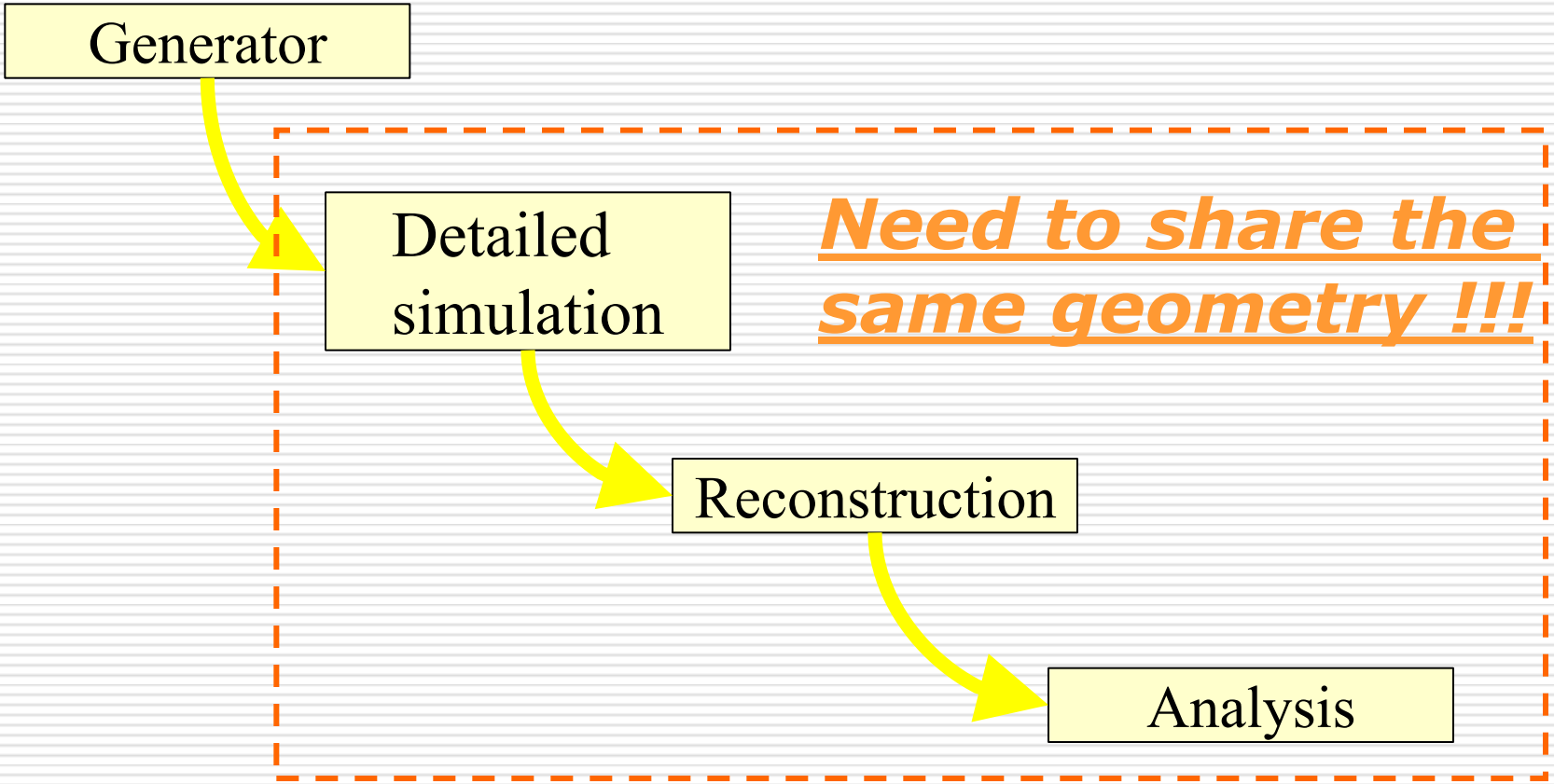
New Gear implementation in Mokka

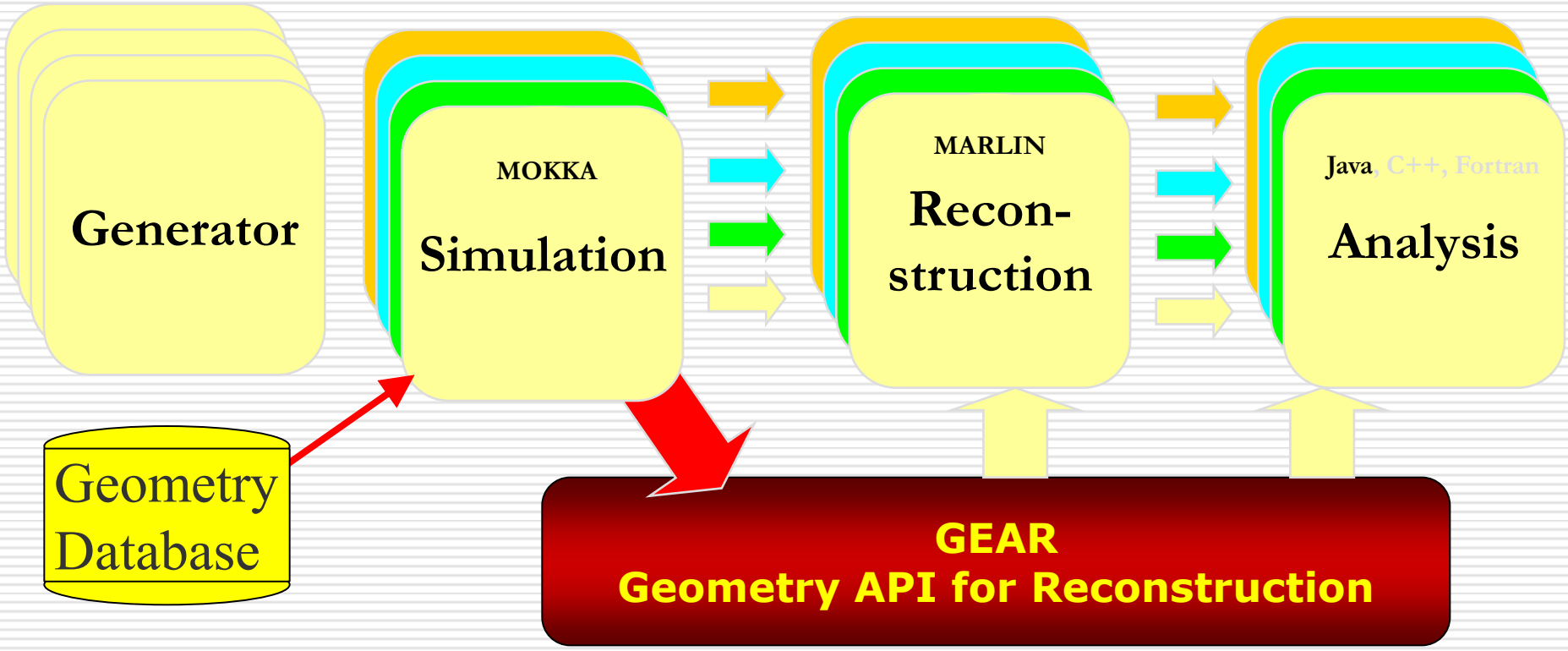
Paulo Mora de Freitas
LLR – Ecole polytechnique

Linear Collider Software Meeting July 2010, DESY

Geometry have to be shared



So Gear


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

GEAR - geometry

```
- <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

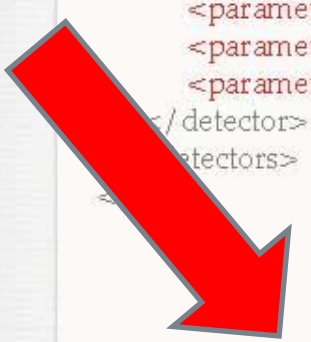
GEometry **A**PI for **R**econstruction

- 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

Reconstruction code should not depend on the simulator name space !

But, inside GEAR XML FILE

```
<gear>
  <global detectorName="D09" />
  <!--Gear XML file automatically created with GearXML::createXMLFile ....-->
  <BField type="ConstantBField" x="0.0" y="0.0" z="4.0"/>
  <detectors>
    <detector name="TPC" geartype="TPCParameters">
      <driftVelocity value="0.000000000e+00" />
      <maxDriftLength value="1.967500000e+03" />
      <readoutFrequency value="0.000000000e+00" />
      <PadRowLayout2D type="FixedPadSizeDiskLayout" rMin="3.710000000e+02" .....
      <parameter name="TPCGasProperties_RadLen" type="double" value="1.155205825e+05" />
      <parameter name="TPCGasProperties_dEdx" type="double" value="2.669216431e-07" />
      <parameter name="TPCWallProperties_RadLen" type="double" value="8.896320560e+01" />
      <parameter name="TPCWallProperties_dEdx" type="double" value="4.328948956e-04" />
      <parameter name="tpcInnerRadius" type="double" value="3.050000000e+02" />
      <parameter name="tpcInnerWallThickness" type="double" value="1.160000000e+00" />
      <parameter name="tpcIonPotential" type="double" value="3.200000000e-08" />
      <parameter name="tpcOuterRadius" type="double" value="1.580000000e+03" />
      <parameter name="tpcOuterWallThickness" type="double" value="1.510000000e+00" />
    </detector>
  </detectors>
</gear>
```



Userparameter written from MOKKA:
NO DEFINED INTERFACE !!!

Indeed it's available via the GEAR API

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)

New GearCGA

- Proposed at the ”Workshop on Geometry Toolkit for the Linear Collider” at Cern, Feb 2010
- A complete Gear API (standard + Point and Distance Properties) on the box, right now
- Forward compatible code with future Gear implementations

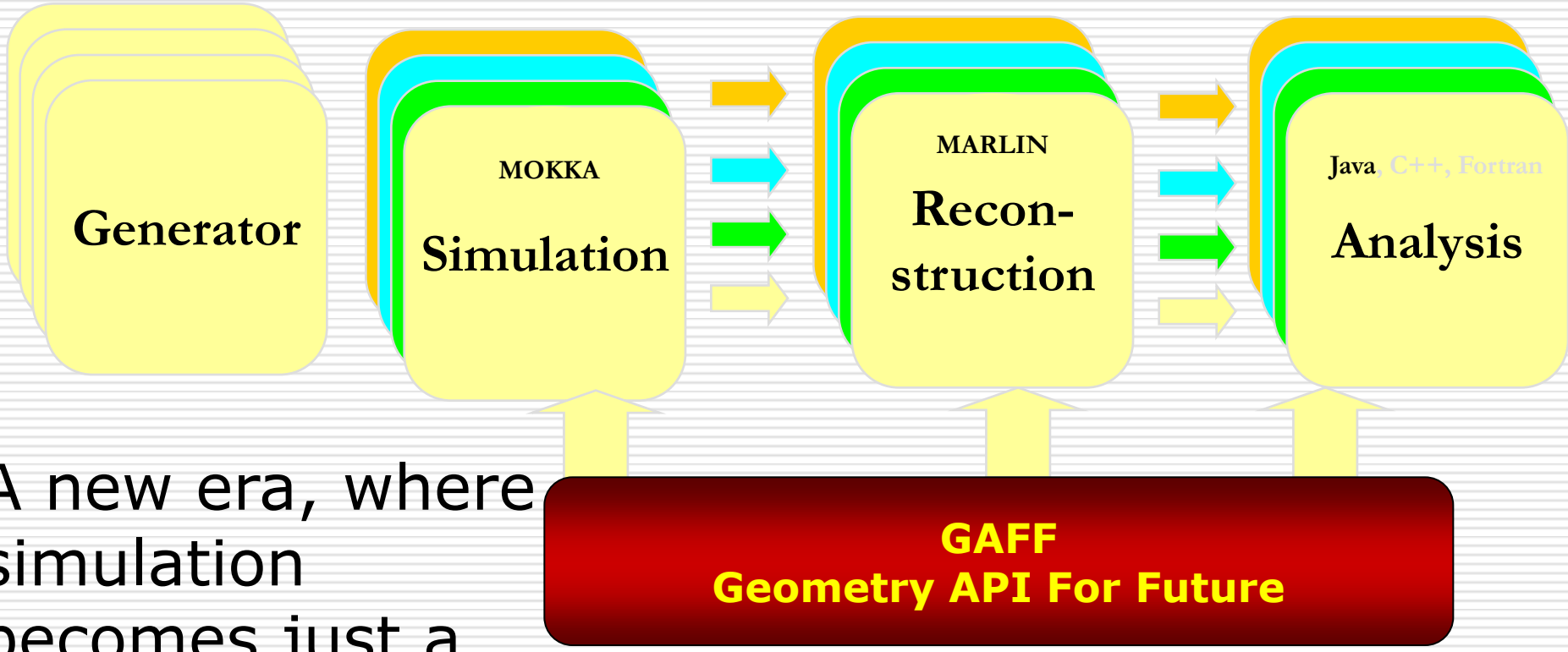
New GearCGA features

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

But GearCGA …

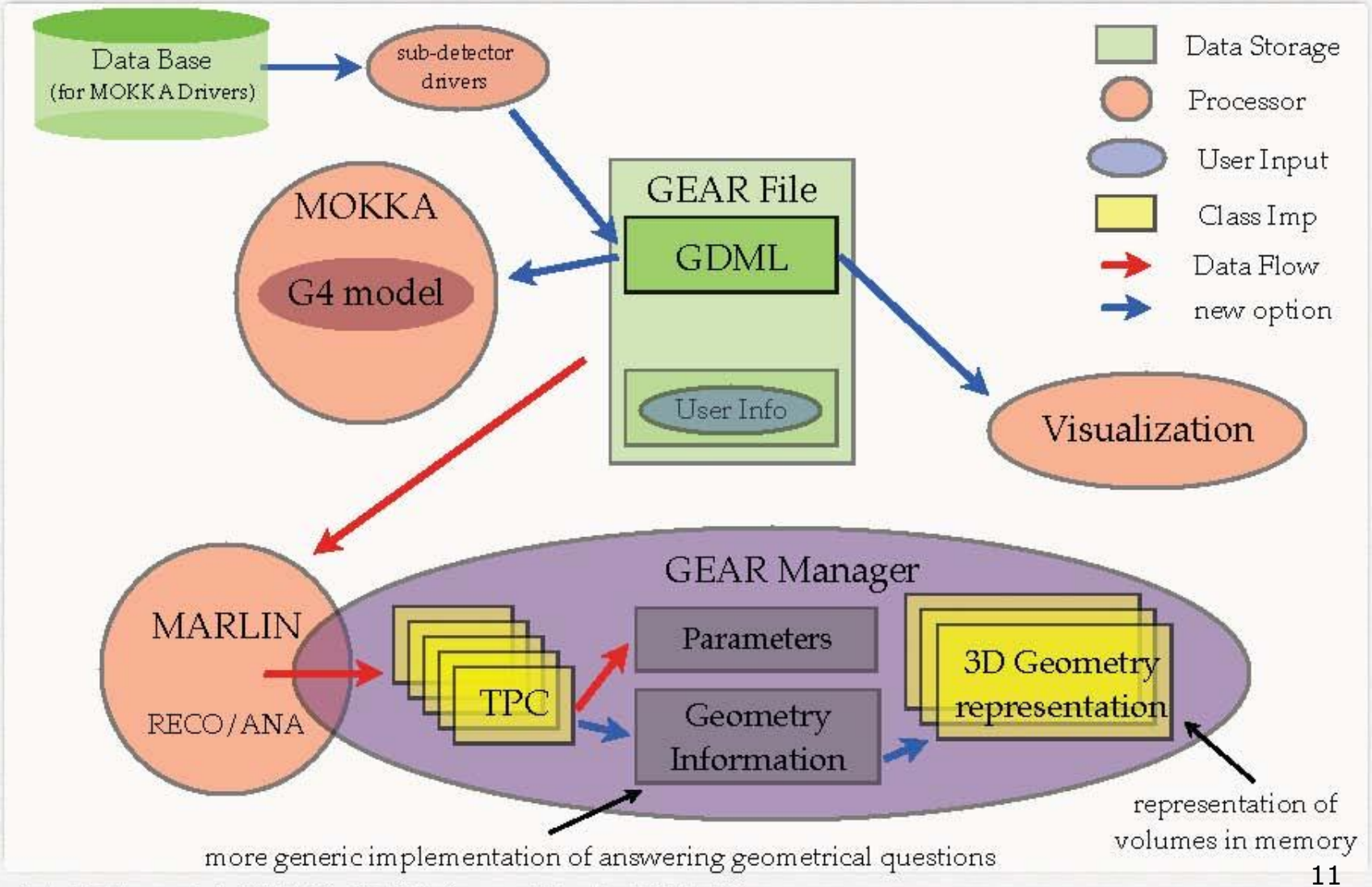
- Co-works only for files created by Mokka
- Has to link against Geant4 libraries
  - All needed G4 libraries packed together with the GearCGA distribution, indeed
- Have to access the Mokka DB
  - Can be solved via a SQLite implementation
- Anyway
  - Definitively people don’t like it
  - It seems that to link against ROOT is better
  - So a good option: to use GearTGeo

Towards a better geometry system in the future: GAFF

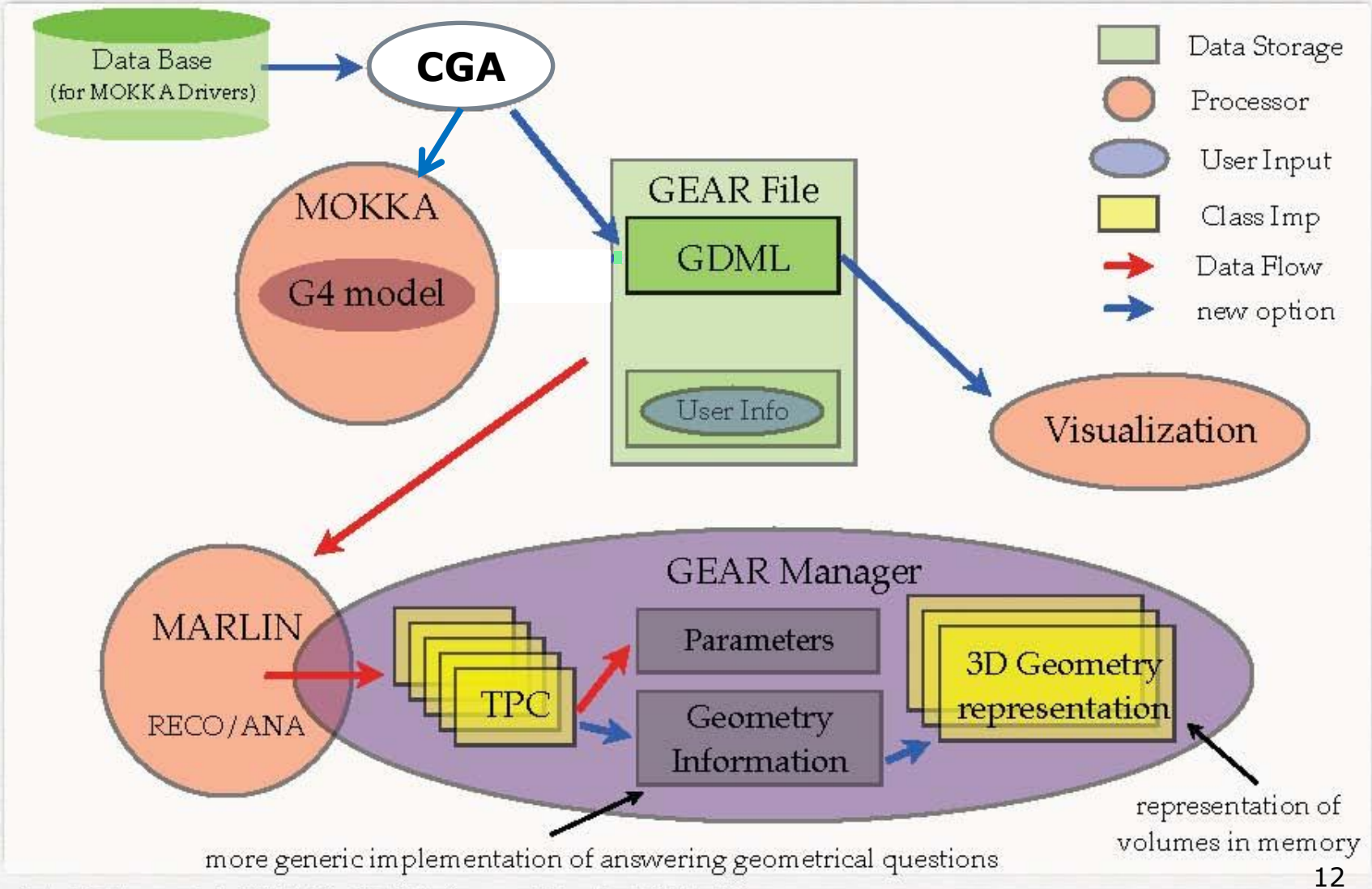


A new era, where simulation becomes just a client

Towards the GAFF



And you have it right now



Dumping the Geometry in GDML

- Mokka is built on top of CGA
- So just launch Mokka and type :

/Mokka/Visu/Detector/DumpGDML

to create the GDML file and have fun.

About the future: AIDA

- "A new generic HEP geometry tool"
- In my opinion :
  - It should start by collecting the user requirements
  - It should provide a high level user interface to create detector geometries (like CAD tools for engineers)
  - Mokka should be able to simulate these geometries, to insure a smooth transition

Waiting for the future: Gear2 ?

- Do we need to extend the current Gear interface?
  - Towards Gear2: to extend Gear to implement new geometrical parameters, not yet available
  - Should start by updating the user requirements for reconstruction, etc., before start coding

Do we really need it?

Acknowledgements

- Many thanks to Gabriel Musat, which has contributed a lot for GearCGA
- Many thanks to Frank and Astrid for the slides and ideas I reused here