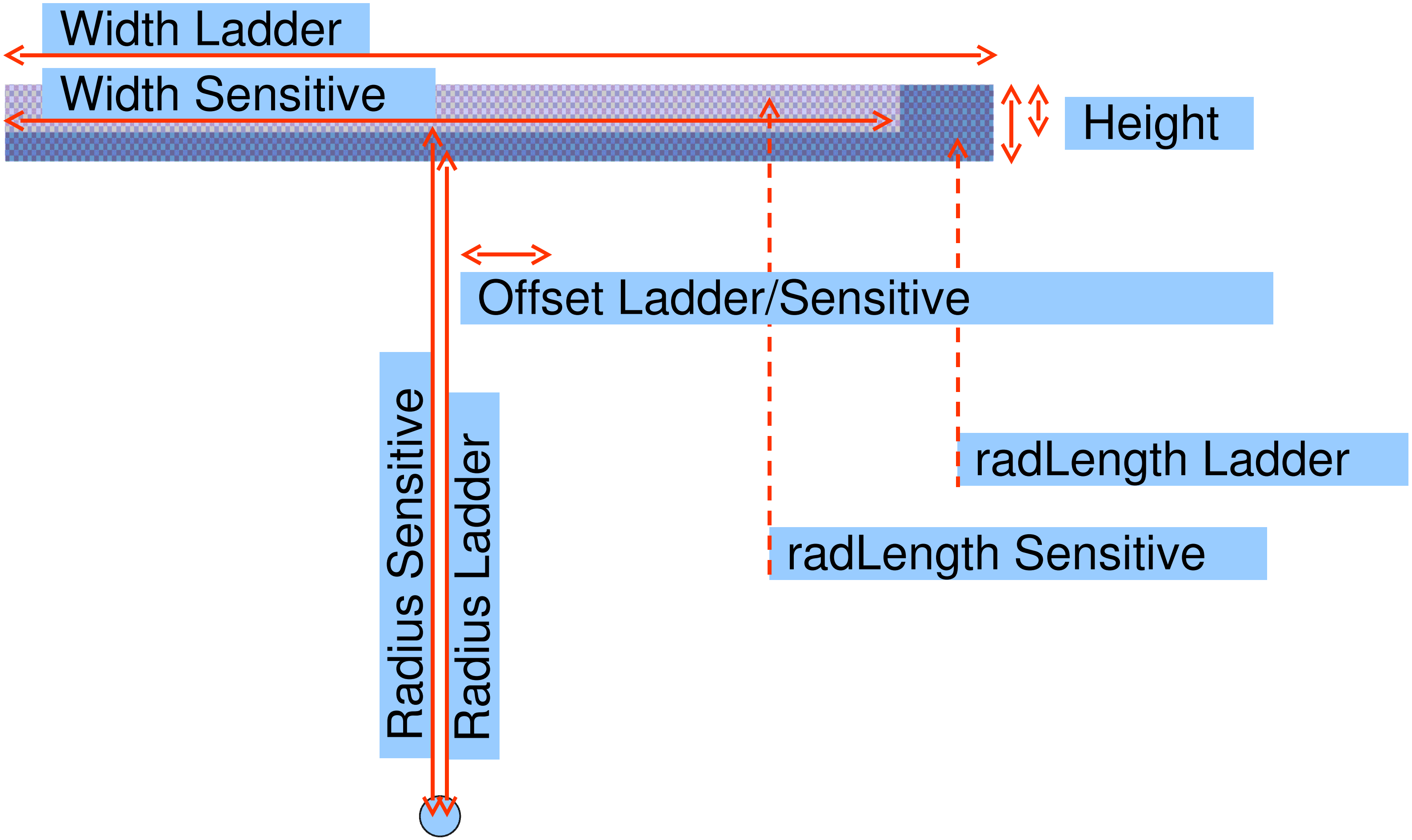
```
CGAGearPointProperties * pointProp =
    new CGAGearPointProperties(steer.str(),...);

 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) ?

VTX XML description

```
<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>
```

VTX ladder



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

Gear status

- version v00-03
  - TPC, Hcal, Ecal and VTX interfaces defined and implemented
  - user parameters
  - write xml files from parameters in memory
  - tool to merge files: gearmerge
  - description of TPC prototypes (rectangular pad plane)
- first version of VXD description
- GearCGA - material properties

not addressed in GEAR

- material properties per volume
  - -> easy to incorporate, e.g. via XML list and tags
- cellid to position conversion for noise simulation
  - -> no straight forward implementation with xml
  - -> could extend interface and use CGA
- nearest and next to nearest neighbor cells for clustering
  - -> same as above
- subdetector interface in GEAR lacks some level of detail, e.g. gaps/corners in calorimeter

wish list for new system

- known deficiencies of GEAR should be addressed/covered
- ideally we should define a common API a la LCIO and have one or several implementations
- multi language support, ie. Java and C++
- final goal should be to have one true source of the geometry for simulation and reconstruction
- smooth transition to new common system should be possible – also true for LCDD/org.lcsim
- system needs to be flexible for extension:
  - start with a first implementation that is (at least) as powerful as current systems
  - gradually increase the level of sophistication
- new system should have no known limitations, i.e in principle every detail that could be needed should be describable