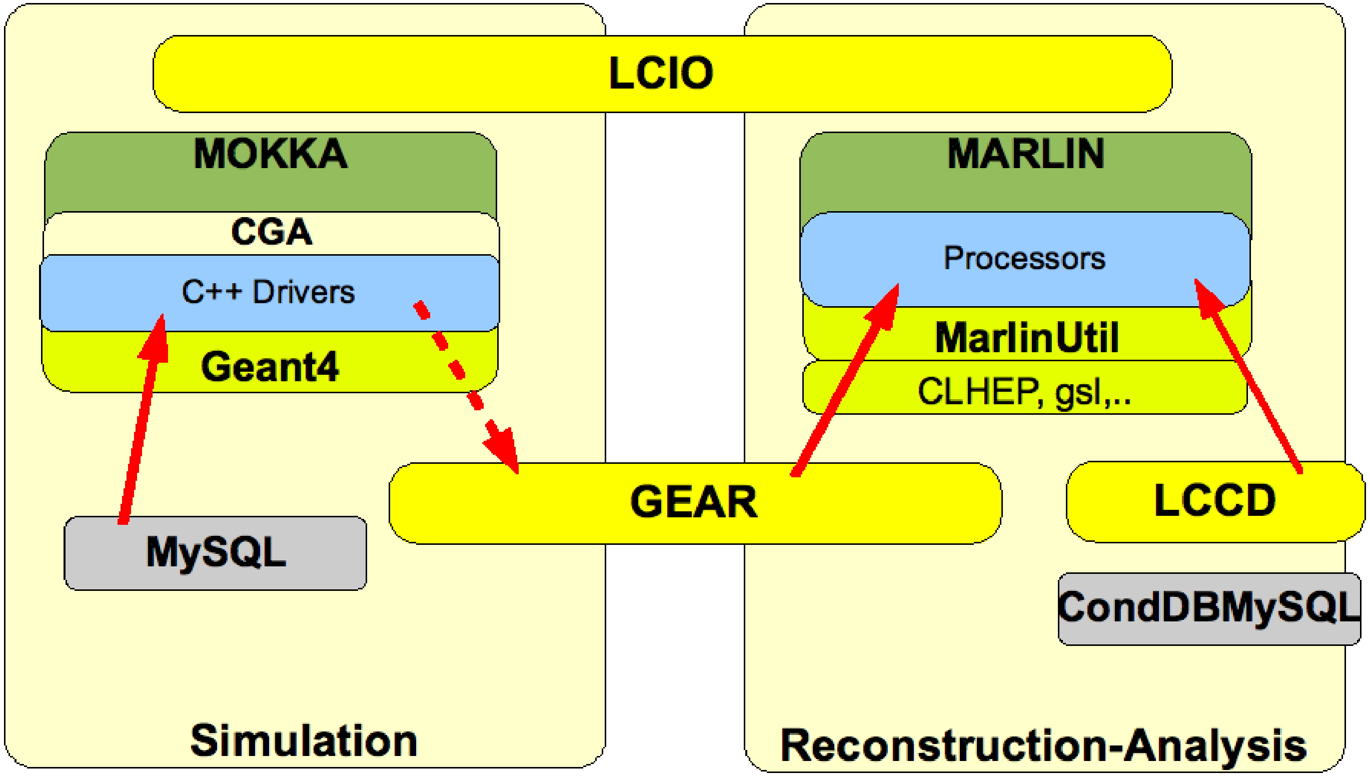
GEAR – geometry API for reconstruction

Frank Gaede
DESY

Geometry Toolkit for LC Workshop
CERN Feb. 24, 2010

Outline

- Introduction
- ILD geometry description
  - simulation / reconstruction
- GEAR
  - design
  - features
- beyond GEAR
- Summary & Outlook

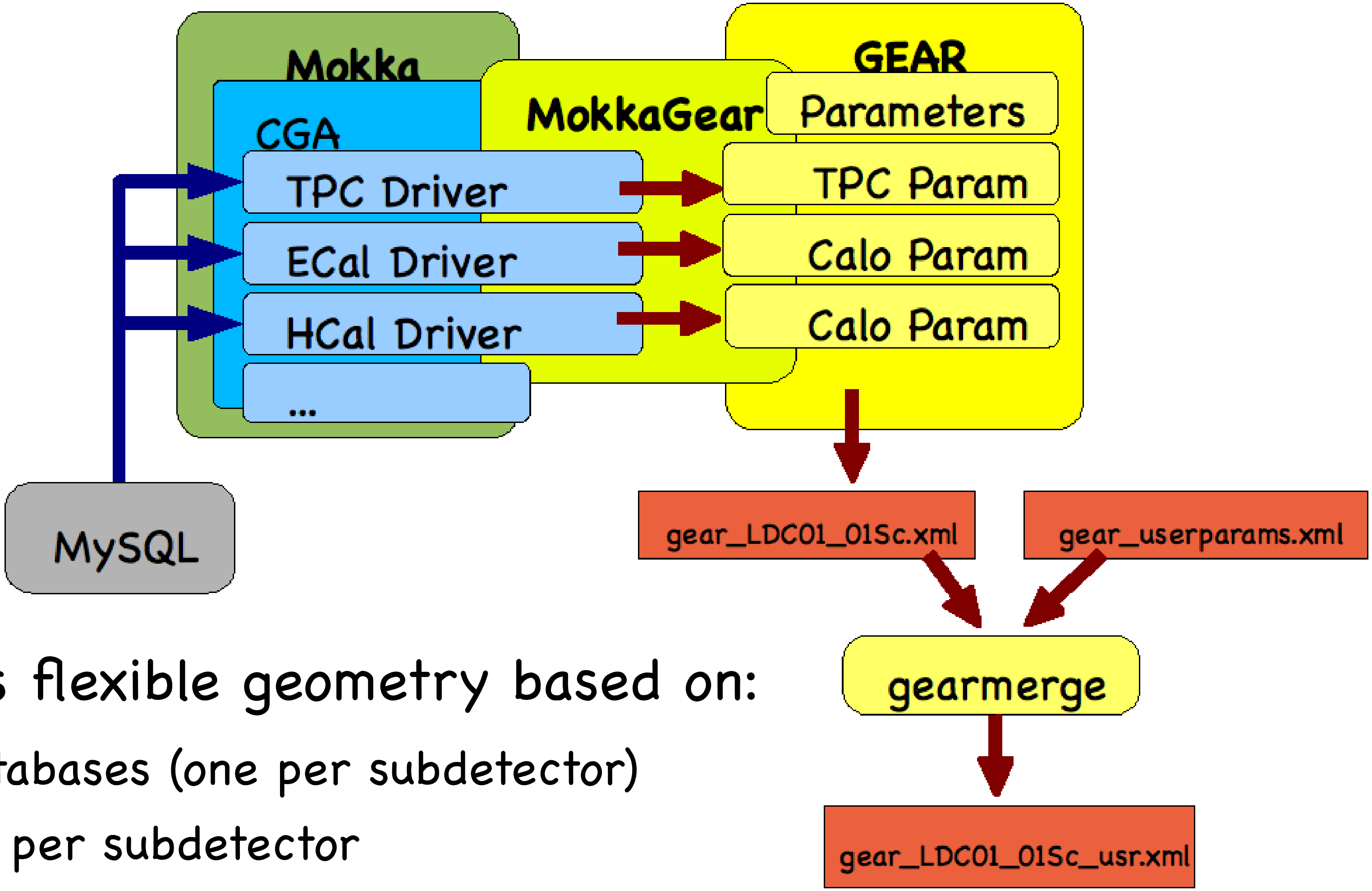


Introduction

- A detector geometry toolkit that provides a single source of information during simulation, reconstruction and analysis is central to every HEP software framework
- there are different views and levels of detail at the various stages of computing, e.g.
- simulating detector response needs 'material distribution in space' + sensitive detector
- reconstruction of tracks, clusters and particles requires more abstract high level view:
- reconstruction code, e.g. pattrec is written for a certain type of detector such as a TPC – need to answer specific questions which are different from simulation
- in ILD currently a two stage approach: Mokka-geometry and **GEAR** (bridged through MokkaGear)

Frank Gaede, DESY, LC geometry meeting, CERN, Feb 24, 2010

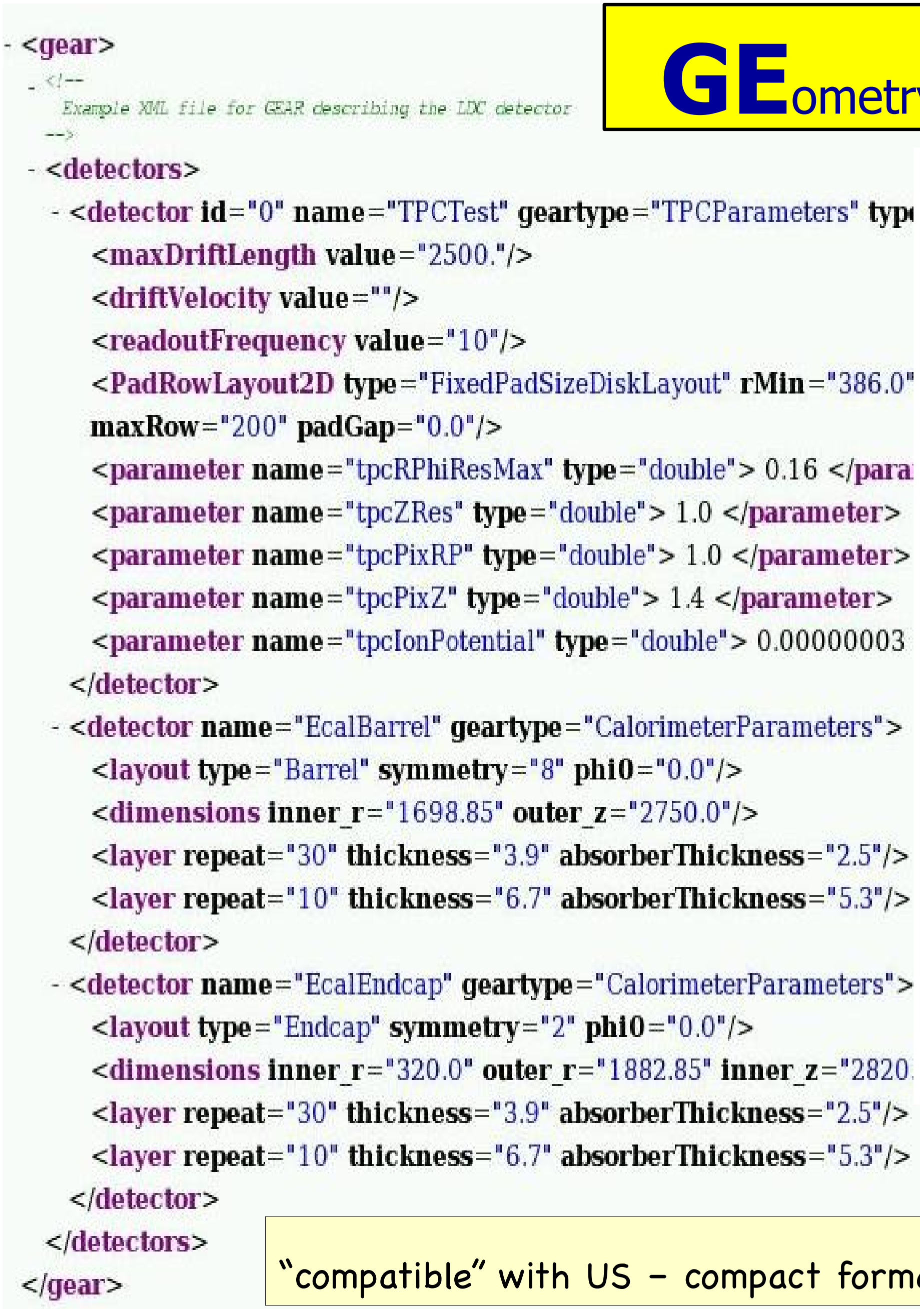
Mokka geometry description



- Mokka has flexible geometry based on:
  - MySQL databases (one per subdetector)
  - C++ driver per subdetector
- need to 'export' the geometry for reconstruction:
  - MokkaGear: subdetector drivers provide the needed parameters at geometry initialization

GEAR - geometry

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

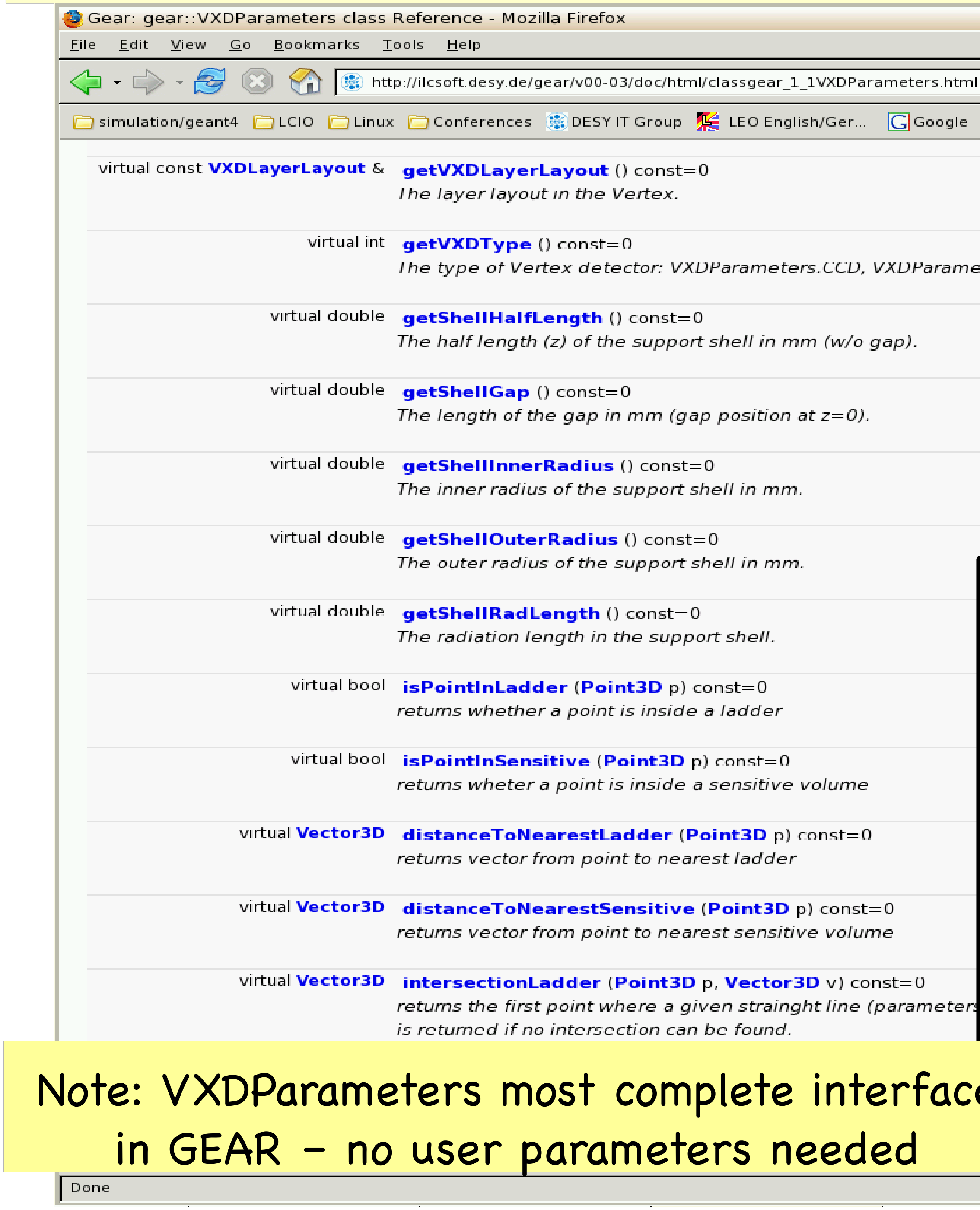
- well defined geometry definition for reconstruction that
  - is flexible w.r.t different LC detector concepts
  - has high level information needed for reconstruction
  - provides access to material properties
- **abstract interface** (a la LCIO)
  - implementation in C++
  - currently: persistency with XML
  - and Mokka-CGA - geant4

Frank Gaede, DESY, LC geometry meeting, CERN, Feb 24, 2010

GEAR - Classes

- Subdetector description
  - high level description of detector shape and geometry
  - one class for every ILC subdetector type, e.g.
  - TPC, VTX, Calorimeter (Ecal, Hcal,...) ...
  - provides geometry parameters and some navigation
  - additional named user attributes
    - Note: currently used quite extensively in ILD reconstruction - this should be addressed in a future release !
    - the interface for a subdetector should be as complete as possible, such that no user parameters are needed
- Material properties
  - point properties (density, material, radlen,...)
  - distance properties integrated along straight path
  - use Mokka-CGA interface to geant4 geometry at runtime

example – GEAR API VXD


Gear: gear::VXDParameters class Reference - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://ilcsoft.desy.de/gear/v00-03/doc/html/classgear_1_1VXDParameters.html
Go
simulation/geant4 LCIO Linux Conferences DESY IT Group LEO English/Ger... Google MyHome Ctim
Done

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 VXD

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

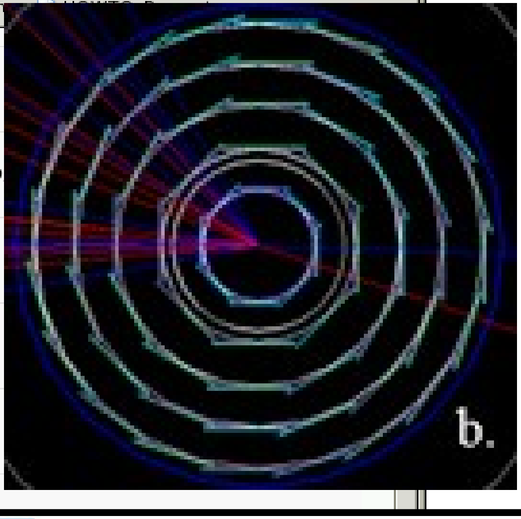
virtual bool **isPointInSensitive** (**Point3D** p) const=0
returns wheter a point is inside a sensitive volume

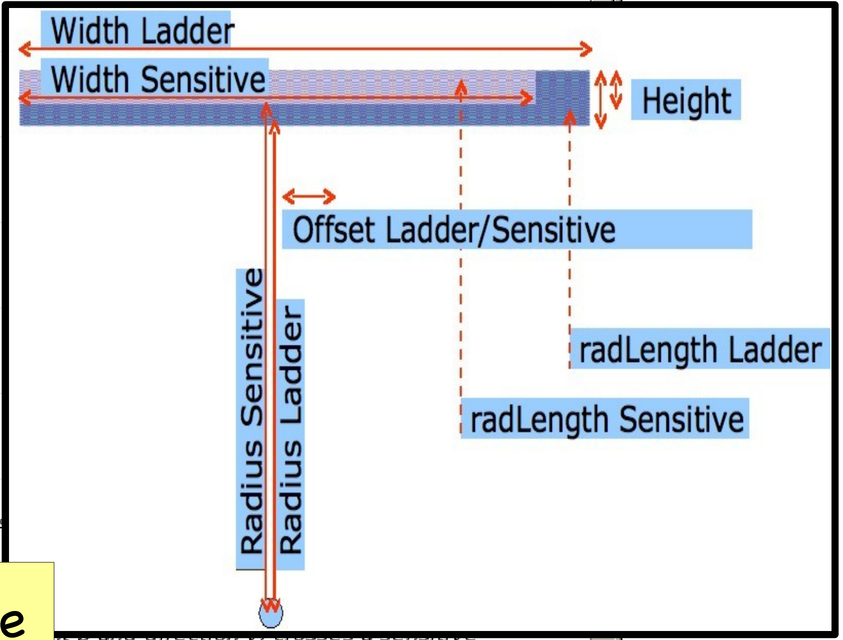
virtual **Vector3D** **distanceToNearestLadder** (**Point3D** p) const=0
returns vector from point to nearest ladder

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 (parameter
is returned if no intersection can be found.

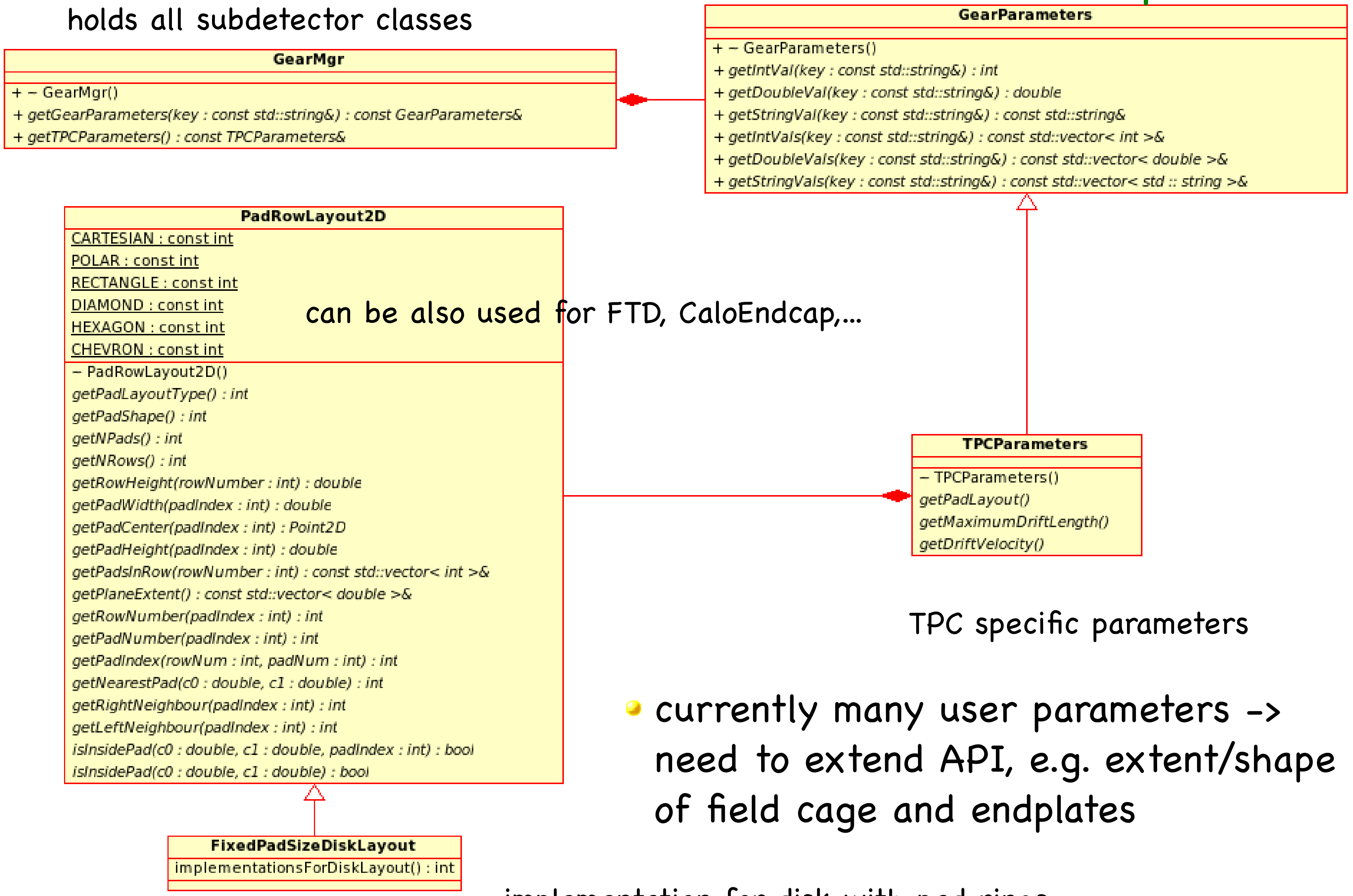
...t p and direction v) crosses a sensitive





Note: VXDParameters most complete interface
in GEAR – no user parameters needed

GEAR example:TPC API



holds all subdetector classes

GearMgr

- + − GearMgr()
- + *getGearParameters(key : const std::string&) : const GearParameters&*
- + *getTPCParameters() : const TPCParameters&*

additional named **user parameters**

GearParameters

- + − GearParameters()
- + *getIntVal(key : const std::string&) : int*
- + *getDoubleVal(key : const std::string&) : double*
- + *getStringVal(key : const std::string&) : const std::string&*
- + *getIntVals(key : const std::string&) : const std::vector< int >&*
- + *getDoubleVals(key : const std::string&) : const std::vector< double >&*
- + *getStringVals(key : const std::string&) : const std::vector< std :: string >&*

PadRowLayout2D

CARTESIAN : const int
POLAR : const int
RECTANGLE : const int
DIAMOND : const int
HEXAGON : const int
CHEVRON : const int

− PadRowLayout2D()
getPadLayoutType() : int
getPadShape() : int
getNPads() : int
getNRows() : int
getRowHeight(rowNumber : int) : double
getPadWidth(padIndex : int) : double
getPadCenter(padIndex : int) : Point2D
getPadHeight(padIndex : int) : double
getPadsInRow(rowNumber : int) : const std::vector< int >&
getPlaneExtent() : const std::vector< double >&
getRowNumber(padIndex : int) : int
getPadNumber(padIndex : int) : int
getPadIndex(rowNum : int, padNum : int) : int
getNearestPad(c0 : double, c1 : double) : int
getRightNeighbour(padIndex : int) : int
getLeftNeighbour(padIndex : int) : int
isInsidePad(c0 : double, c1 : double, padIndex : int) : bool
isInsidePad(c0 : double, c1 : double) : bool

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

TPCParameters

− TPCParameters()
getPadLayout()
getMaximumDriftLength()
getDriftVelocity()

TPC specific parameters

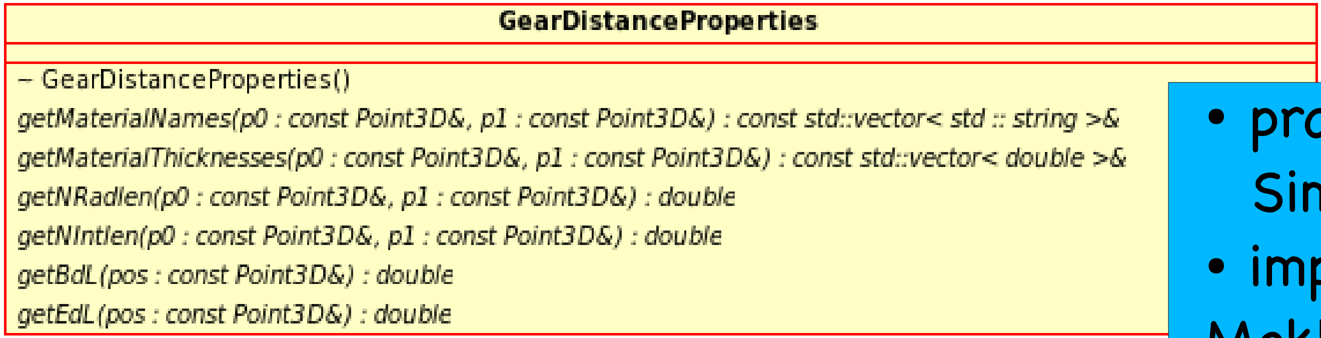
FixedPadSizeDiskLayout

implementationsForDiskLayout() : int

implementation for disk with pad rings

- currently many user parameters -> need to extend API, e.g. extent/shape of field cage and endplates

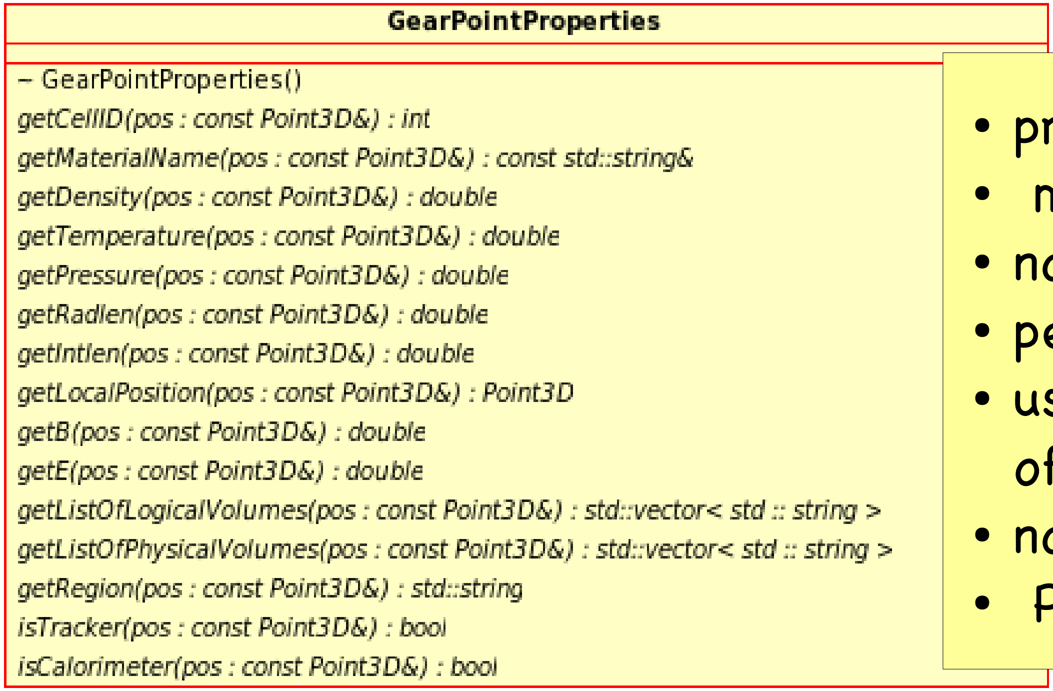
GEAR – material properties



```
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(!)
- 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
```

- provide detailed access to
- materials and field
- no navigation
- performance !?
- used e.g. to get material budget of detector
- not used in current tracking and
- ParticleFlow

- in principle one can get all the needed material properties e.g. for pattrec from this interface together with geometrical properties before actual reconstruction starts (performance)

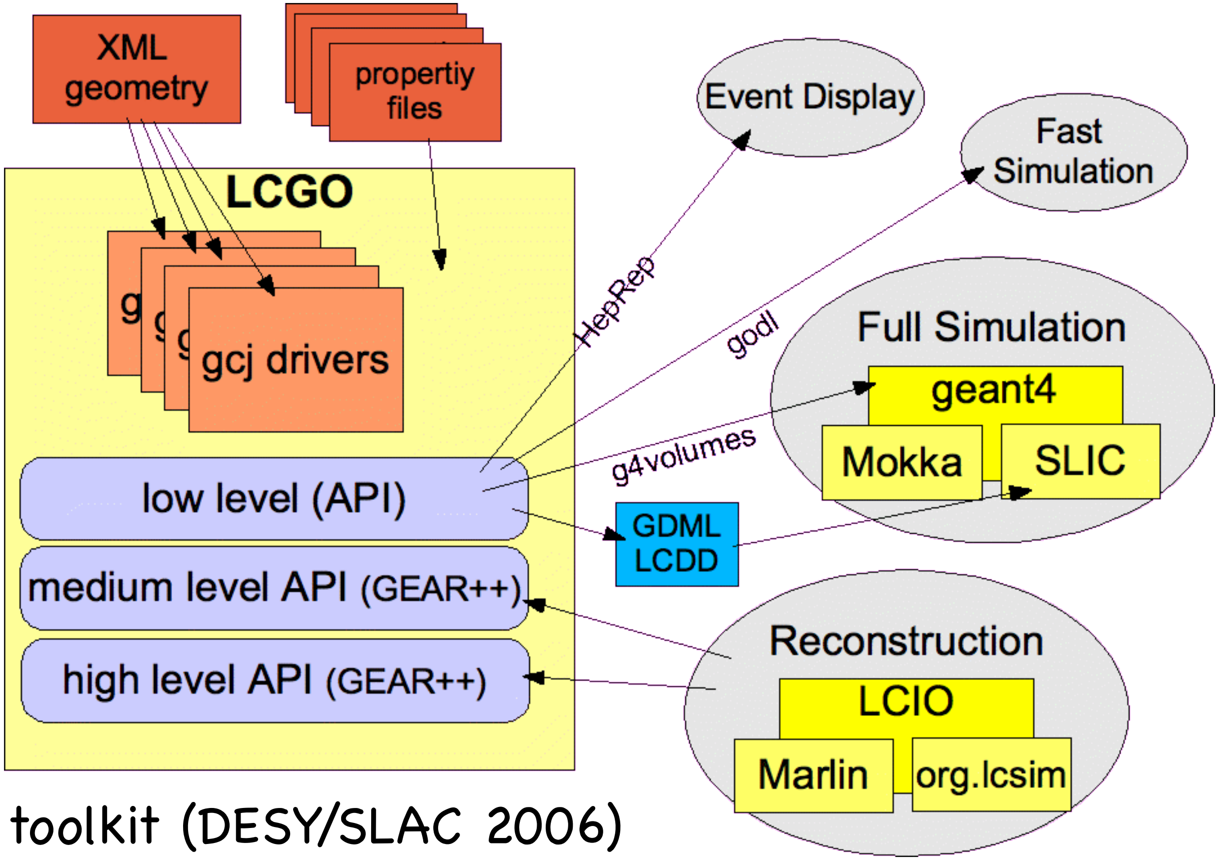
Improving the geometry description

- Mokka MySQL being the leading system not optimal
- ideally have standalone geometry system – for
  - simulation, reconstruction, analysis, event displays
  - provide interfaces with the appropriate level of detail at the various stages
- allow for smooth transition from existing tools (e.g. extend existing GEAR interfaces)
- unified/combined with conditions data base !?
- request from CALICE testbeam to extend GEAR...
- A common geometry toolkit for all (?) LC detector R&D groups would be very useful for interoperability (together with LCIO) => this workshop !

LCGO geometry tool – a conceptual idea

- driver based approach a la Mokka
- MySQL DB replaced by xml files

Key concept: common code base for all clients !



- LCGO – a <u>planned</u> geometry toolkit (DESY/SLAC 2006)
  - based on geometry drivers – written in **JAVA** !
  - use **gcj-compiler** to compile to binary & interface with C++
  - issues with performance – 4 times slower than C++ (2007)
- Note: Java is probably to the optimal choice for current LC studies
- -> could look into implementing a similar concept in C++
- investigate existing packages TGeo, VGeometry,...

Summary & Outlook

- ILD concept uses Mokka/geant4 for the geometry in simulation and **GEAR** at reconstruction
- GEAR provides the needed core functionality for reconstruction – with room for improved, e.g.
  - cellID <-> position conversion
  - colcal to global coordinate conversion
  - next neighbours
  - tracking/navigation
  - ...
- effectively all existing reconstruction code in the Marlin framework uses GEAR – so any new system should implement the GEAR interface – or better an improved version (GEAR++)