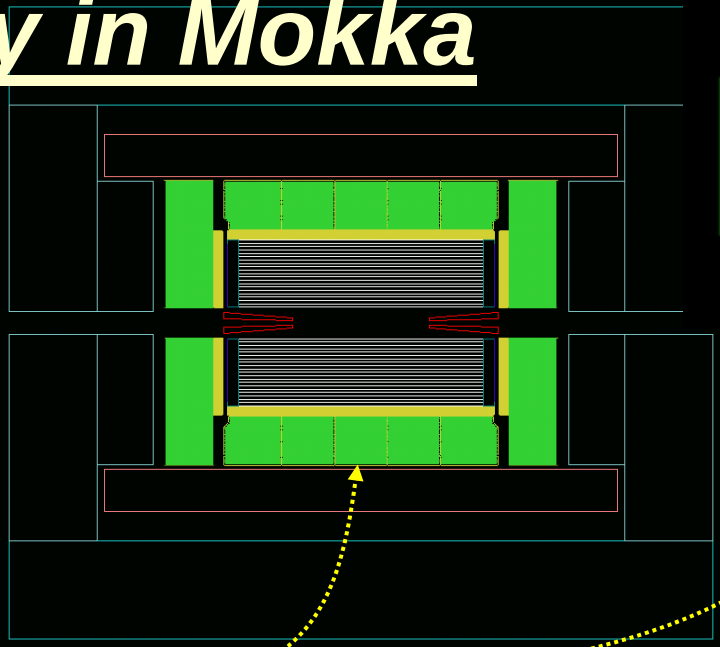
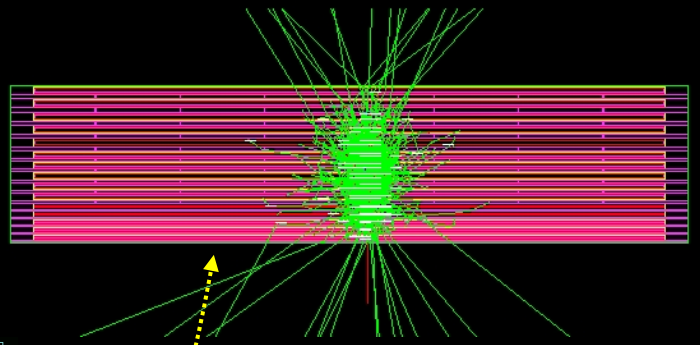
Gear improvements with Mokka-CGA : GearCGA

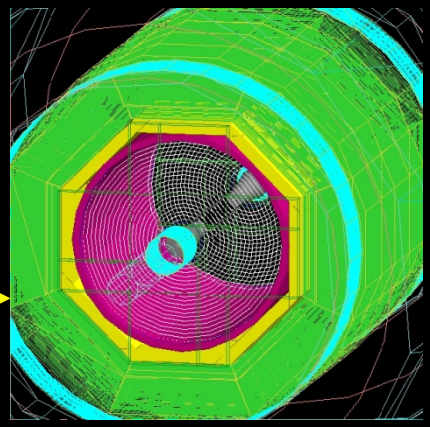
Paulo Mora de Freitas
Gabriel Musat
LLR – Ecole polytechnique

ILD Software and Integration Workshop July 2010, DESY

Geometry in Mokka



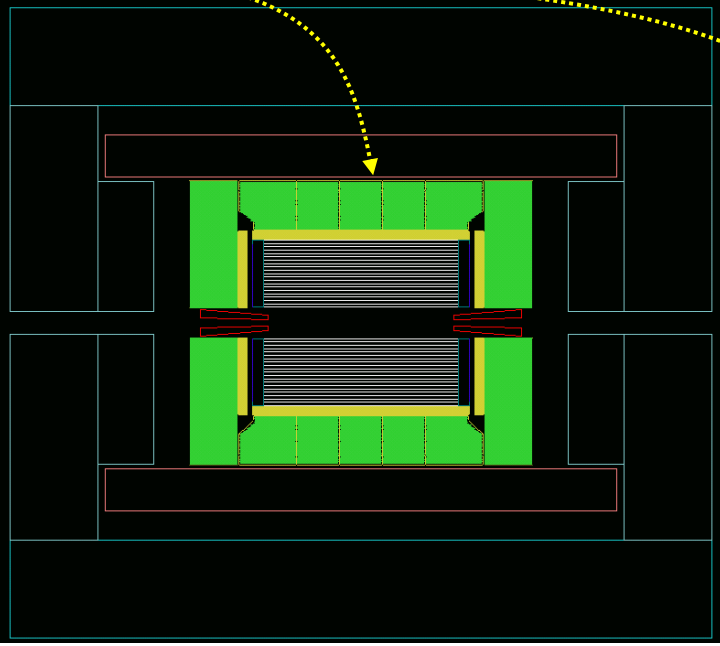


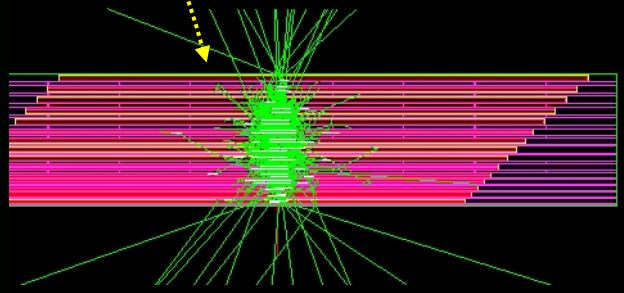


Mokka

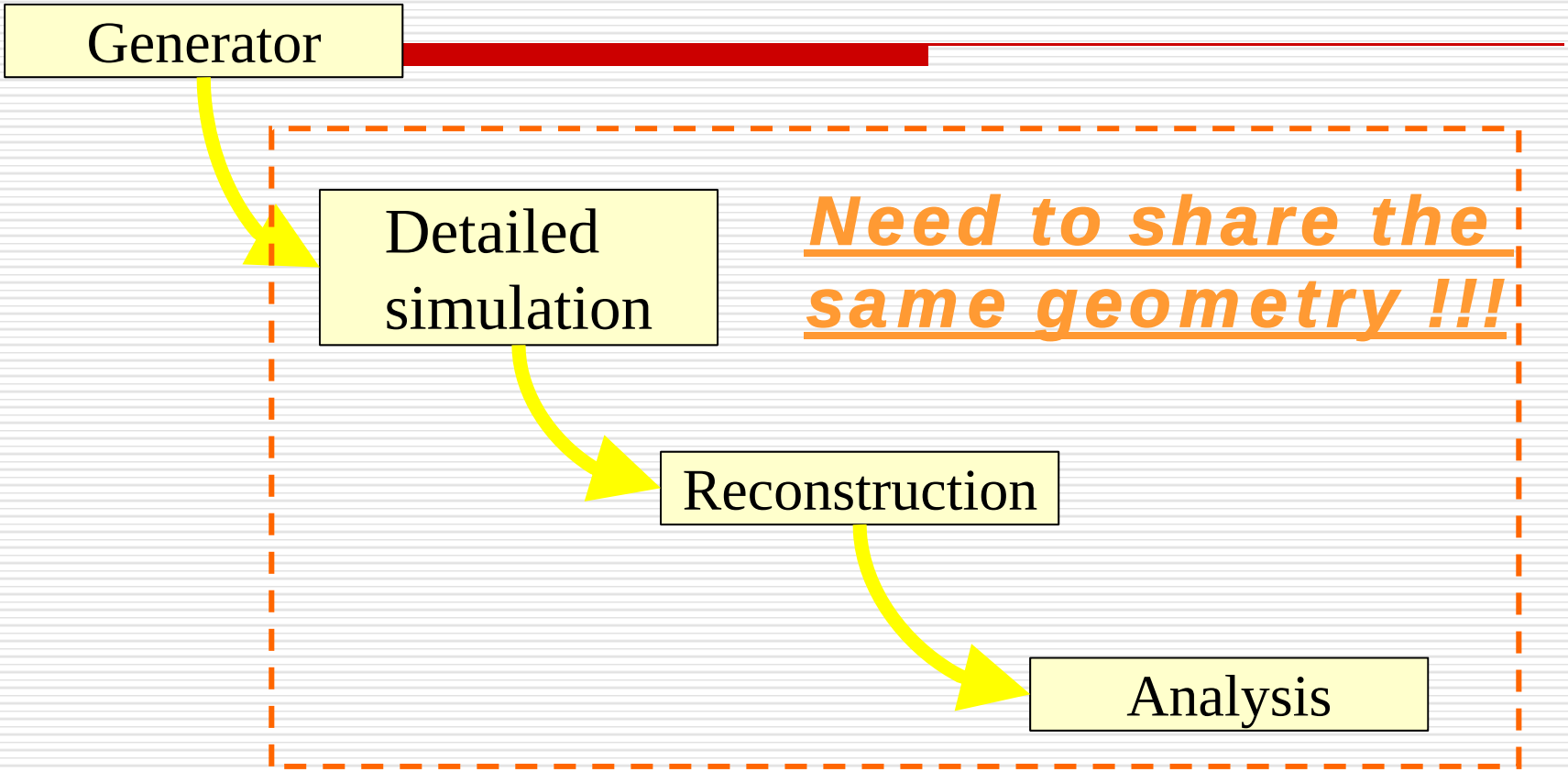
Geometry drivers

Geometry Database



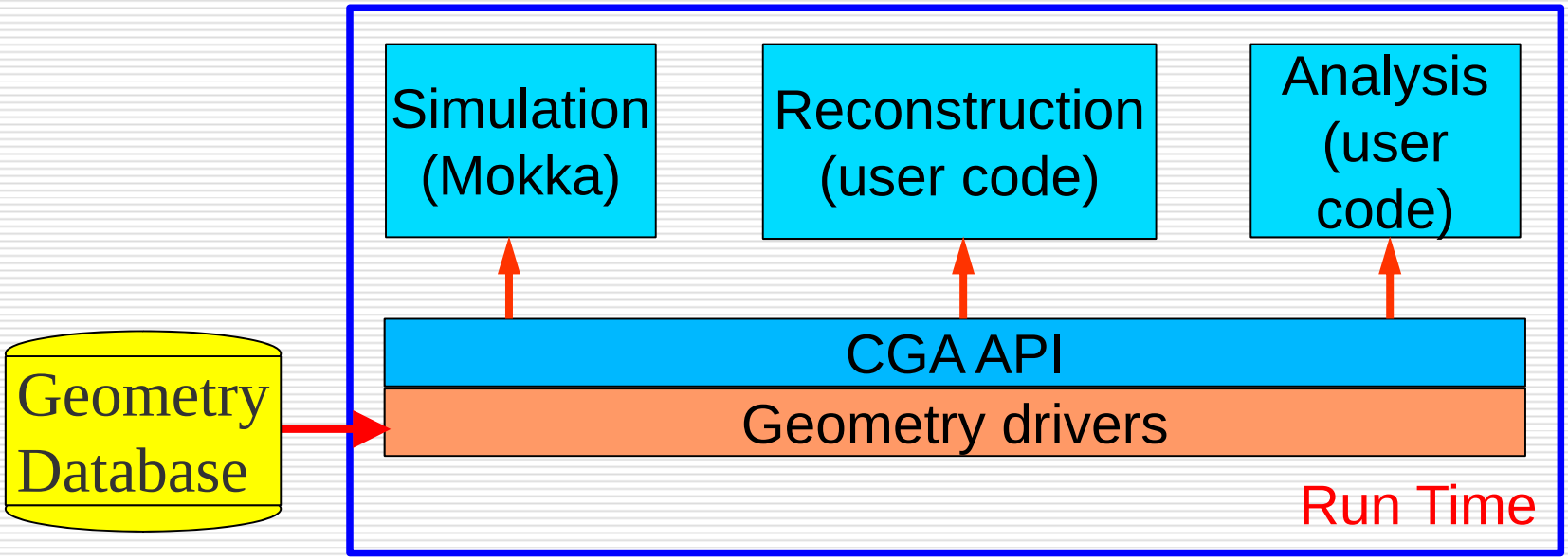


Geometry have to be shared



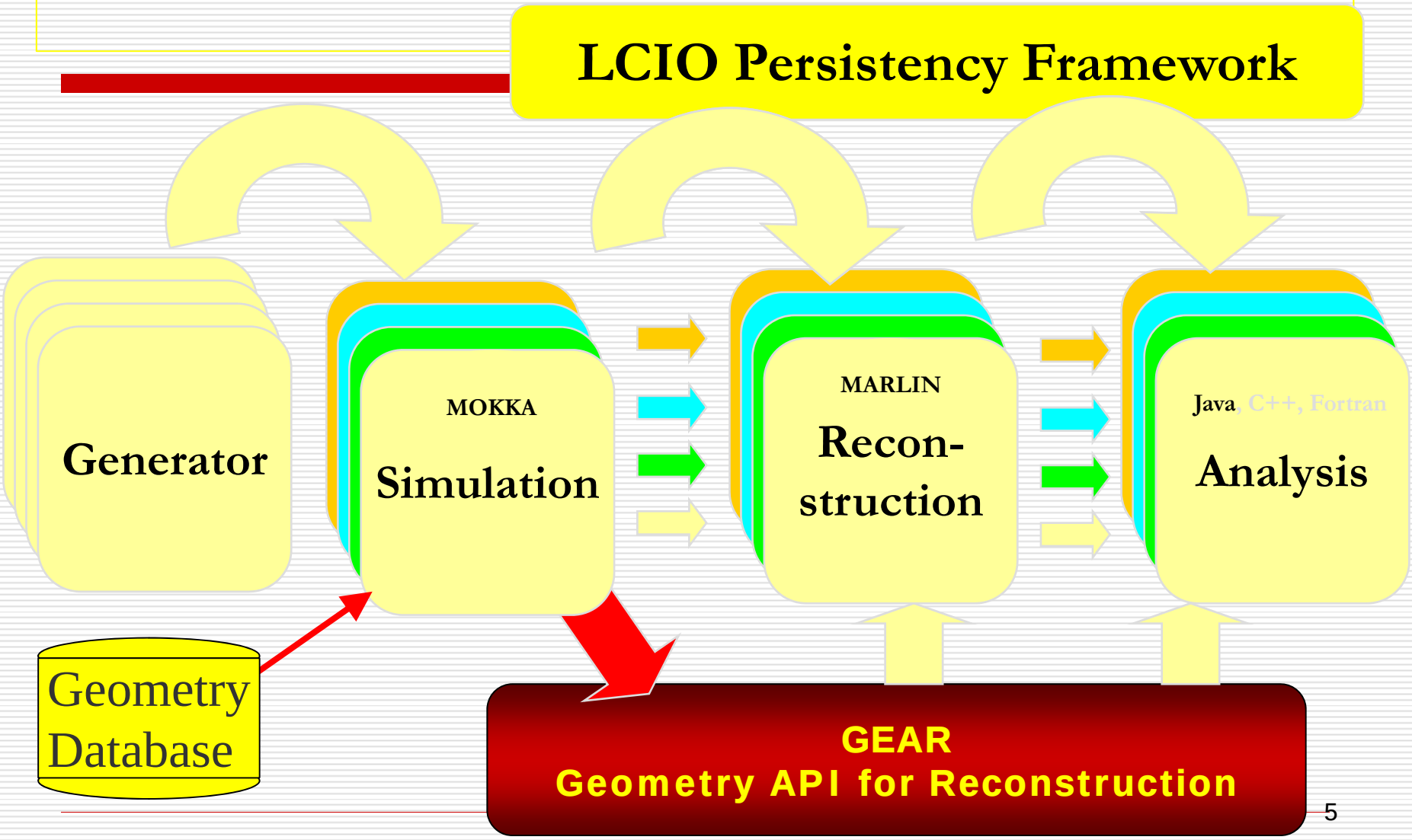
And nobody never tried to retrieve geometrical information via SQL queries, except in the geometry drivers in MOKKA !!!

The Common Geometry Access API (GGA), since 2004



- Available for F77, C++,C and Java
- Links against Geant4 libraries
- Exports only low level geometry (G4 volumes level)
- A few users only

So Gear


LCIO Persistency Framework
Generator
MOKKA
Simulation
MARLIN
Recon-
struction
Java, C++, Fortran
Analysis
Geometry
Database
GEAR
Geometry API for Reconstruction

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"
        <parameter name="tpcPixRP" type="double
        <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"
        <dimensions inner_r="1698.85" outer
        <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

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)

GearCGA

- **First implementation in May 2006**
  - Point- and Distance- Properties
  - Requested by Frank Gaede
  - Based on the **Mokka CGA** interface
  - Three C++ Classes:
    - CGAGeometryInitializer
    - CGAGearPointProperties
    - CGAGearDistanceProperties
- First version – part of GEAR distribution
- Later copied by Frank to Mokka distribution:
  Mokka/source/Geometry/MokkaGear

A new GearCGA implementation

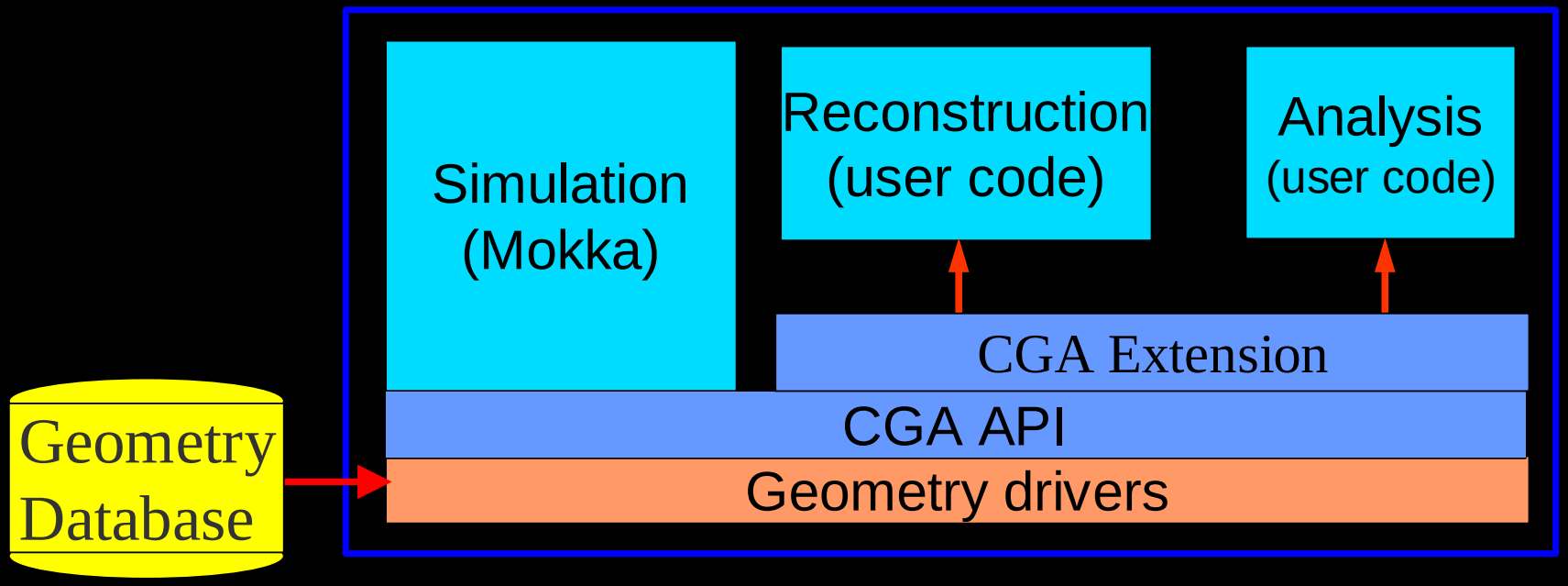
- Proposed at the ”Workshop on Geometry Toolkit for the Linear Collider” at Cern, Feb 2010
  - A full Gear implementation (standard + Point and Distance Properties) “on the box”, available right now
  - Enable users to develop today using the complete Gear API, without waiting for future implementations

    Insures forward compatibility for code

Some GearCGA features

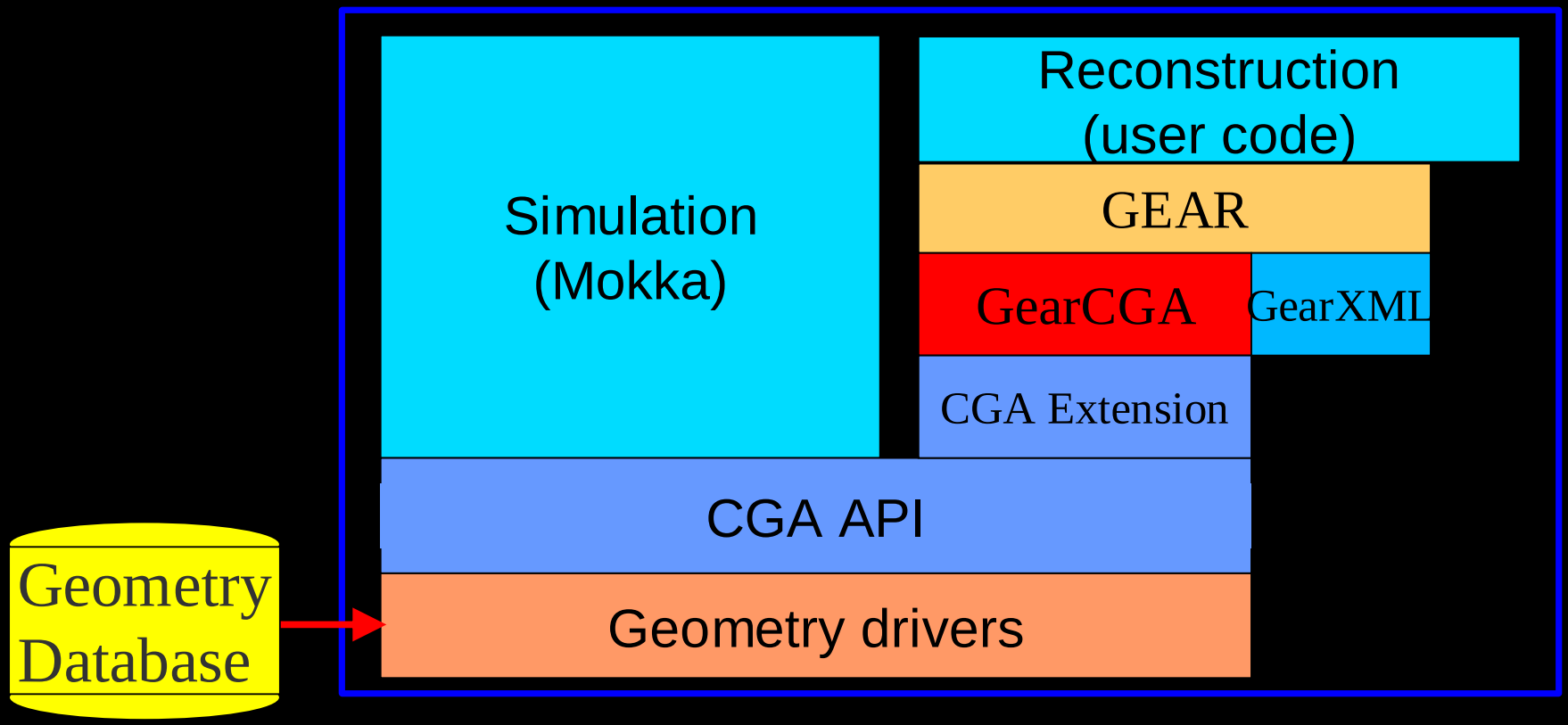
- Don’t need the GEAR xml to initialize Gear: the model name is enough
- The standard Gear API and the Point + Distance Properties extension are managed by a single Gear Mgr object
- The Gear Mgr object can be initialized also by a given LCIO file
  - Insures that the reconstruction task uses exactly the same geometry built in simulation.

Mokka - Common Geometry Access API (F77, C++,C, Java)



- Implements some reconstruction utilities.

GearCGA


Simulation
(Mokka)
Reconstruction
(user code)
GEAR
GearCGA
GearXML
CGA Extension
CGA API
Geometry drivers
Geometry
Database

CGAGeometryInitializer

- Initializes the geometry
  - Able to initialize from an LCIO file, using the steering file stored by Mokka in the LCIO RunHeader
  - **CGAGeometryInitializer * geoInit =**

    CGAGeometryInitializer::

    GetCGAGeometryInitializer(lcioFileName);
- Returns a GearMgr object
  - **GearMgr* gearMgr = geoInit->**

    getCGAGearMgr();

The GearMgr object

- Has all the information until now contained in the GEAR file

- Also provides the information on the Point and Distance Properties
  - const GearDistanceProperties& distProp=
    gearMgr->getDistanceProperties();

Remarks on GearCGA changes

- The usage of these classes is shown in example **Mokka/examples/CGA/Ex07.cc**
- User application has to
  - **#include "CGAGeometryInitializer.h"**
- since we aimed to replace the GEAR file, we had to give access to the ' dEdx ' property :
  - GearCGA had to use the same Physics List as Mokka (specified in the steering file)
  - The user must set the environment variables pointing to the X-section data

Packing necessary libraries

- makes easier the usage of the GearCGA
- the libraries belonging to Mokka, Geant4, CLHEP, MySQL, GEAR, LCIO are put together in one library
  - Library name: libCGAPack.so
  - Location: $G4WORKDIR/lib/$G4SYSTEM
- done after linking Mokka, if the environment variable ' MOKKA_PACK_LIBS ' is set
- For details, see examples and GNUmakefile in: Mokka/examples/CGA

GearCGA TODO list

- Unify GearCGA usage with that of GearTGeo via a “factory” class
- To be able to overwrite parameters found in the steering file stored in the LCIO run header.
  - The MySQL host for example
- To provide a stand-alone SQLite implementation, avoiding the DB server connection at run time

Acknowledgements

- Many thanks to Gabriel Musat, which has contributed a lot for GearCGA