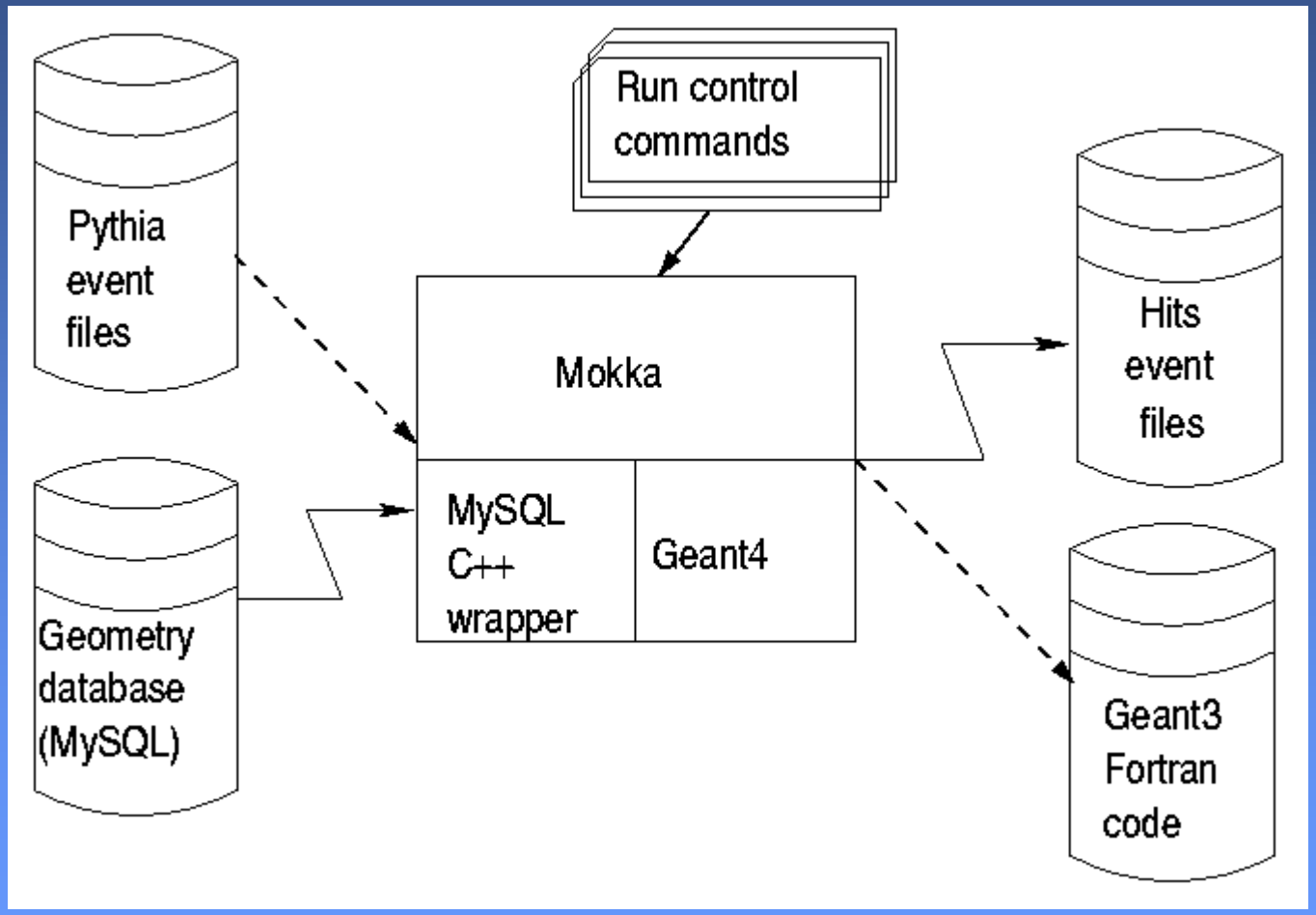
Geometry in Mokka

Workshop on Geometry Toolkit for the Linear Collider

24 February 2010 - CERN

Paulo Mora de Freitas - Ecole polytechnique

First releases

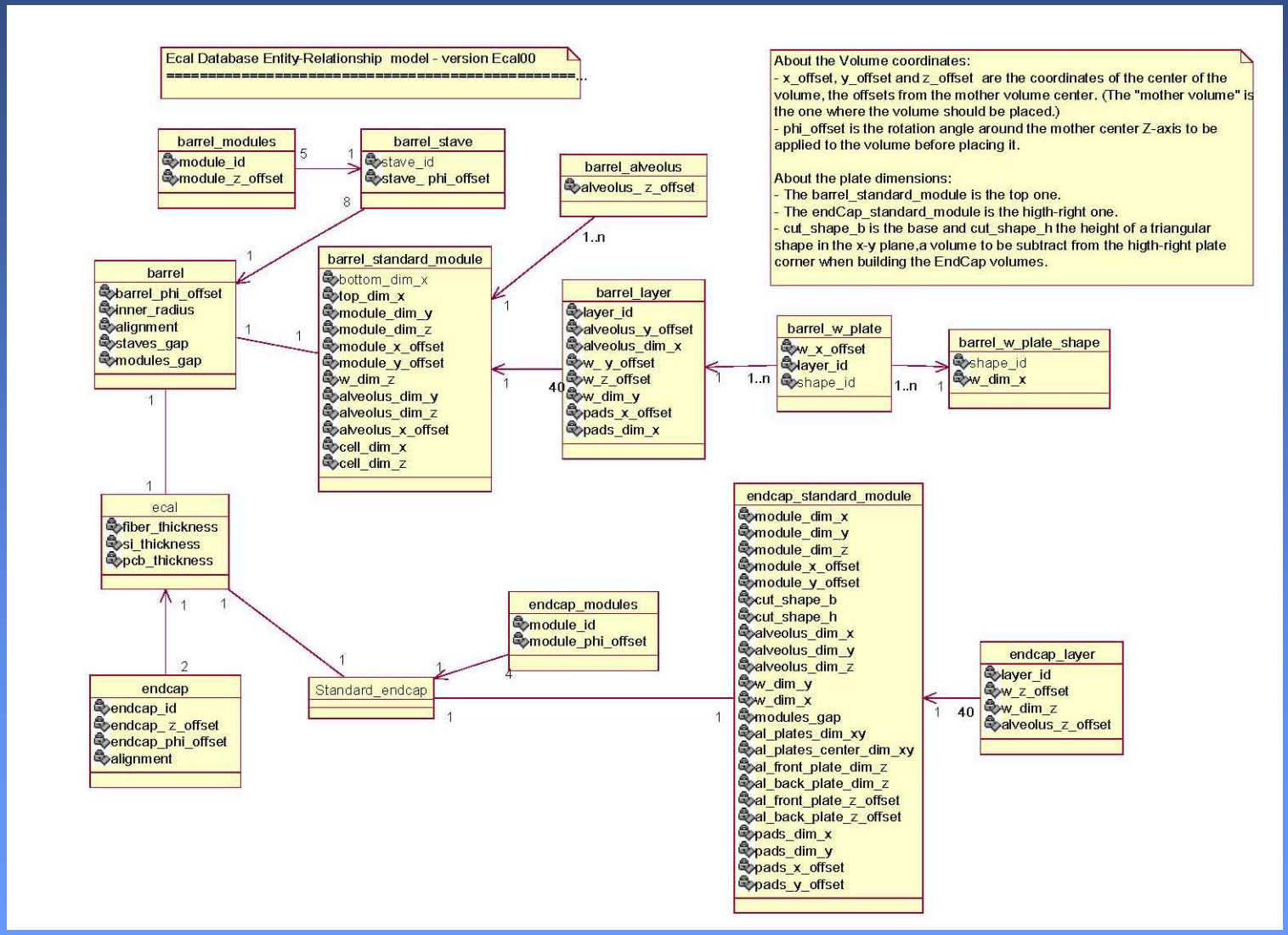


Why a Entity Relationship database

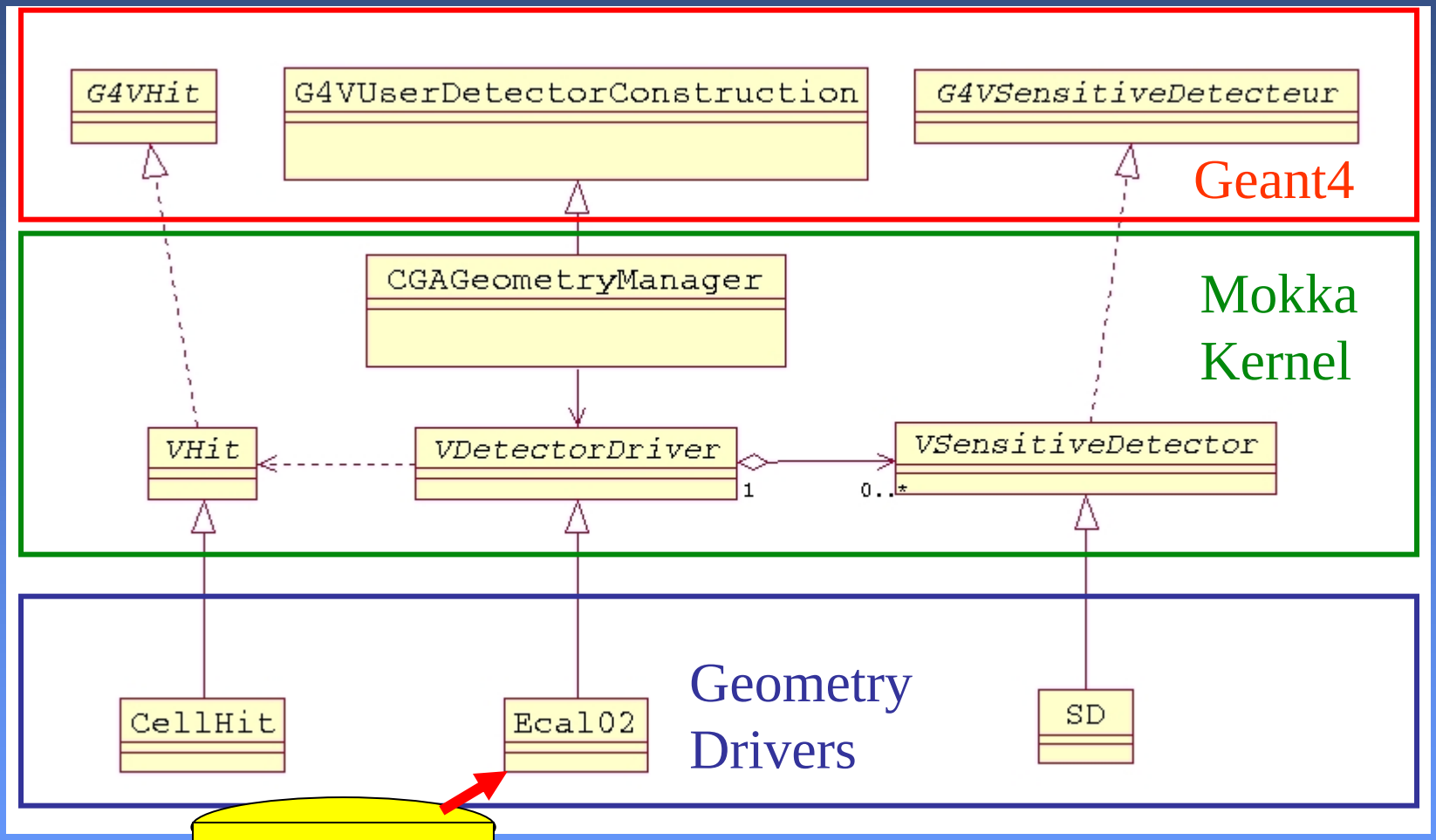
- To keep geometry values outside the code
- An central reference for detector studies
  - To avoid comparing software instead detector performances…
- To have a generic user interface
  - SQL implements a Data Manipulation Language (DML)
  - Several GUI and API are available for MySQL

"We cannot know what the users will need as geometrical information in the future"

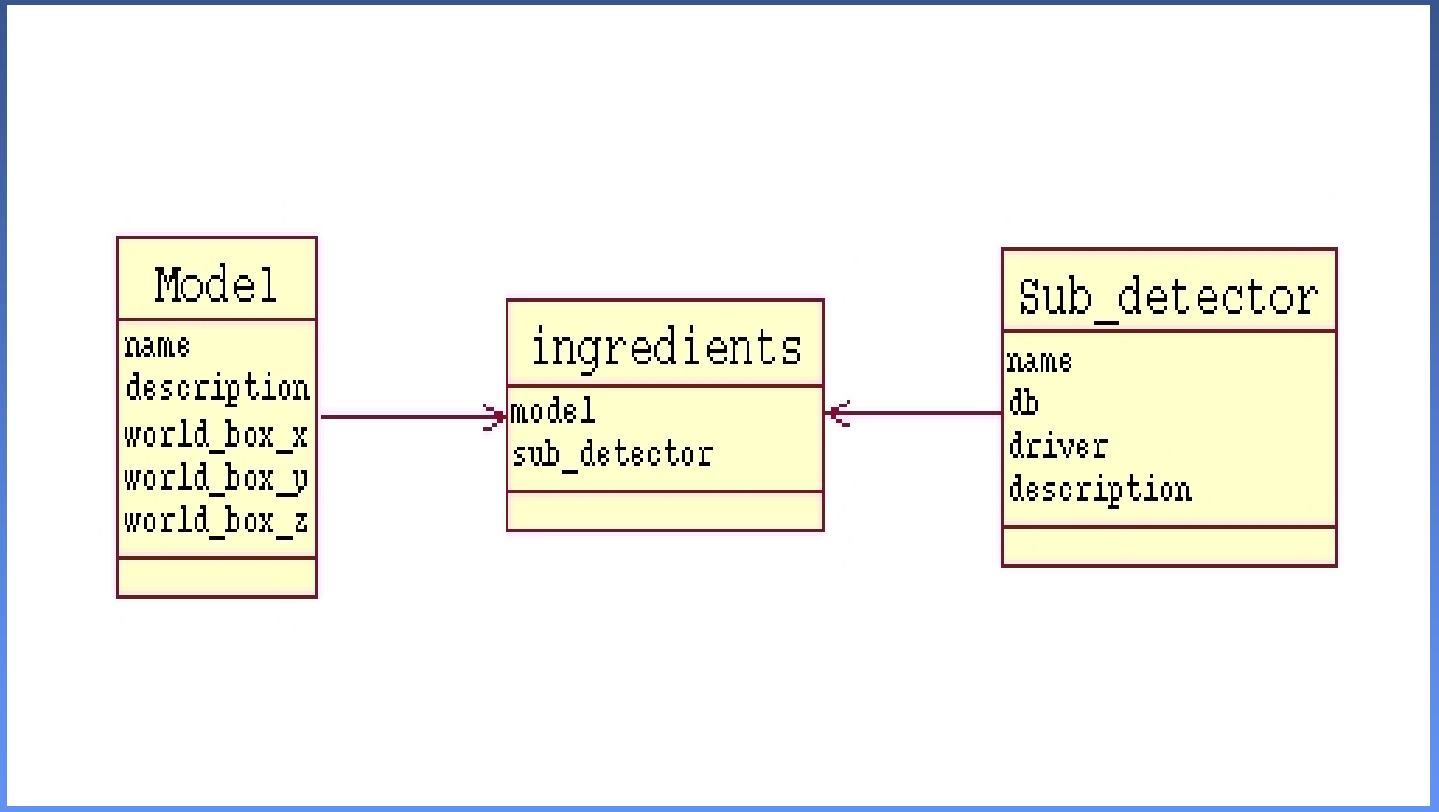
Ecal Entity-relationship model



Mokka's kernel framework



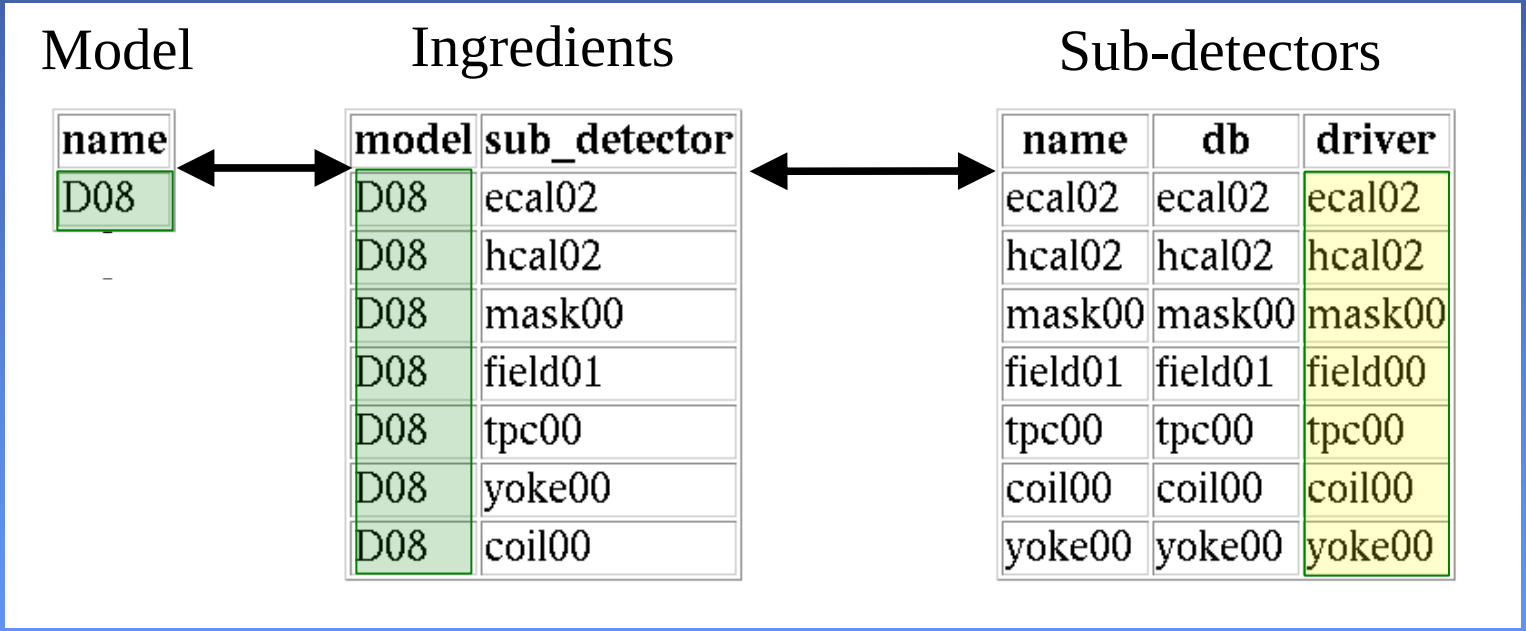
The detector models in DB



- A model = a set of sub detectors (TPC, Ecal, Hcal, etc.)
- A sub detector = a driver <-> DB association

The geometry database

A detector model sample: “D08”

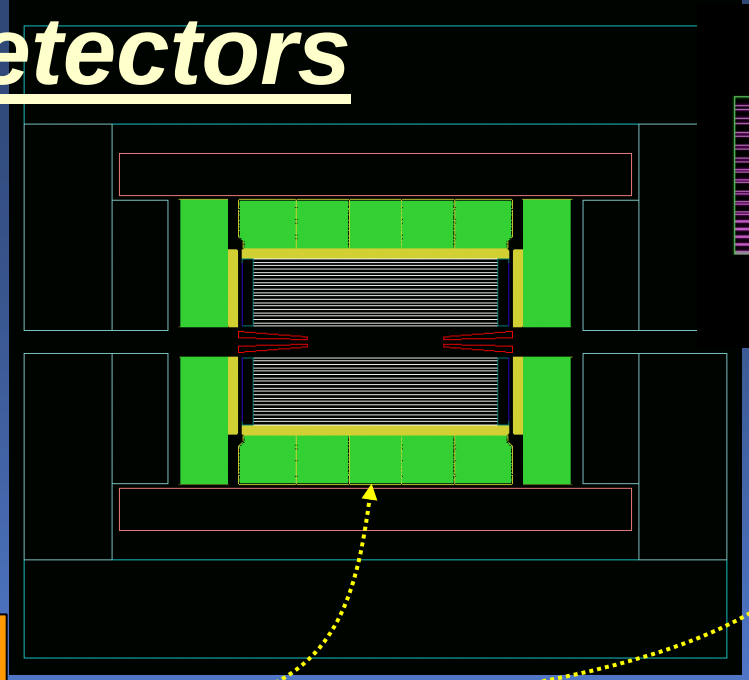


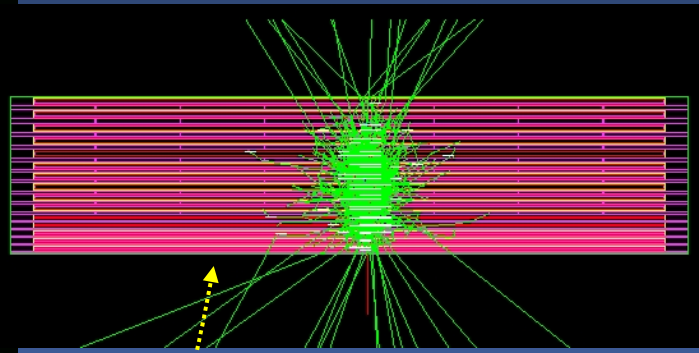
name
D08

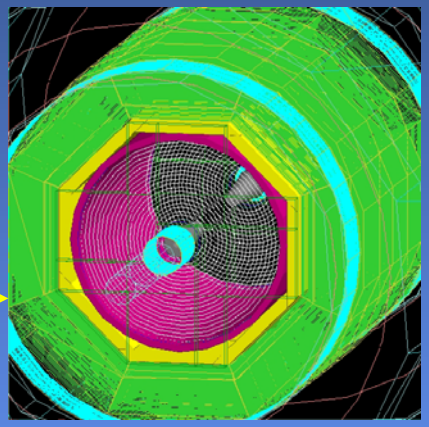
model	sub_detector
D08	ecal02
D08	hcal02
D08	mask00
D08	field01
D08	tpc00
D08	yoke00
D08	coil00

name	db	driver
ecal02	ecal02	ecal02
hcal02	hcal02	hcal02
mask00	mask00	mask00
field01	field01	field00
tpc00	tpc00	tpc00
coil00	coil00	coil00
yoke00	yoke00	yoke00

Mokka detectors



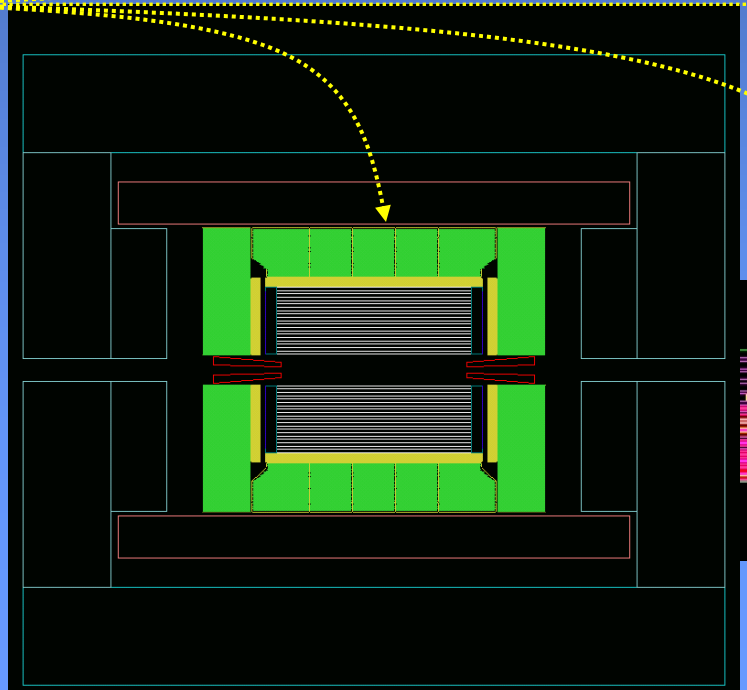


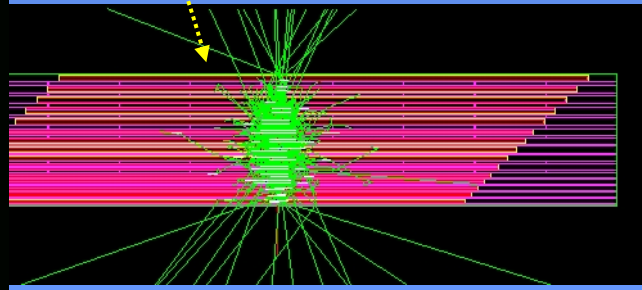


Mokka

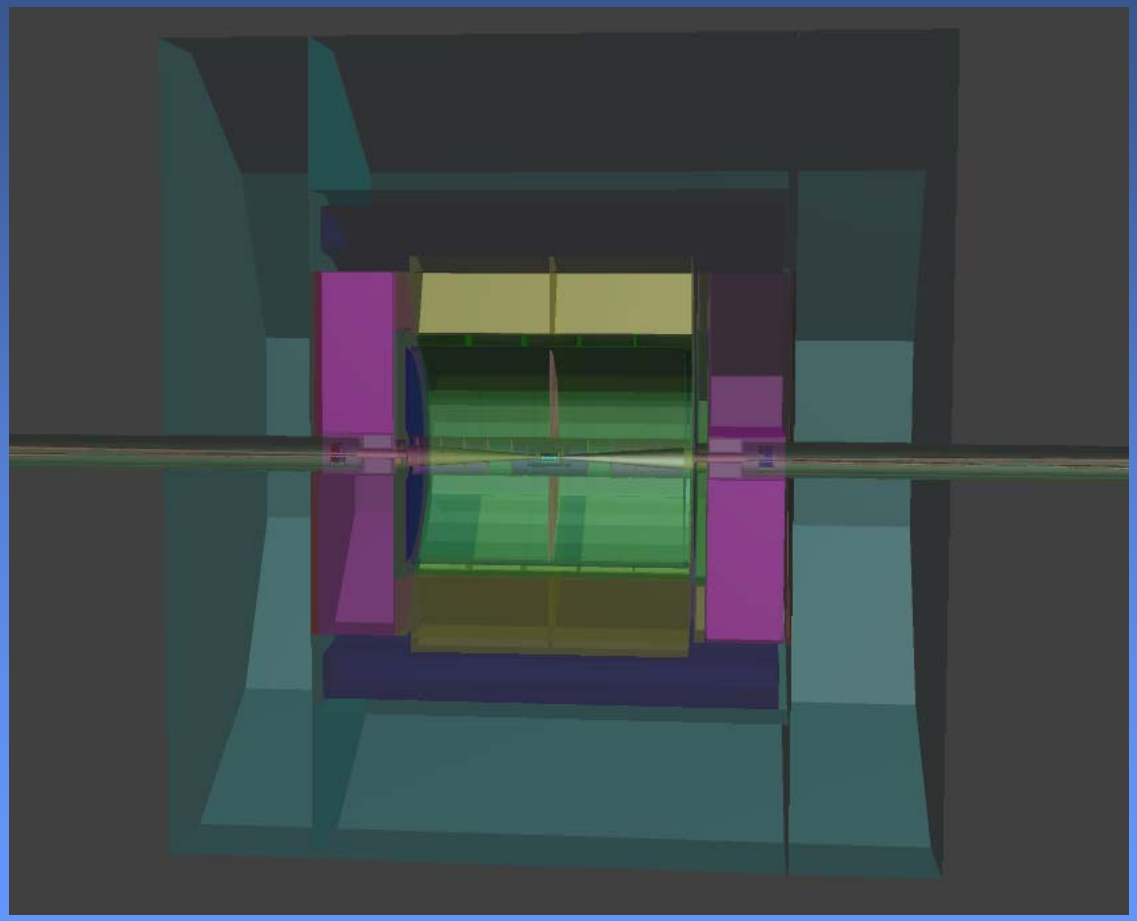
Geometry drivers

Geometry Database

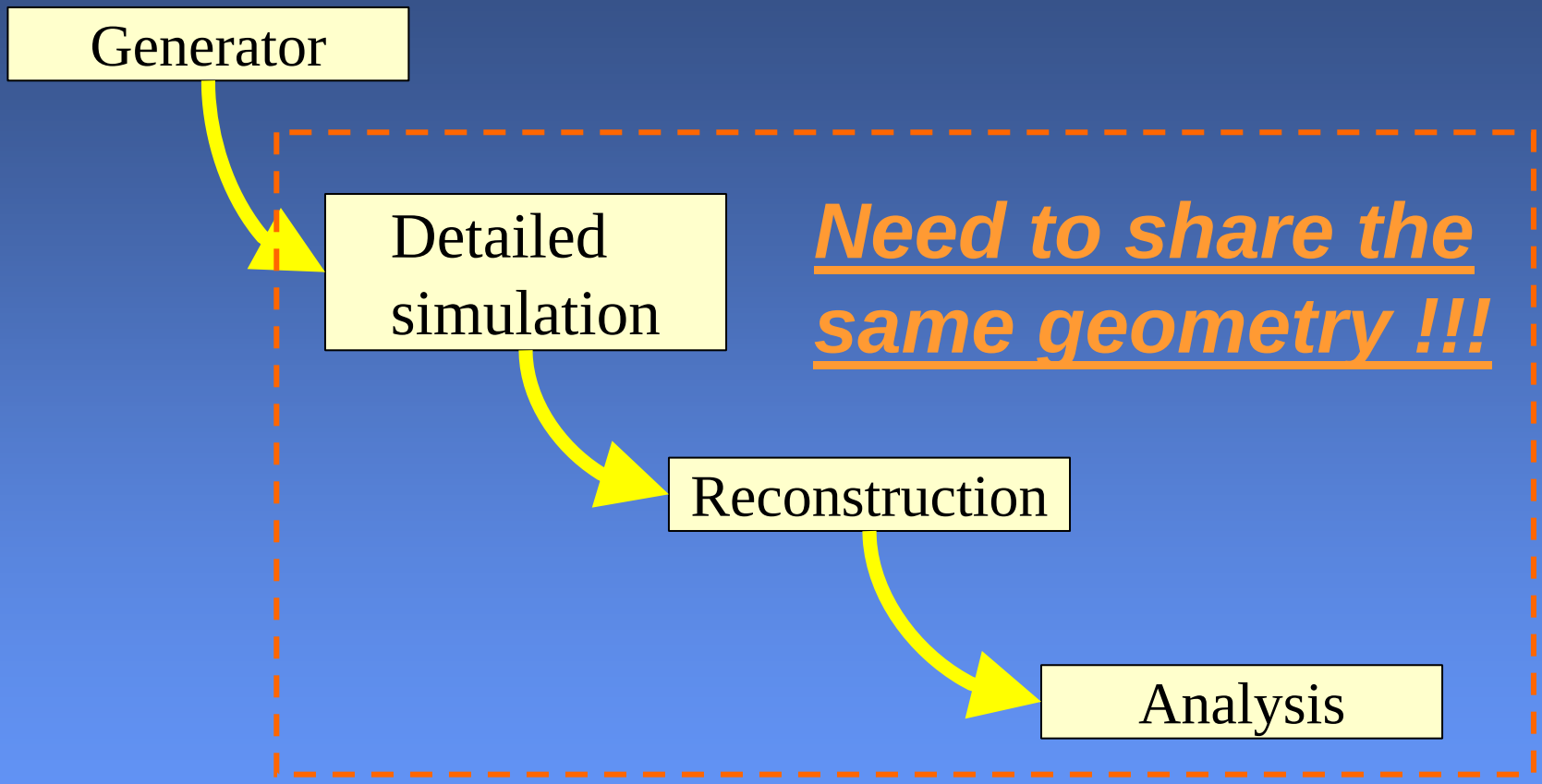




The ILD model today



Geometry have to be shared



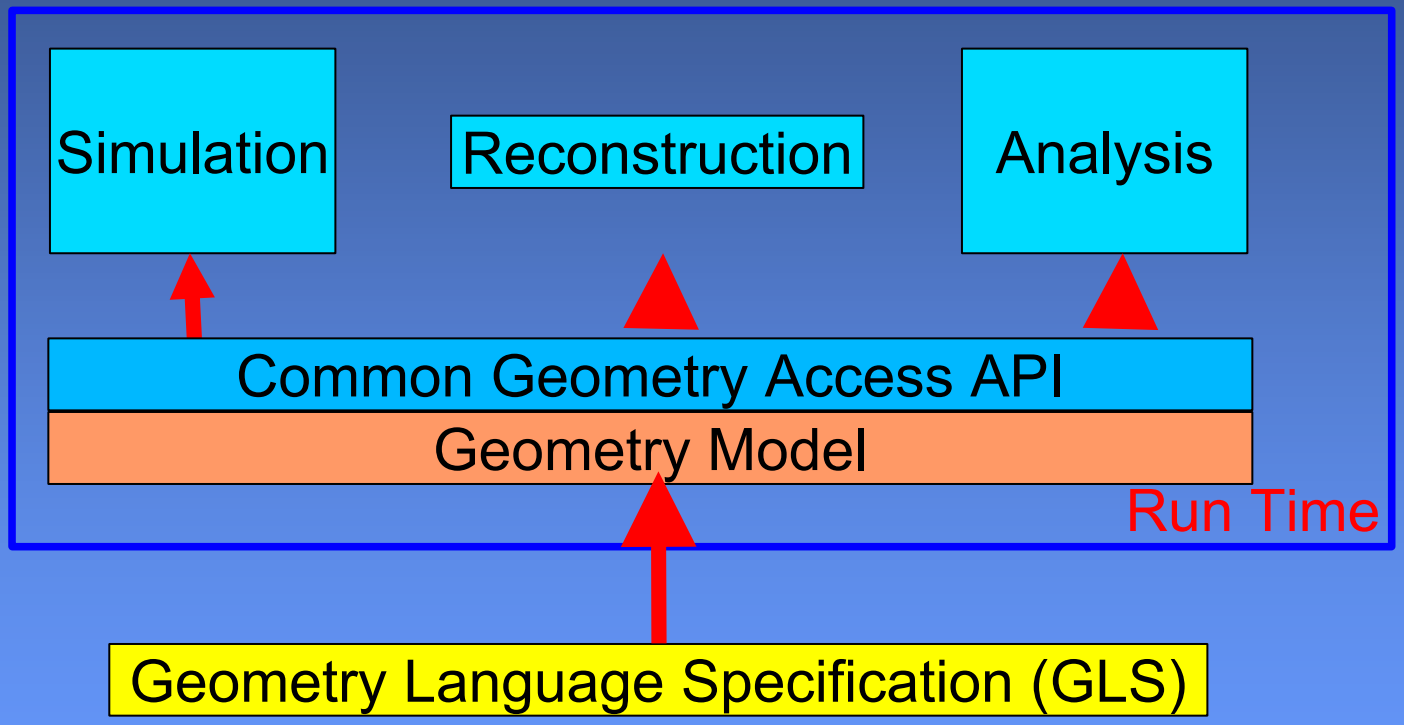
And nobody never tried to retrieve geometrical information via SQL queries, except in the geometry drivers in MOKKA !!!

Need of a standard geometry description ?

- Henri Videau talk at Saint-Malo
- “Specifications for a detector geometry description language”, 2002, Videau and Mora de Freitas

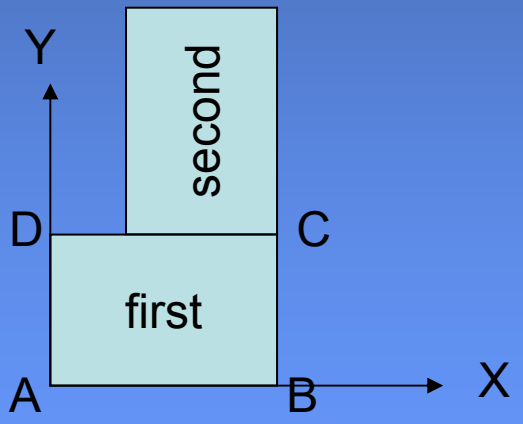
http://polype.in2p3.fr/geant4/tesla/www/index.html#dgdl

Geometry data sharing between processes



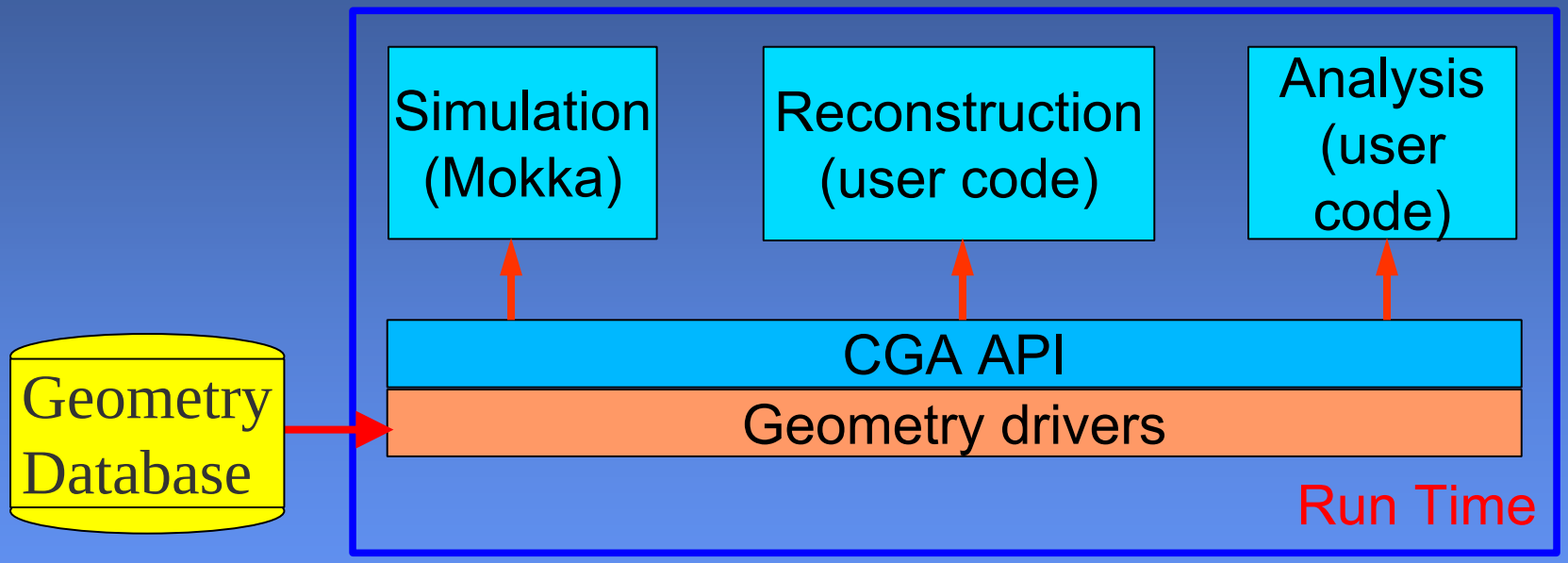
DGDL (2002)

- // Defines Box A with side =1
- BOX A DX= .5 DY = .5 DZ= .5
- // Defines Box B with side = 2, “DV” stands to “definition vector”
- BOX B DV=(1, 1, 1)
- //Defines C as A−B
- UNION C = A − B



- #Placement
- box first 0. 0. 3. 2.
- box second box first placement box first C B−C

Instead GDML: the Common Geometry Access API (GGA)

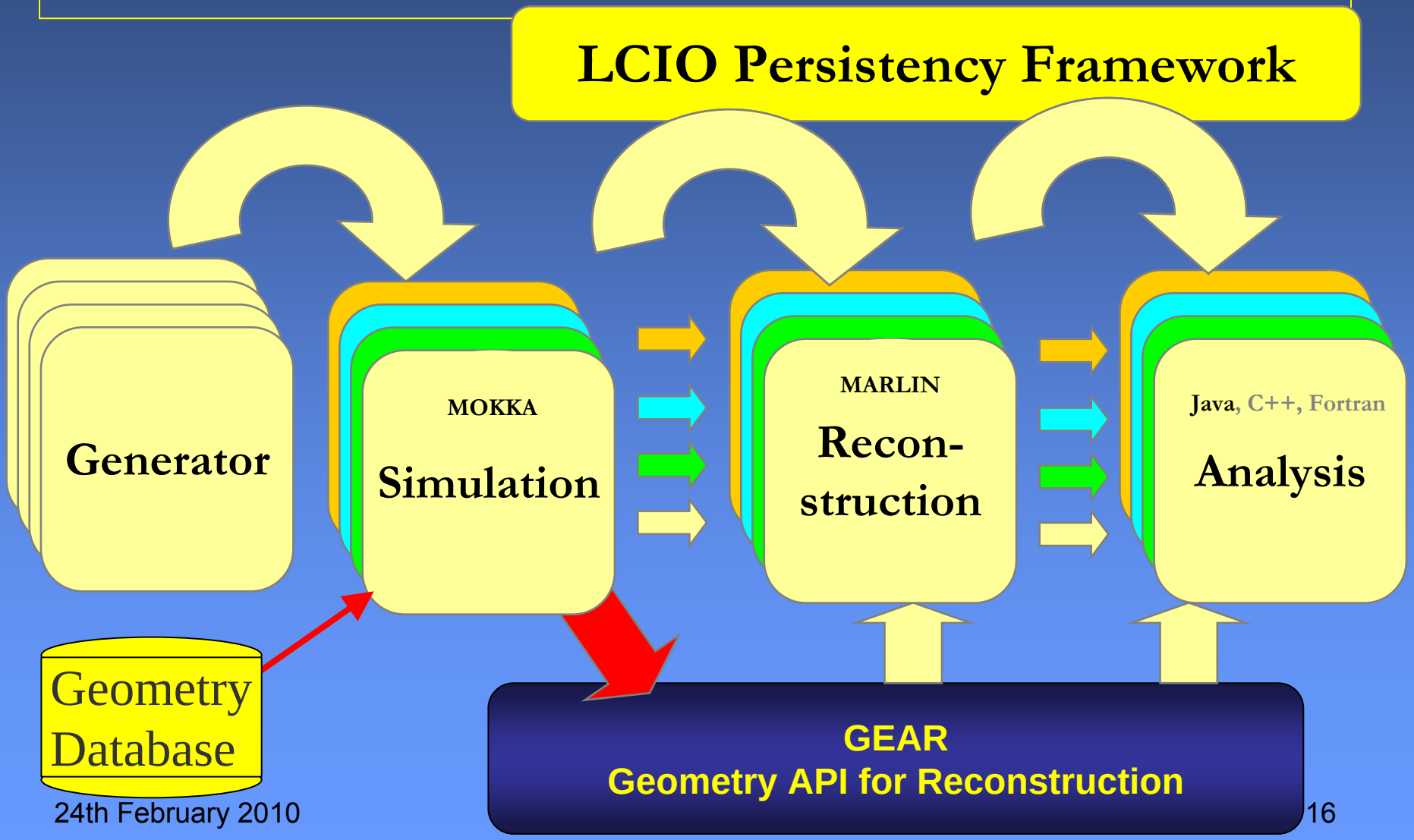


- Relies on Geant4 geometry layers
- Implements some reconstruction utilities
- Available for F77, C++,C and Java

About CGA

- Good features in CGA:
  - Automatic and cheap
  - Provides a full consistent geometry system for simulation, reconstruction and analyses.
  - Provides a scan mechanism to explore the detector geometry and material proprieties (geantino tracking)
- But…
  - Links against Geant4 libraries
  - Exports only low level geometry (G4 volumes level)
  - a few users only
- Available with Mokka since 2004, anyway

ILD Current framework



GEAR today

- Good features
  - Implements an higher level interface for the Geometry access than CGA (Ecal radius, etc.)
  - Abstract interface
  - Exploit CGA for point proprieties (X0, etc.) as an user option (nobody knows…)
  - ***Adopted by the community***
- But :
  - Implemented as an ad-hoc code in Mokka drivers (hard to maintain and to insure coherence)
  - Incomplete abstract interface doesn't cover all the user needs
    - Inflation of user's parameters, making the reconstruction code to depend on the simulator tool

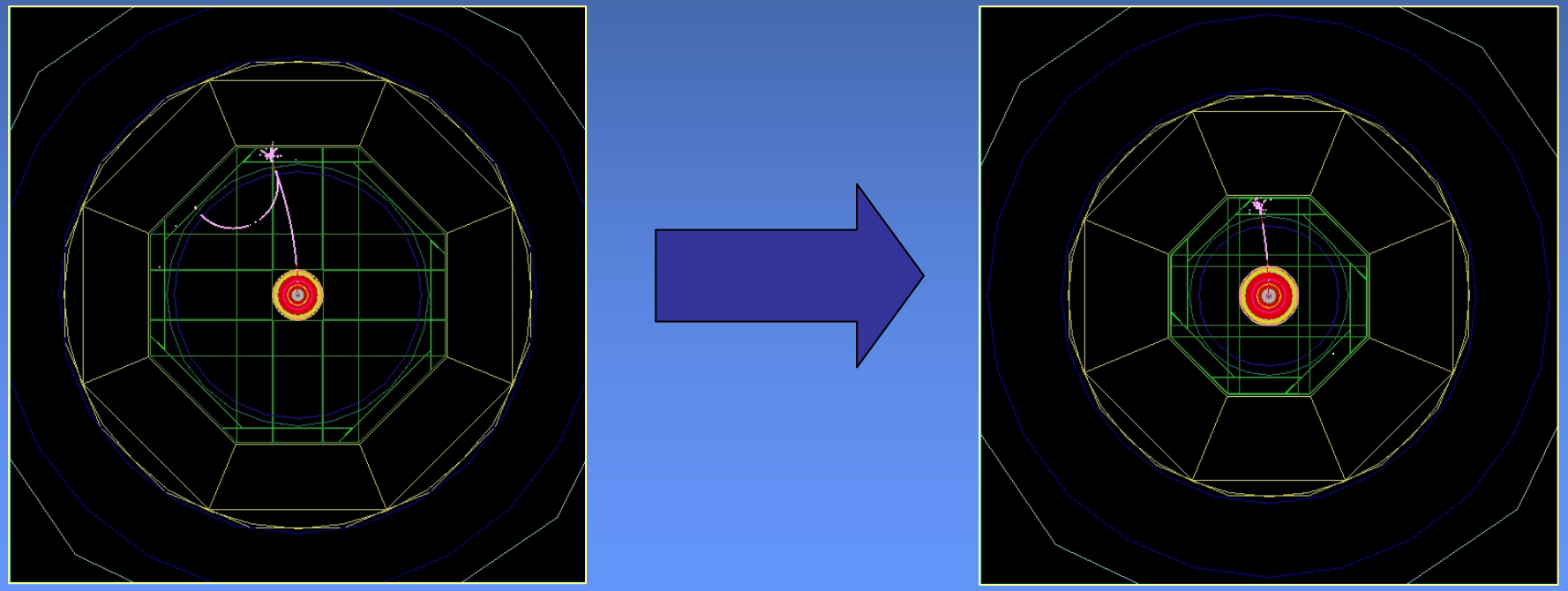
Mokka, other geometry features

- “Scaling”, the user is able to modify the model's main parameters at launch time
  - To easily be able to study different detector options, like TPC size, number of layers in calorimeters, etc.
- “Cooking”, the user is able to modify the model ingredients at launch time
  - To easily be able to study different detector technologies, like analogical versus digital Hcal, etc.
- To be useful:
  - Changes should be propagated for other devices
  - reconstruction should follow the changes introduced by the user in the Mokka steering files

Scaling

- Example :

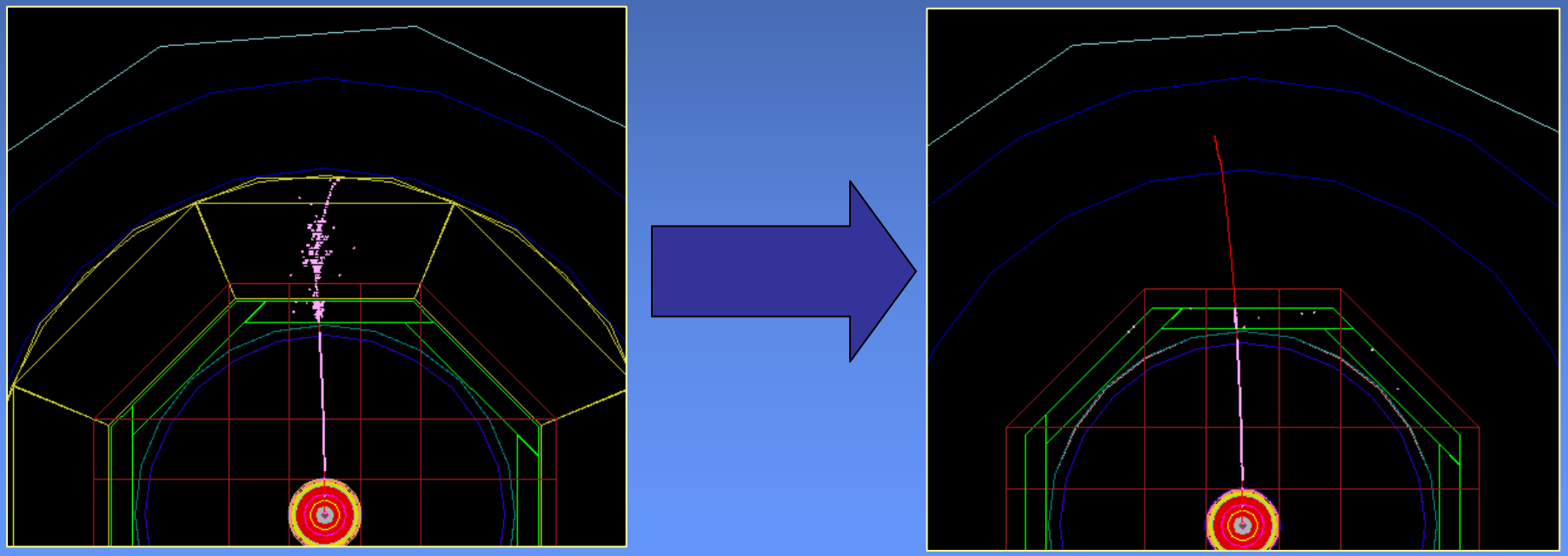
  /Mokka/init/globalModelParameter TPC_outer_radius 800



Cooking

- Example :

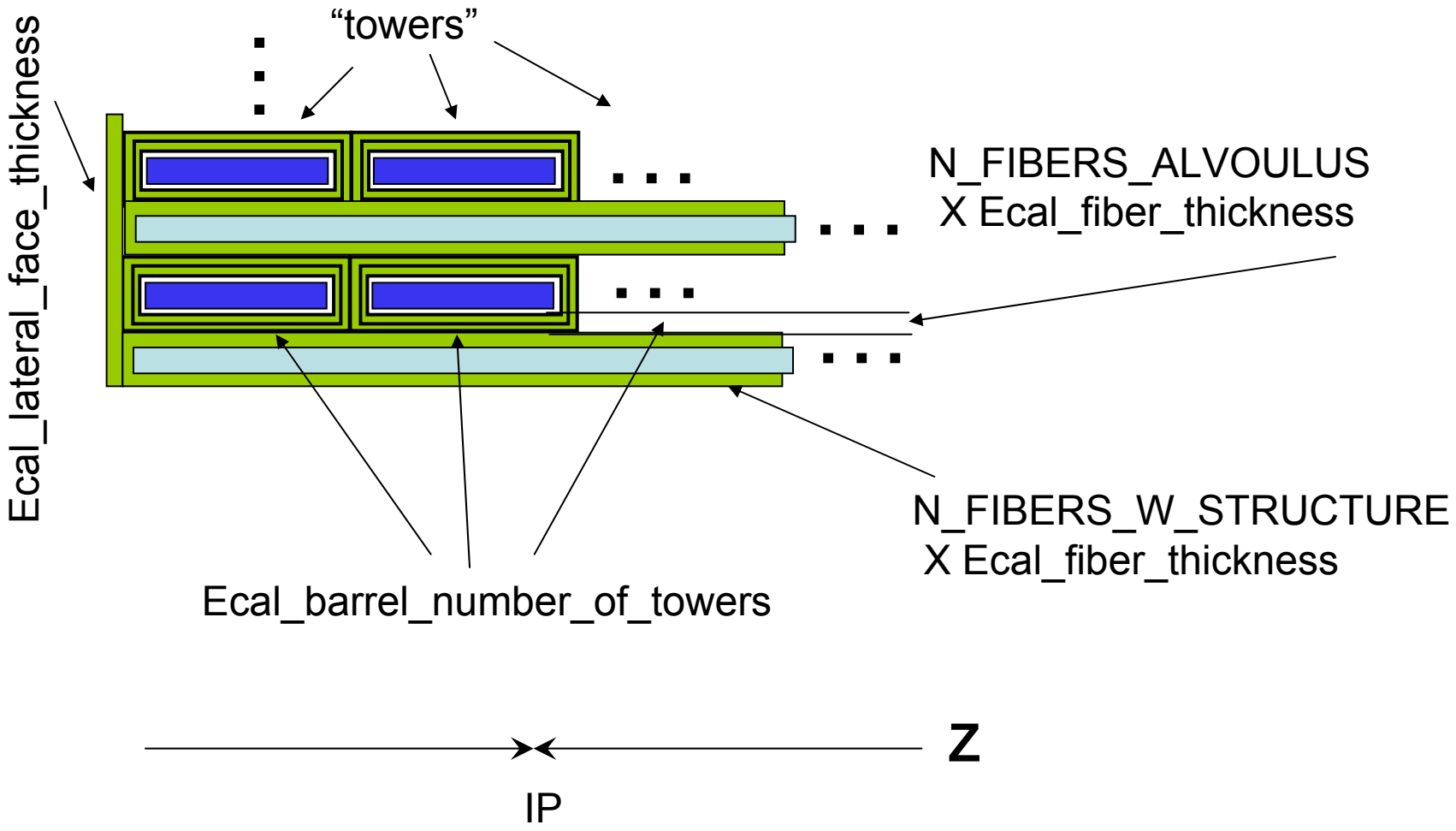
  /Mokka/init/EditGeometry/rmSubDetector SHcal01



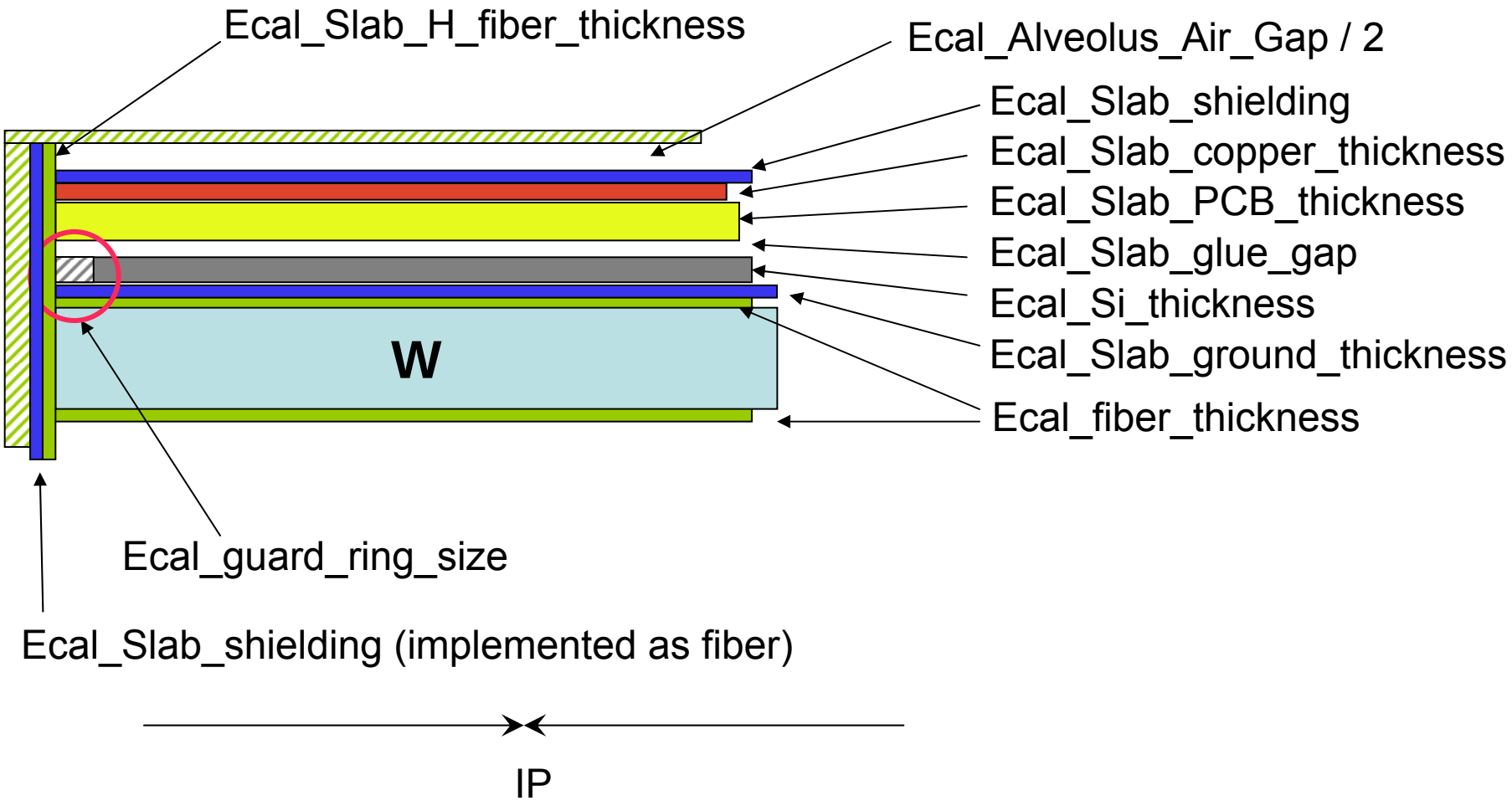
Scaling implementation

- Very hard to populate by hand the detector databases for complex devices like Ecal
- Historically done via MySQL scripts, in function of a few main parameters for the Ecal, like inner_radius, Ecal_nlayers, Si_thickness, etc.
- DB extension to keep all these main parameters, for all devices, and their dependencies when scaling
- Implementation of “Super Drivers” in MOKKA
  - First generation: able to create on the fly a temporary database for the old geometry drivers
  - Currently : able to create directly the device

Ecal “towers” in modules
(1 slab per alveoli, 2 wafers per slab in Z) :



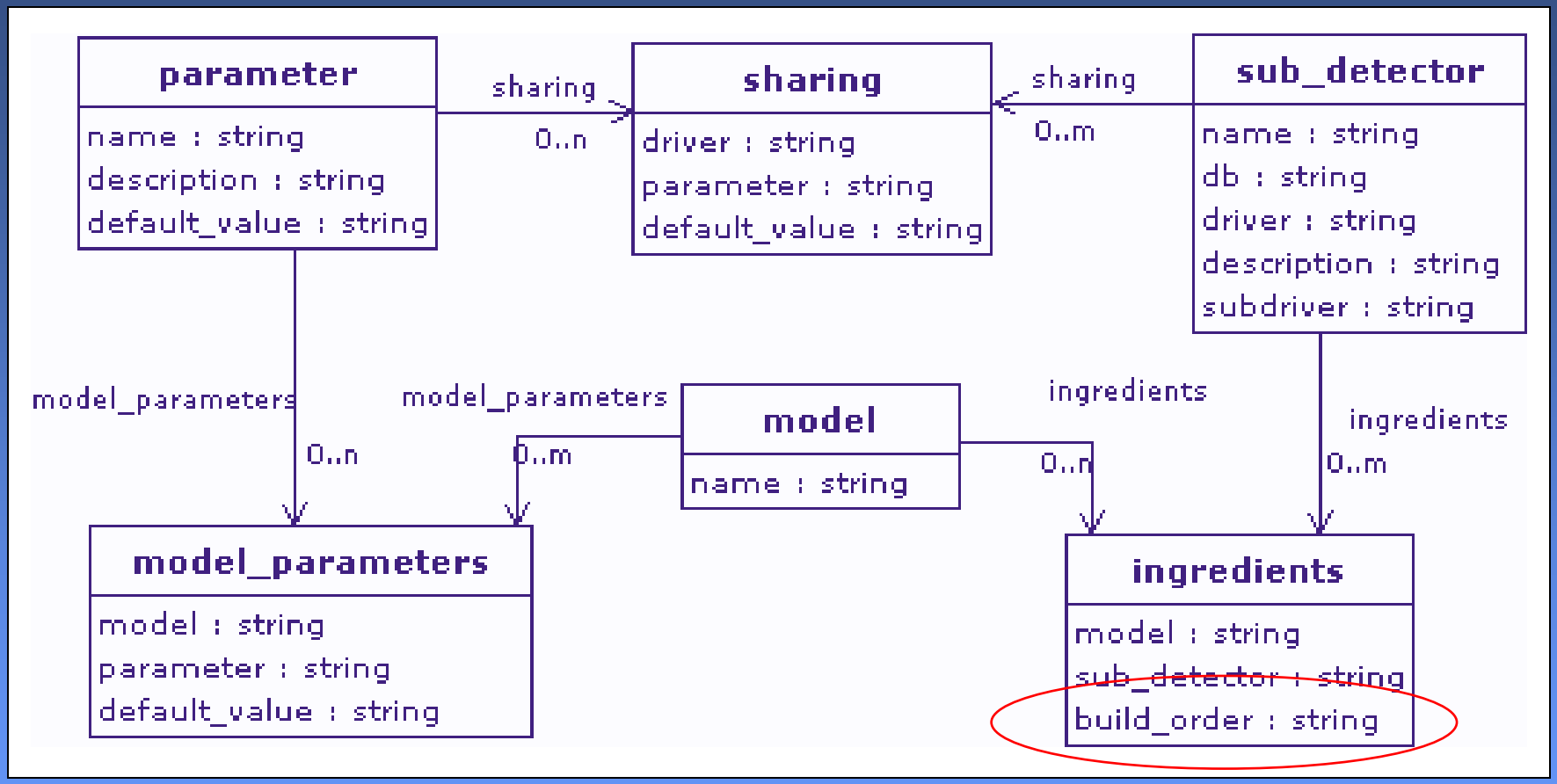
Detail Alveoli & “H” slab structure:



Example of a MySQL script (Ecal)

```
#
# initialisation des tables
#
delete from ecal;
insert ecal set
   fiber_thickness = 0.1,
   si_thickness = 0.5,
   pcb_thickness = (2.1 - si_thickness)/2.;
…
```

Parameters in Mokka DB



- Parameter values are overwritten by the sub_detector default, then the model_parameter default, then steering file value if any, and finally by the environment value if modified at run time by a previous driver.
- Scaling follow the model build_order in ingredients

Proposals

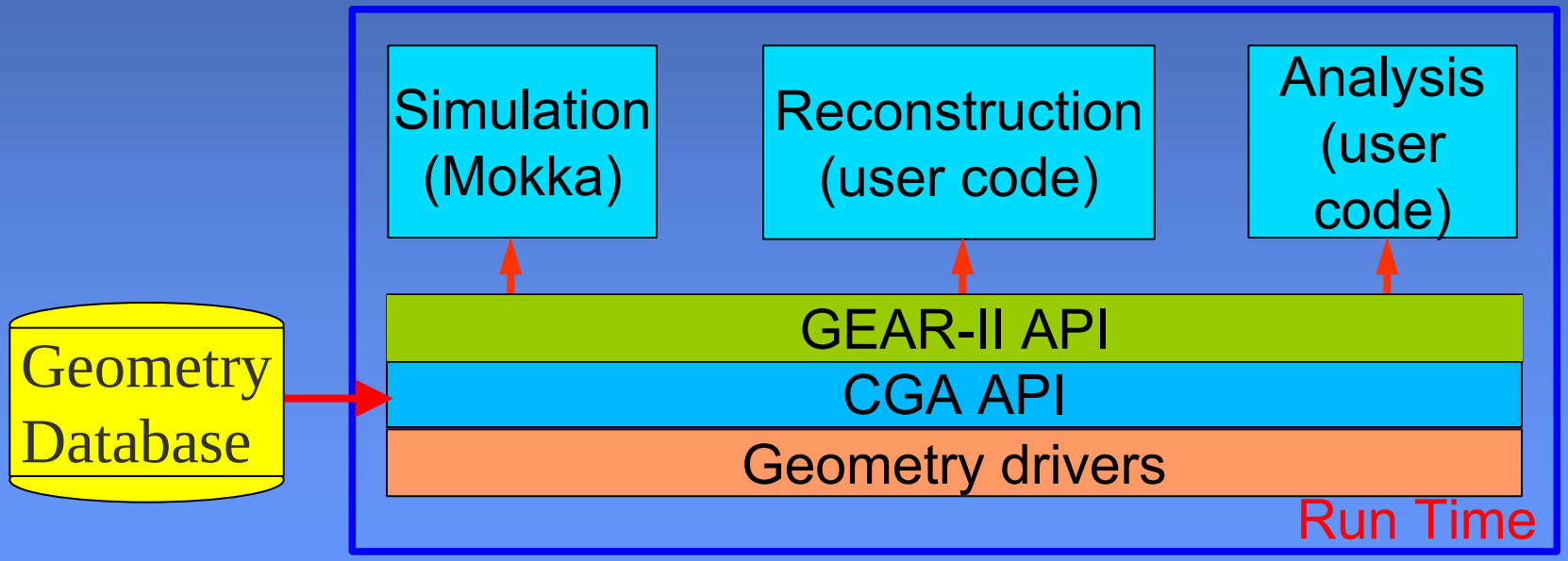
- LC people simulating the ILD model have to survive with MOKKA for a while
  - ➢ short term improvements for sharing geometry for LC studies
- MOKKA history rises a set of user's requirements for the Geometry Toolkit
  - ➢ To be generic, flexible and friendly for the final user, etc.

Proposals - I (short term for ILD)

- Very low expensive, easy and short term improvement for sharing geometry for ILD
  - For each sub detector, automatically to export in the Gear file all the actual values used as parameters, as “user parameters”
- Parameters well described in DB and Mokka output: easy for the user
- Can be done right now
  - ***But the reconstruction code remains depending on the simulation tool***

Proposal - II (short term for ILD)

- A more expensive short term solution:
  - To extend the Gear abstract interface
  - To implement it on the top of CGA (*)



(*) no more ad-hoc, but as part of the geometry driver responsibility

Ideas for the LC geometry tool

- To adopt a CAD graphic interface:
  - No more script language neither XML edited by hand
  - See Open Cascade (http://www.opencascade.org)
- Abstraction levels for the detector geometry:
  - To cope with the different views for simulation, reconstruction, analysis, event displays…
  - To use metadata to describe the abstraction levels, to keep the tool as generic as possible
- To define an abstract API (C++, Java, etc.)
- To provide a Data Manipulation Language (DML)?
- To provide at least one implementation for this API
- To provide at least one simulation/reconstruction/ event display chain compliant with it.