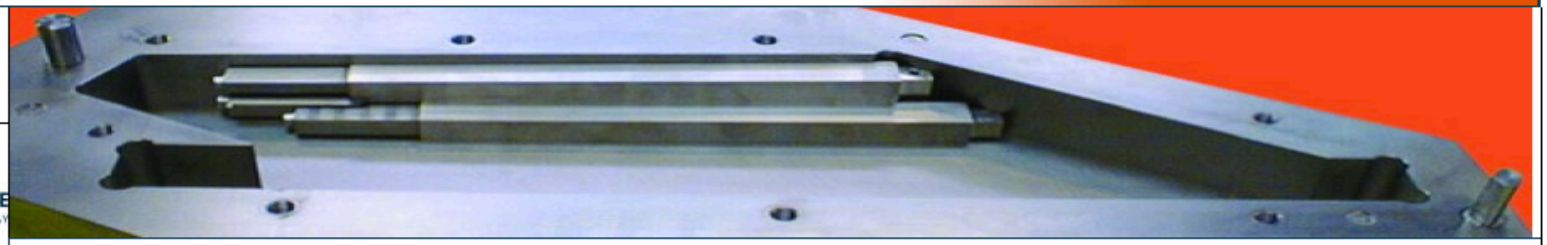
Laboratory Leprince-Ringuet (LLR)

- Palaiseau, 20 km from Paris at the campus of Ecole Polytechnique
- Many relationships with University Paris Sud
- Scientific activities: High Energy Particle Physics (CMS, ILC/ILD, T2K, PHENIX, Babar), Astrophysics and High energy gamma ray (HESS, Fermi, CTA …), GALOP (plasma-lasers), Medical Physics, Geant4 (Marc Verderi)
- GRID activities: computing center with 1500 computing cores and 1PB of storage

llr.in2p3.fr

Laboratoire Leprince-Ringuet
UMR (7638)

Le CNRS | Autres sites CNRS



- Le LLR
- Activités Scientifiques
- Activités Techniques
- Stages, thèses et enseignements
- Communication
- Recrutements au LLR
- Seminaires

Twitter

Annuaire

Rechercher ok

UMR 7638
Laboratoire Leprince-Ringuet
Ecole polytechnique
91128 PALAISEAU Cedex
France

Toutes les "A La Une"

À la une

27 juin 2014

Conseil Scientifique

Le prochain conseil scientifique du LLR se déroulera le mercredi 16 juillet de 9h30 à 12h30 dans l'amphi Becquerel.

Lire la suite

18 juin 2014

Premier événement "anti-neutrino" observé par T2K !

Le premier faisceau d'anti-neutrinos muoniques produit par l'accélérateur JPARC a donné naissance à cet événement enregistré dans le détecteur Super-Kamiokande.

Lire la suite

Toutes les actualités

Actualités

23 juillet 2014

Rencontre de l'infiniment grand à l'infiniment petit, « promotion Frédéric et Irène Joliot-Curie ».

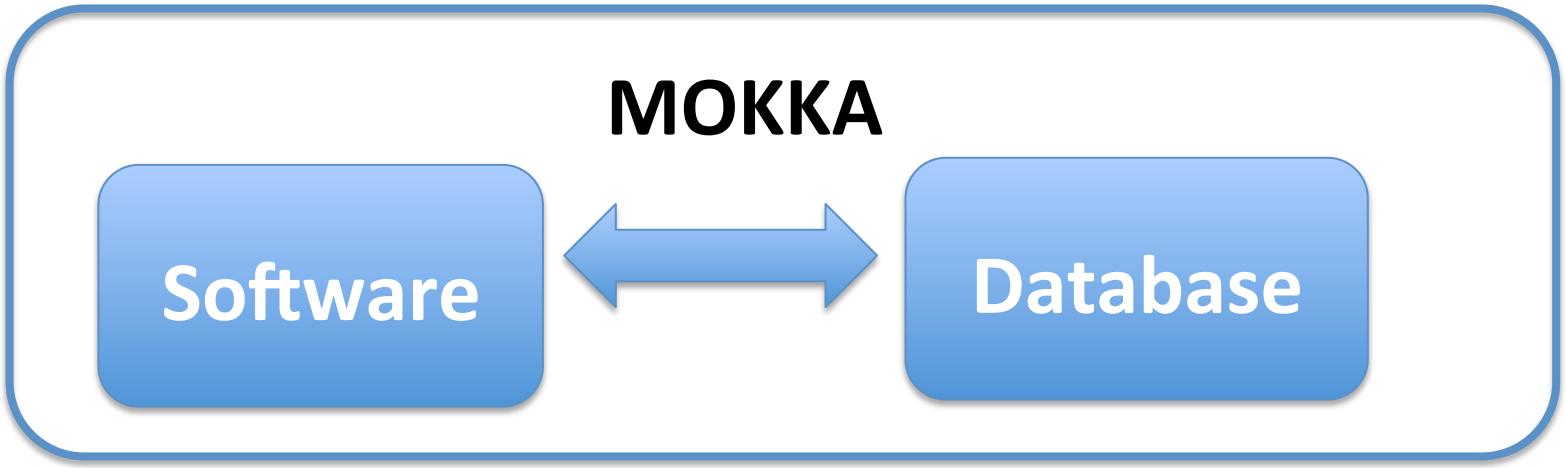
Pour la quatrième année consécutive, le laboratoire, faisant partie du comité scientifique de ces rencontres de physique (subventionné par le labex P2IO), accueille pour une journée une trentaine d'étudiants en fin de L3. Hier matin, après une matinée de cours dans une ambiance conviviale, ils ont pu échanger avec des chercheurs et doctorants du LLR et participer à une visite virtuelle de CMS tout en communiquant en direct avec le CERN. Cette journée s'est clôturée par une rencontre-débat sur le (...)

Lire la suite

3 juillet 2014

H.E.S.S.-II, le plus grand observatoire gamma au monde, détecte son premier pulsar

Installé en Namibie, l'observatoire H.E.S.S.-II vient de détecter des milliers de rayons gamma en provenance du pulsar Vela situé à environ 1 000 années-lumière de la Terre dans la Voie Lactée. Grâce à un nouveau télescope géant, il a ainsi repéré son premier pulsar, une étoile à neutrons qui correspond au coeur effondré d'une étoile



I. General presentation of Mokka software and database architecture, overview relationships between software and database

II. Modifying detector geometry using steering commands

III. Creating new geometry with new drivers

Get better understanding of internal structure of the code, learn how to create a driver template

IV. Creating new geometry model in the database

Mokka software and database architecture

Emilia BECHEVA
Laboratoire Leprince-Ringuet – Ecole Polytechnique, CNRS

Outline

- Motivations for Mokka and general presentation
- Mokka SW-Relationships: Geant4, MySql database,
- Mokka SW architecture
- Presentation of Mokka svn repository
- Motivations for MySQL database
- The models and the parameters in the database

- Mokka geometry features

- Getting started with Mokka – practical information

Motivations for Mokka

- Have a realistic simulation for ILC detectors
- Simulate final detector and prototypes in a common framework

- Insure the same conditions for the simulation

Share: physics
tool for geometry construction
Geant4, Common database

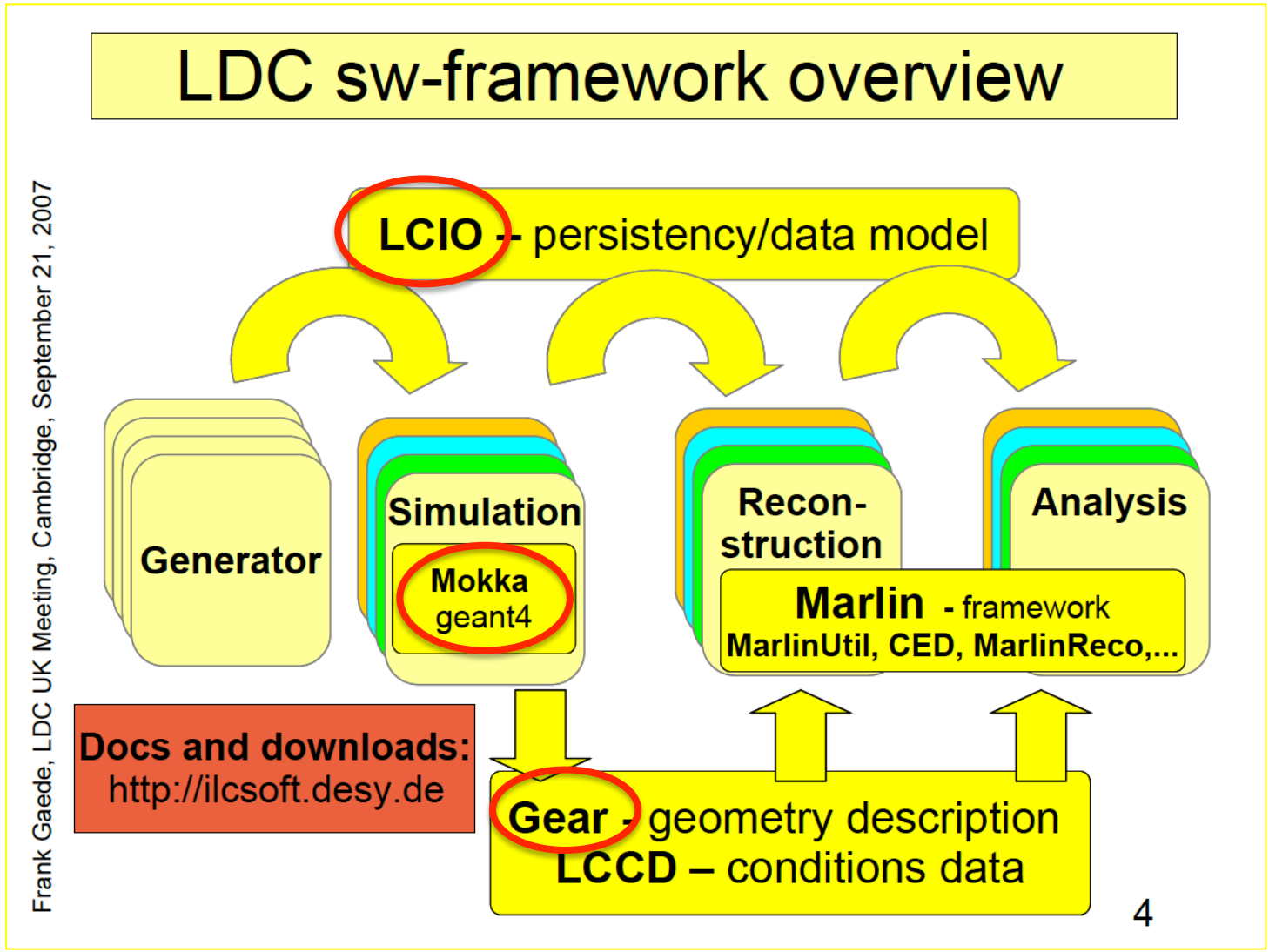
same input data files
same output data format
High-energy physics events
LCIO, ASCII

What is Mokka?

Modellierung mit **O**bjekten eines **K**ompakten **Ka**lorimeters
Object Modeling for compact calorimeters

- Mokka is a full simulation framework for the **International Linear Collider detectors**
- Mokka is strongly based on the **Geant4 framework**
  - Geometry simulation
  - Particle transportation
- Mokka's detector data driven model is strongly based on **MySQL**
  - Store models information
  - Store geometry parameters' values
- Development started at LLR in 1999: Paulo Mora de Freitas, Henry Videau and Jean-Claude Brient joined later by Gabriel Musat
- Several International teams (DESY, Cambridge University, Tokyo University, ...) joined Mokka collaboration

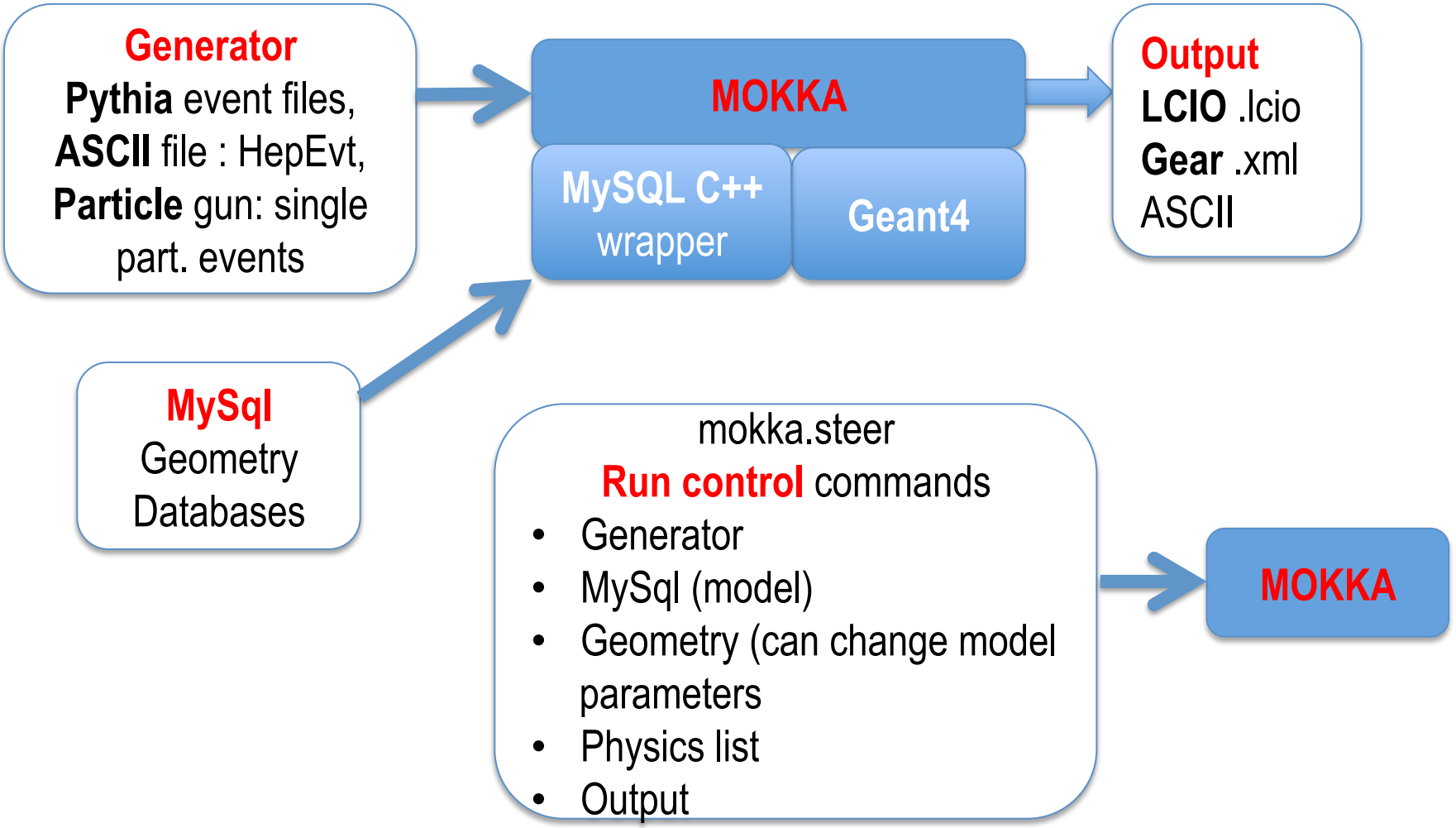
Mokka and ILCSOFT



LCIO: persistency framework and data model

Gear: **Ge**ometry **A**pplication for **R**econstruction

Mokka SW-Relationships



Mokka and Geant4

Mokka is a regular Geant4 application
⇒ Mokka main() programme creates an instance of RunManager class

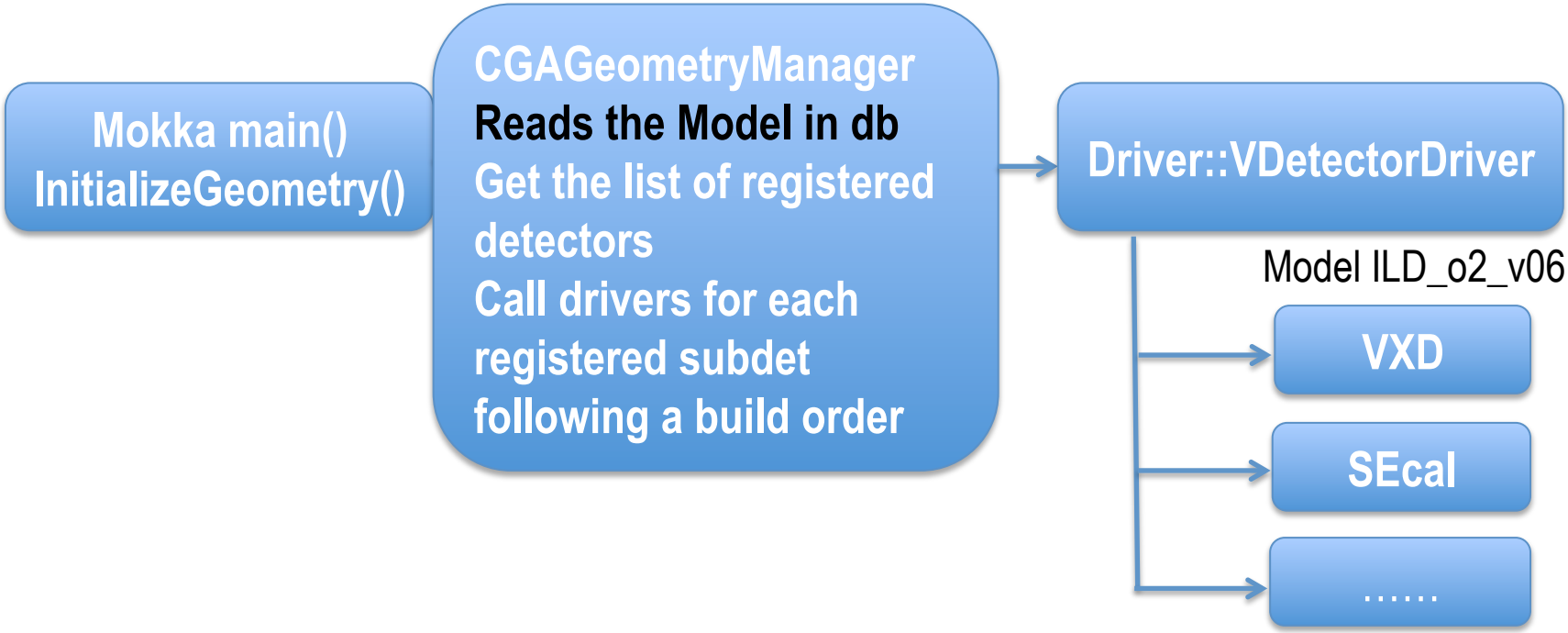
Initialize
Geometry => special class CGAGeometryManager : G4VUserDetectorCnstruction
Physics list => default physics list (hard coded): **QGSP_BERT**

Set user classes: PrimaryGeneratorAction
StackingAction
TrackingAction
SteppingAction
EventAction
RunAction

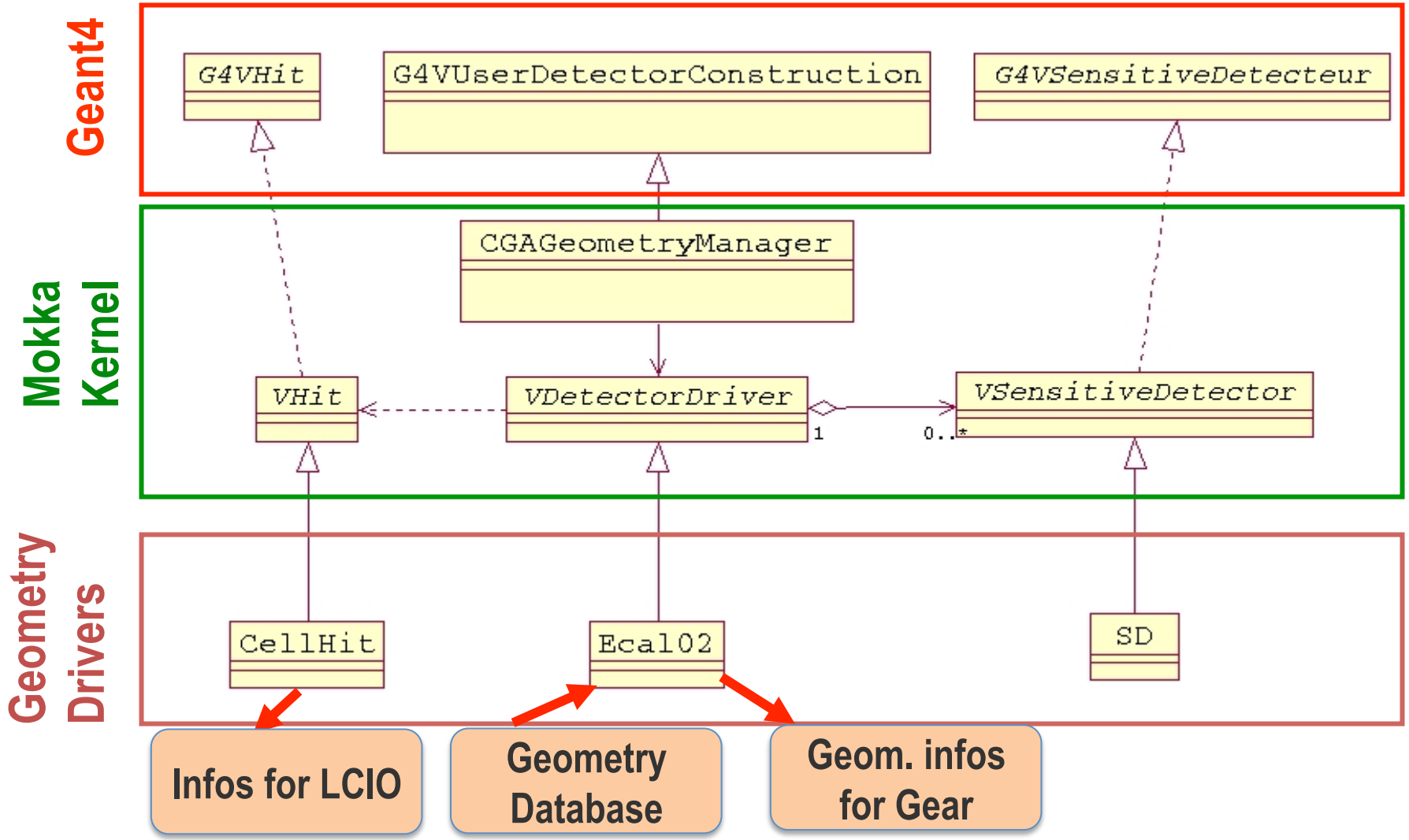
Messenger classes derived from G4 messenger classes: specific commands for geometry construction, visualisation, physics list management exist but the G4 native UI commands for visualisation etc.

Geometry construction in Mokka

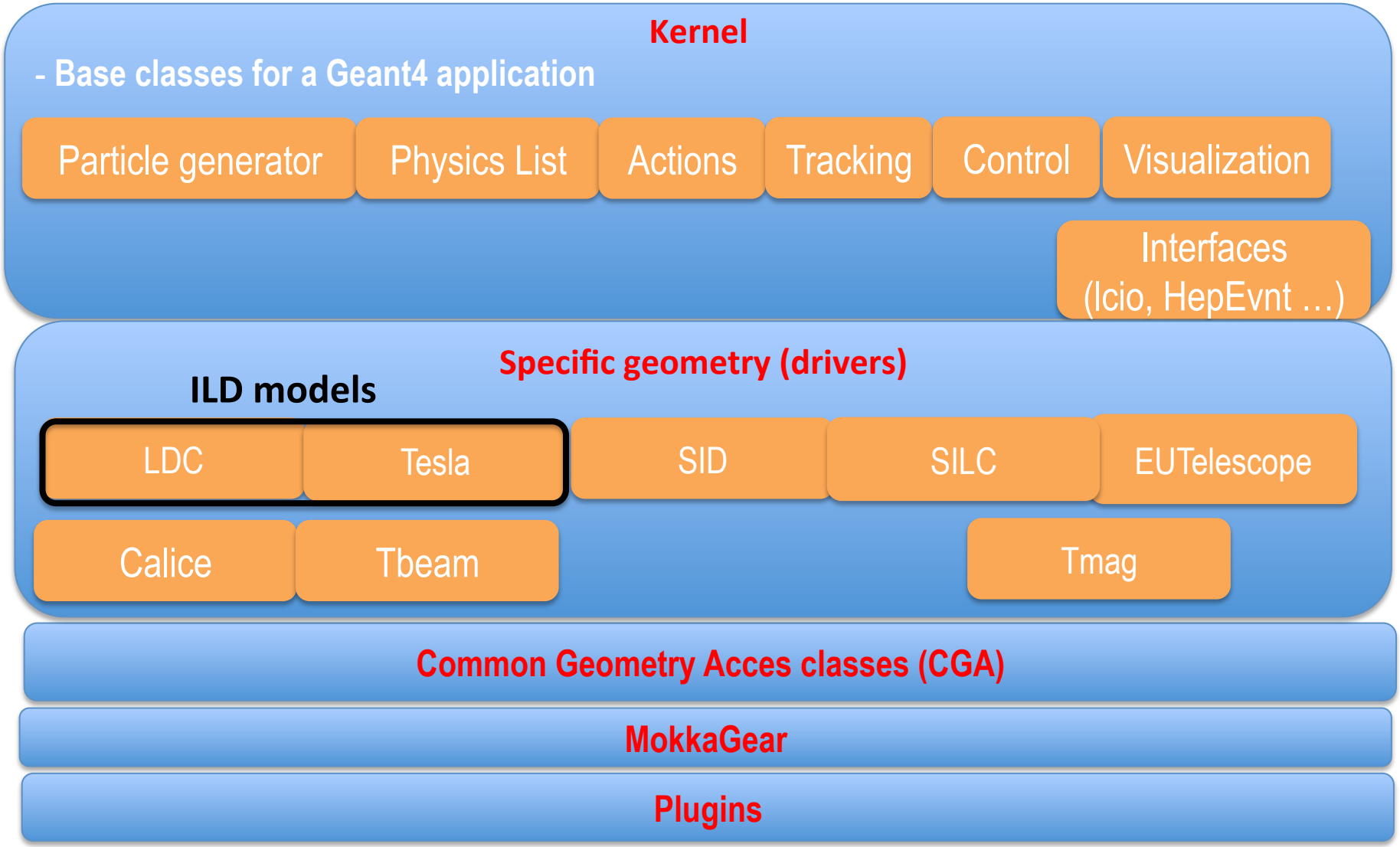
Geometry => special class CGAGeometryManager : G4VUserDetectorCnstruction managing the subdetector construction
The geometry construction for subdetectors is performed by drivers (C++ code) that reads the parameters from the database, there is one driver per subdetector



Mokka's framework



Mokka SW-Architecture

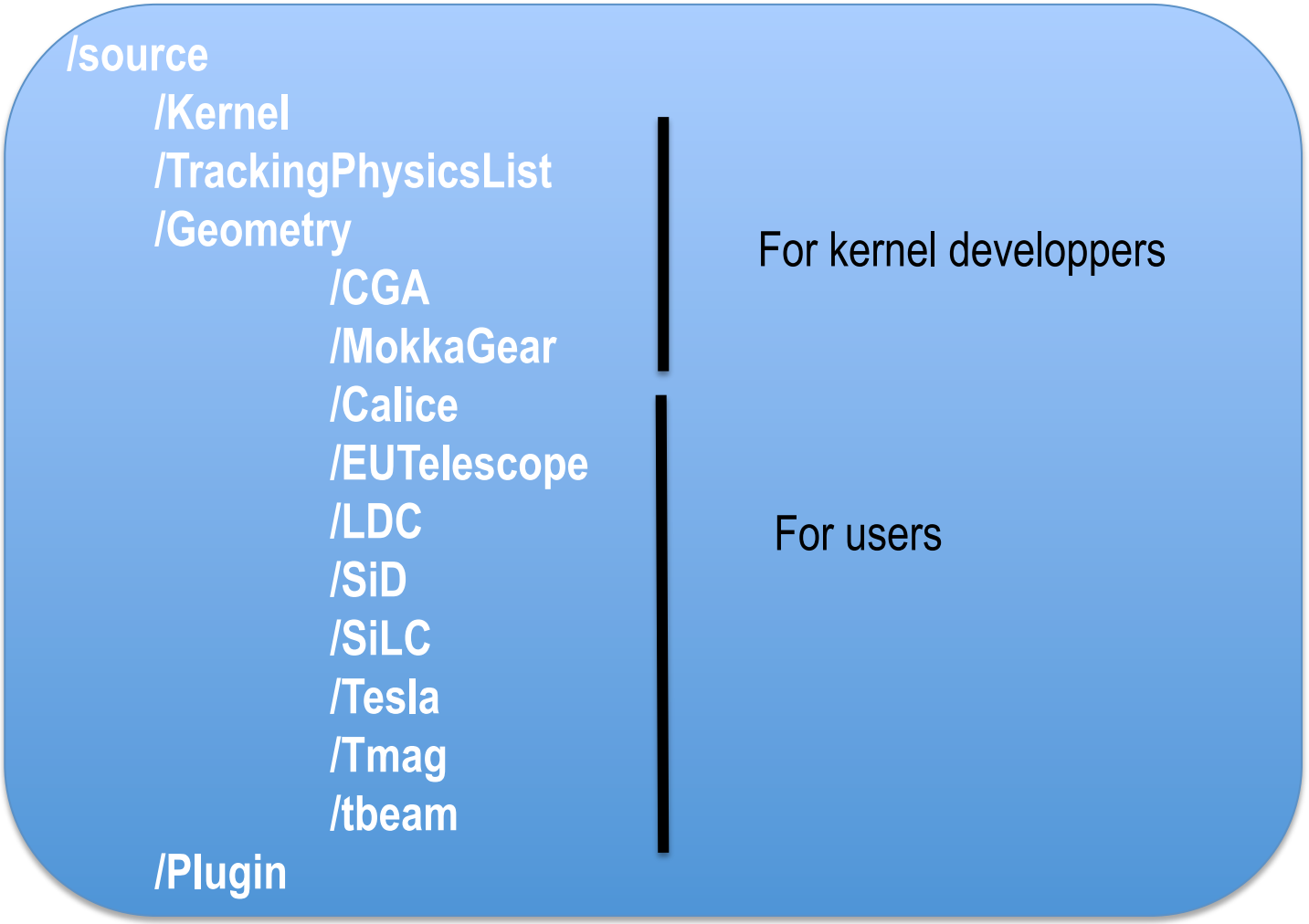


Mokka svn repository

- Svn repository code is organized in sub-directories (packages) following Mokka software architecture
- Geometry drivers for ILD are Geometry/LDC and Geometry/Tesla packages
- Each driver corresponding to one sub-detector is maintained by a specific working group
- Currenly there are no new developments in the kernel part of the code, the activities are focussed on geometry drivers and on test beam simulation

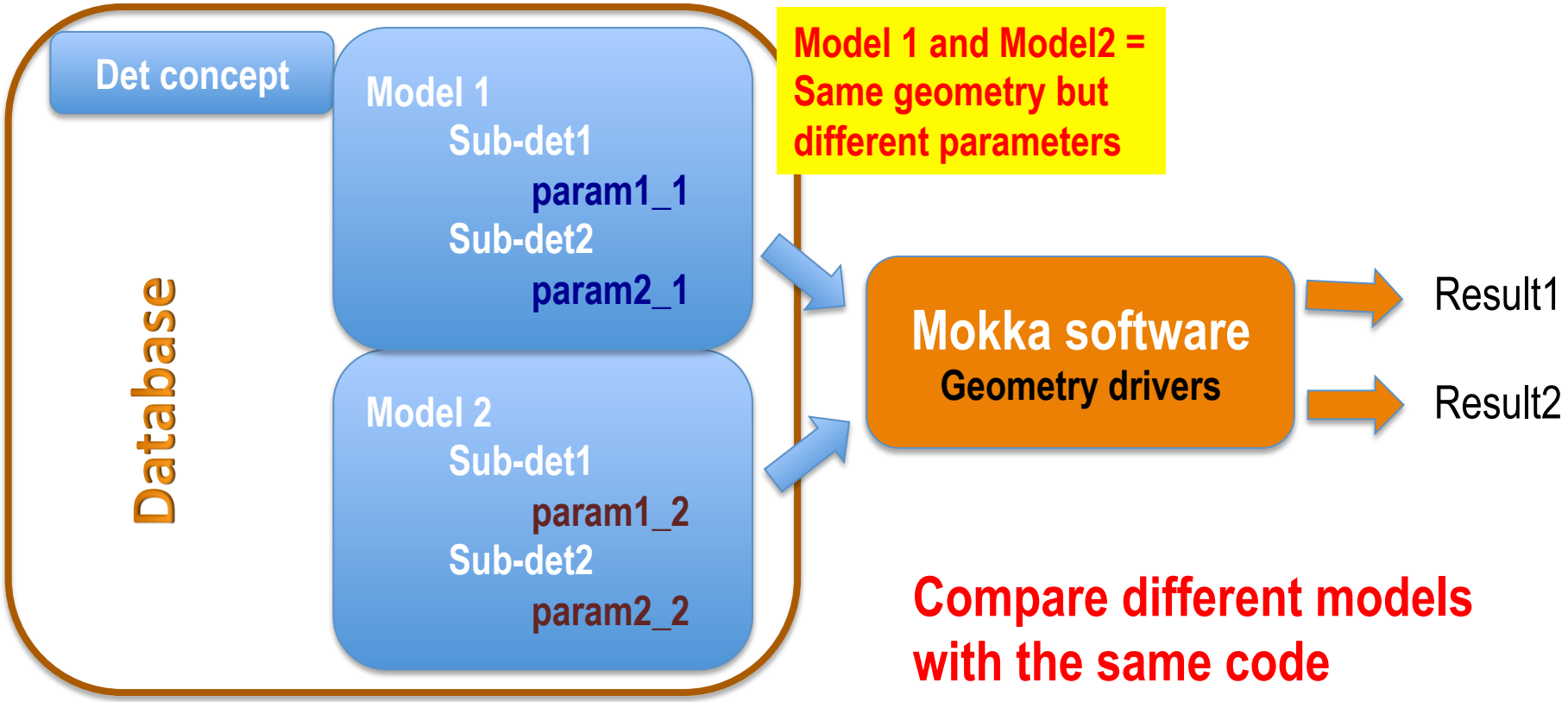
Mokka svn repository architecture

https://llrforge.in2p3.fr/viewvc/Mokka/trunk/



Motivations for MySQL database (1)

- Keep geometry values outside the code
- Compare detector performances independently of the software



Motivations for MySQL database (2)

- To have a generic user interface
- Structured Query Language (SQL) consists of data definition and data manipulation language

- Several GUI and API are available for MySQL

Technical information about mokka db

- Mokka databases are hosted at LLR server **llrmokka.in2p3.fr** : alias **pollin1.in2p3.fr** points to llrmokka

- Somes caracteristics: Hardware: VM KVM, Scientific Linux 6.5, MySql 5.1.73

- The server is backup on independent hardware
- Access to the databases :
  - ✧ user login: consult

This user has rights to only read the data.

Global Mokka databases
General presentation

Mokka databses include several mysql databases

- Databases for subdetectors:

vxd, tpc, yoke, mask, coil, services…,

models01, models03 (for subdetectors ecal, hcal,)

material01, materials02

With the recent ILD models we use models03 and materials02dbs

- Databases for Calice prototype simulation
- Databases for tests beam
- Temporary databasesTMP_..

Explore the database

Mokka Geometry Database Cross-Reference by Adrian Vogels

http://www-flc.desy.de/ldcoptimization/tools/mokkaxref.php

Detector, Concept, Model name, Subdetectors, C++drivers, Databases, Parameters

Mokka Geometry Database Cross-Reference

Jump to: Detector Concepts · Detector Models · Subdetectors · C++ Drivers · MySQL Databases · Geometry Parameters

Detector Concept "Biomed"

Description	first simulator for biomed
World Box	10000 × 10000 × 10000 mm³ (octant)
Models	Phantom01

Detector Concept "CHIC"

Description	The CHIC detector concept
World Box	9000 × 9000 × 14000 mm³ (octant)
Models	CHIC_00

Detector Concept "CILD"

Description	The ILDish CLIC detector concept
World Box	7500 × 7500 × 14000 mm³ (octant)
Models	CILD_00, CILD_00fw, CLIC01_ILD, CLIC_ILD_CDR, CLIC_ILD_CDR500

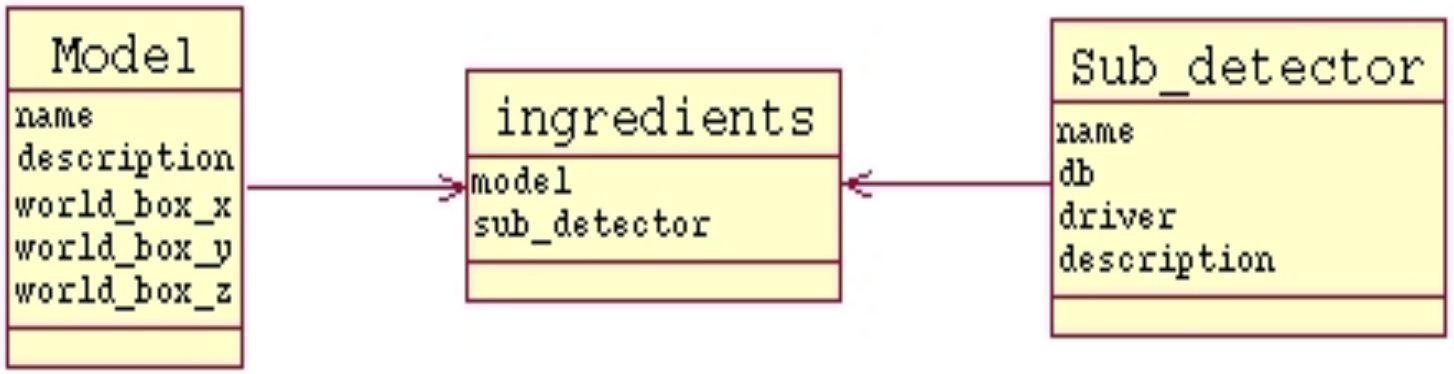
Detector Concept "ILD"

Description	The ILD detector concept
World Box	9000 × 9000 × 14000 mm³ (octant)
Models	ILD_00, ILD_00_EcalSc01, ILD_00_EcalSc02, ILD_00Dhcal, ILD_00fw, ILD_00fw_Dhcal, ILD_00fwp01, ILD_01_dev, ILD_01_SciW_pre00, ILD_01_SDH_pre00, ILD_01pre00, ILD_01pre01, ILD_01pre01fw, ILD_01pre02, ILD_01_v01, ILD_01_v02, ILD_01_v03, ILD_01_v04, ILD_o1_v05, ILD_o1_v06, ILD_02_v01, ILD_02_v02, ILD_02_v03, ILD_02_v04, ILD_o2_v05, ILD_o2_v06, ILD_03_v01, ILD_03_v02, ILD_03_v03, ILD_03_v04, ILD_o3_v05, ILD_o3_v06

Describing detector geometry by models

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

Database models03



Examples of detector models in Mokka db

Svn package => Concept name

Concepts name

name
CILD
ILD
LDC
LDC Extended
SiD
Tesla TDR
Tesla TDR 2
Tesla TDR Extended
Test Beam Calice
Test Beam Cern

Concept : ILD

Model name
ILD_O1_v03
ILD_O2_v03
ILD_O3_v03
ILD_O3_v04
ILD_O2_v04
ILD_O1_v04
ILD_o3_v05
ILD_o2_v05
ILD_o1_v05
ILD_o1_v06
ILD_o3_v06
ILD_o2_v06
……

Concept : SiD

Model name
SiD01
SiD02

Concept Test Beam Calice

name
TB00
TB01
TB02
CaliceEcal03
CaliceEcalHcalRPC
TB03
TB03_hcalcatch
ProtoDesy0205
TB04_hcal
TB04_hcalcatch
TB04
TB05
TBDesy0205
TB06

The geometry in DB

VOID=models03

concept name
ILD

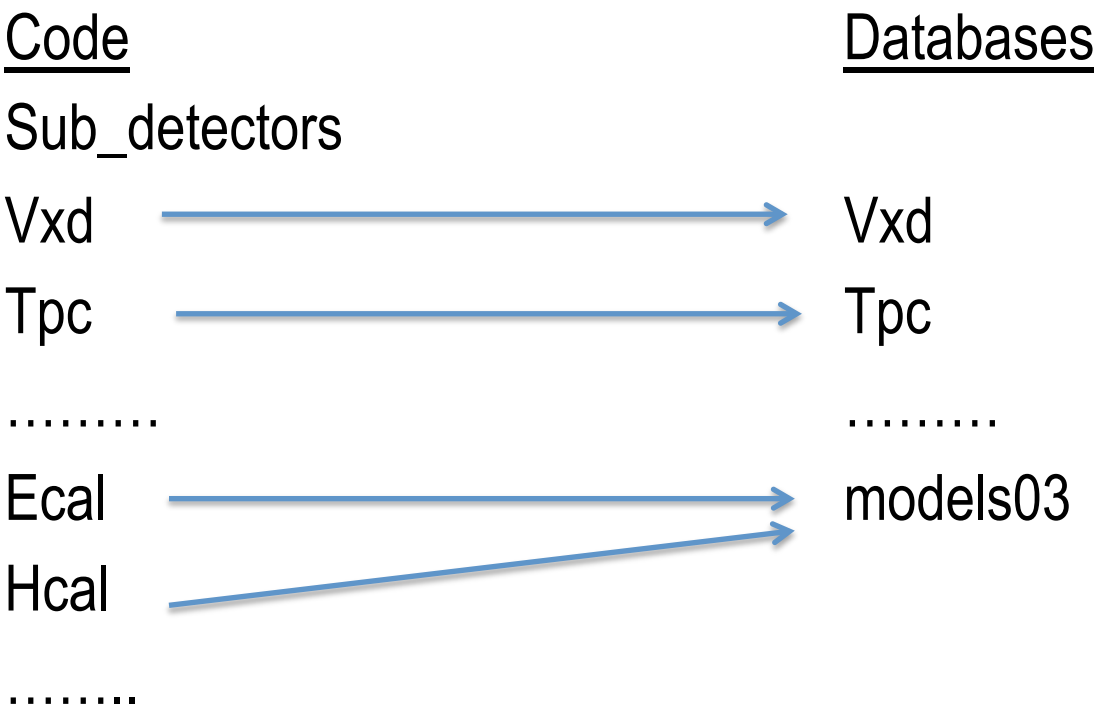
Table ingredients

model	sub_detector
ILD_o2_v06	LHcal01
ILD_o2_v06	tpc10_01
ILD_o2_v06	ftd_simple_....
ILD_o2_v06	SEcal05
ILD_o2_v06	SHcalRpc01
ILD_o2_v06	SCoil03
ILD_o2_v06	yoke05
ILD_o2_v06	LumiCalV
ILD_o2_v06	tubeX06
ILD_o2_v06	sit_simple_p.....
ILD_o2_v06	SField01
ILD_o2_v06	vxd07
ILD_o2_v06	set_simple.....
ILD_o2_v06	maskX03
ILD_o2_v06	BeamCal08
ILD_o2_v06	SServices_O2_v00

Table sub_detector

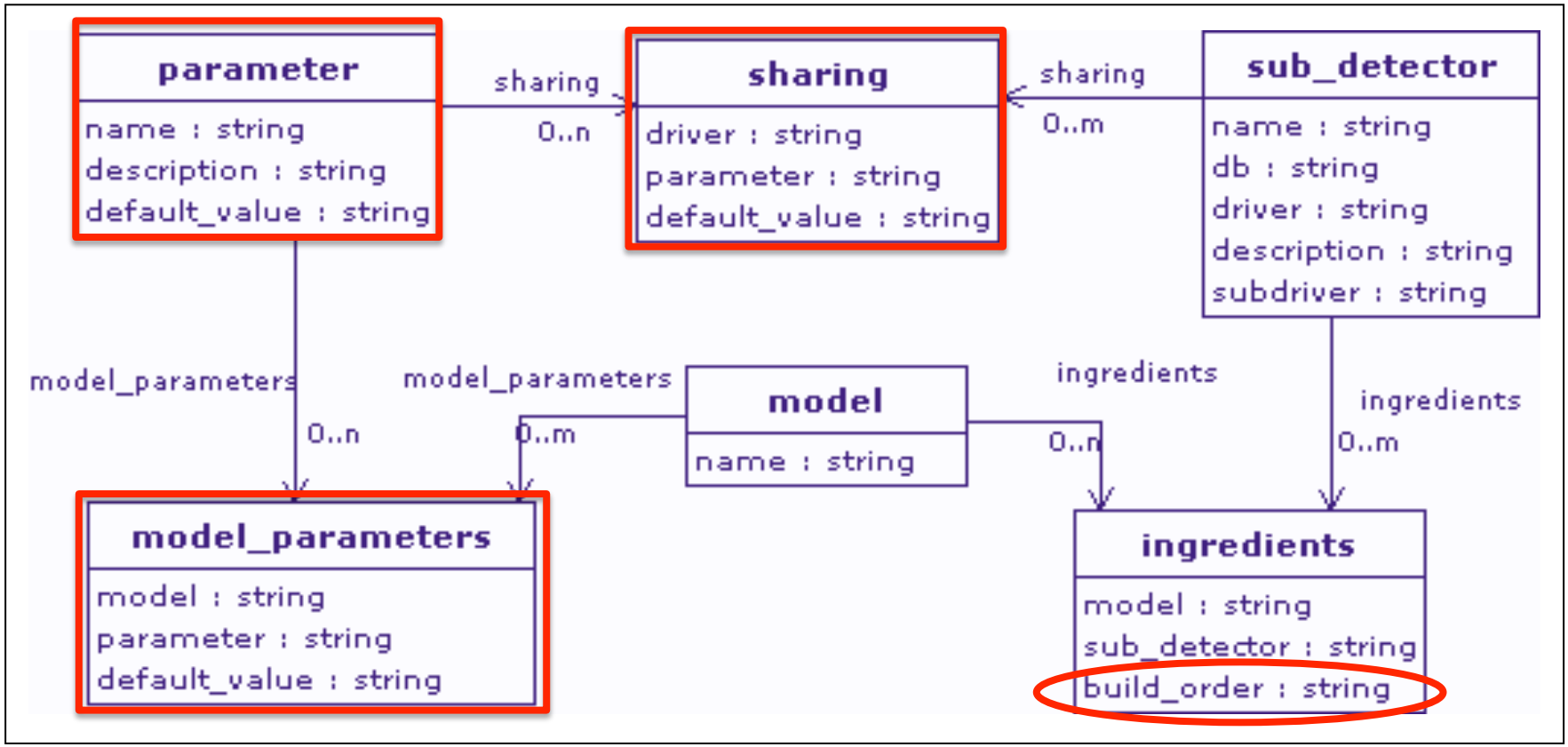
name	db	driver
LHcal01	VOID	SLHcal01
tpc10_01	tpc10_01	tpc10
ftd_simple_st..	ftd_simple_...	FTD_Si.....
SEcal05	VOID	SEcal05
SHcalRpc01	VOID	SHcalRpc01
SCoil03	VOID	SCoil02
yoke05	yoke04	yoke05
LumiCalV	VOID	SLcal03
tubeX06	tubeX06	tubeX01
sit_simple_pl...	sit_simple_..	SIT_Simp..
SField01	VOID	SField01
vxd07	vxd07	SVxd04
set_simple.....	set_simple_...	SET_Si..
maskX03	maskX03	maskX01
BeamCal08	beamcalX08	Beam....
SServices_O2_v00	Sservices_..	Sservices...

Short overview



Let's check this in the db browser

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

Model « global » parameters

model	parameter
ILD_o2_v06	Coil_extra_size
ILD_o2_v06	Coil_Yoke_radial_clearance
ILD_o2_v06	**Ecal_Barrel_halfZ**
ILD_o2_v06	**Ecal_endcap_extra_size**
ILD_o2_v06	**Ecal_support_thickness**
ILD_o2_v06	**Ecal_Tpc_gap**
ILD_o2_v06	Field_nominal_value
ILD_o2_v06	Hcal_back_plate_thickness
ILD_o2_v06	Hcal_Coil_additional_gap
ILD_o2_v06	Hcal_Ecal_gap
ILD_o2_v06	Hcal_endcap_center_box_size
ILD_o2_v06	Hcal_endcap_ecal_gap
ILD_o2_v06	Hcal_endcap_sensitive_...
ILD_o2_v06	Hcal_endcap_zmin
ILD_o2_v06	Hcal_nlayers
ILD_o2_v06	Hcal_sensitive_model
ILD_o2_v06	ILC_Main_Crossing_Angle
ILD_o2_v06	Lcal_inner_radius

model	parameter
ILD_o2_v06	Lcal_outer_radius
ILD_o2_v06	Lcal_z_begin
ILD_o2_v06	Lcal_z_thickness
ILD_o2_v06	**TPC_inner_radius**
ILD_o2_v06	**TPC_outer_radius**
ILD_o2_v06	TUBE_crossing_angle
ILD_o2_v06	TUBE_opening_angle
ILD_o2_v06	VXD_active_silicon_thickness
ILD_o2_v06	**VXD_inner_radius**
ILD_o2_v06	**VXD_length_r1**
ILD_o2_v06	**VXD_radius_r1**
ILD_o2_v06	**VXD_radius_r3**
ILD_o2_v06	**VXD_radius_r5**
ILD_o2_v06	**VXD_side_band_electronics_...**
ILD_o2_v06	**VXD_support_ladder_material**
ILD_o2_v06	**VXD_support_ladder_thickness**
ILD_o2_v06	Yoke_endcap_inner_radius
ILD_o2_v06	Yoke_thickness

Parameters

- All parameters needed for the geometry construction (including the global **model_parameters**) are in the table « parameters » and « sharing »
- Table « parameters »: contains default values
- Table « sharing »: the driver will override the default values from « parameters »

Parameters

mysql> select parameter, driver_default_value, default_value from sharing as s,parameters where parameter = name and driver='SEcal05';

parameter	"sharing" driver_default_value	"parameters" default_value
Ecal_cells_size	4.9	10.
Ecal_endcap_center_box_size	800.0	600.
Ecal_fiber_thickness	**0.15**	**0.2**
Ecal_Tpc_gap	**NULL**	**20.0**
Ecal_radiator_layers_set1_thickness	**NULL**	**2.1**
Ecal_radiator_layers_set2_thickness	NULL	4.2
Ecal_radiator_material	NULL	tungsten
Ecal_Si_thickness	**0.5**	**0.5**

……

Default value=NULL when: general parameter, should not be changed by a particular driver

Mokka geometry features

- **Modify the model** at launch time **(will be presented in details in the second subject for Mokka in this training)**

No matter the values storage in the database, the user can change the detector geometry just inserting a set of
/Mokka/init/globalModelParameter commands in the steering file.

- **Scaling** feature is **automatic** propagation of the changes made in one subdetector (via steering commands) to other subdetectors in order to avoid overlapping. The user does not have to care about related parameters.

Allows to study different detector options, TPC size, number of layers in calorimeters, etc. Since LDC00/LDC01 models the geometry is « scalable ».

Materials

The materials are registered into the default materials database
« materials02 »

To get access to these materials the "CGAGeometryManager::GetMaterial() » method should be used.

Some available materials are (table « materials »):

name	nistName	density
mylar	G4_MYLAR	NULL
tungsten	G4_W	NULL
silicon	G4_Si	NULL
lead	G4_Pb	NULL
kapton	G4_KAPTON	NULL
polystyrene	G4_POLYSTYRENE	NULL
epoxy		1.3

Materials

The materials that do not have **nistName** in the table « materials » are defined in the table « **components** »:

material	component	nAtoms	fraction
epoxy	H	44	0
epoxy	C	15	0
epoxy	O	7	0

You are free to define new materials or mixtures in the driver code (in geant4 manner).

But in order to keep the coherence in the software usage it's a good practice to use the already defined in the database, or to ask to add new ones in this list when needed.

Materials

Table **materials02**

```
+------------+-----------------------------------------+
| Field      | Type                                    |
+------------+-----------------------------------------+
| name       | varchar(80)                             | YES |
| nistName   | varchar(80)                             | YES |
| density    | double                                  | YES |
| temperature | double                                 |
| pressure   | double                                  |
| state      | enum('','undefined','solid','liquid','gas') |
```

Table **components**

```
+-----------+-------------+
| Field     | Type        |
+-----------+-------------+
| material  | varchar(80) |
| component | varchar(80) |
| nAtoms    | int(11)     |
| fraction  | double      |
+-----------+-------------+
```

Mokka Gear

Geometry **A**pi for **R**econstruction

- A special code implemented in drivers creates geometry xml file containing the current detector description at every startup of the Mokka application and used in reconstruction.
- Code for MokkaGear interface
- Dedicated code in each driver
- Gear xml file automathicaly created

```
<gear>
   <global detectorName="ILD_o2_v06" />
   <!--Gear XML file automatically created with GearXML::createXMLFile ....-->
   <BField type="ConstantBField" x="0.000000000e+00" y="0.000000000e+00"
z="3.500000000e+00" />
   <detectors>
……..
<detector name="EcalBarrel" geartype="CalorimeterParameters">
          <layout type="Barrel" symmetry="8" phi0="0.000000000e+00" />
          <dimensions inner_r="1.847415655e+03" outer_z="2.350000000e+03" />
          <layer repeat="19" thickness="5.250000000e+00” absorberThickness="2.100000000e+00"
cellSize0="5.083333333e+00"    cellSize1="5.083333333e+00" />
          <layer repeat="1" thickness="6.300000000e+00” absorberThickness="2.100000000e+00"
cellSize0="5.083333333e+00"    cellSilayer repeat="9" thickness="7.350000000e+00”
absorberThickness="4.200000000e+00" cellSize0="5.083333333e+00"
       cellSize1="5.083333333e+00ze1="5.083333333e+00" />
          <" />
       </detector>
</detectors>
…..
</gear>
```

Mokka and the grid

- Mokka simulations are run successufully on the grid for detector optimisation studies and for ILD Monte Carlo data Mass Production

- Mokka is installed as a part of **ILCSOFT** on all grid sites supporting **Virtual Organization ILC/CALICE** hosted at **DESY**

- For more information about how to use **ILCSOFT** grid installations please see:

http://grid.desy.de/ilc/

http://ilcsoft.desy.de/portal/general_documentation/

- Leprince-Ringuet Laboratory is a part of the GRIF Tier-2, a distributed Grid site in the Paris region, and contributes to the total GRIF resources (≈ 9k computing cores and 5PB of storage) with 1500 computing cores and 1PB of storage.

GRIF (and **LLR** in particular) supports the activity of the **ILC and Calice virtual organization** providing computing power and temporary storage.

Contact: Andrea Sartirana « sartiran@llr.in2p3.fr »

Getting started with Mokka

How to access the code (1)

- Svn repository is at Leprince-Ringuet Laboratory's server:
  llrforge.in2p3.fr

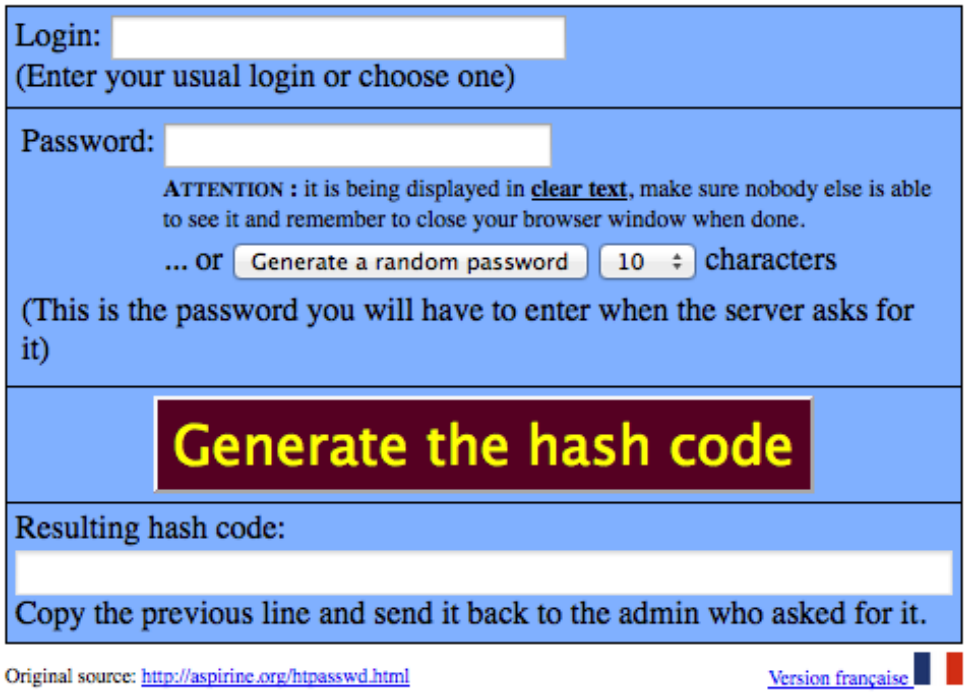
Code browser: https://llrforge.in2p3.fr/viewvc/Mokka

- Ask for access

Connect to https://llrforge.in2p3.fr/pwhash/pwhash_en.html

1) Chose a login
2) Chose a password
3) Scramble a password generate secure, hard-to-guess passwords
4) Send the resulting hash code to emilia.becheva (at) llr.in2p3.fr

How to access the code (2)



Installing Mokka

- Installing ilcsoft: Mokka is available by default

For example in v01-17-04 ilcsoft version the Mokka tag mokka-08-03 is installed.

- For new development or tests, install Mokka tag in your working directory

$ svn co http://llrforge.in2p3.fr/svn/Mokka/tags/mokka-08-03

$ cd mokka-08-03

$ cp $ILCSOFT/init_ilcsoft.sh .

Change the path for Mokka

$ source init_ilcsoft.sh
$ mkdir build
$ cd build

$ cmake -C $ILCSOFT/ILCSoft.cmake ..

$ make install

Running Mokka

- Install environment variables by launching a init_ilcsoft.sh file

Put paths to Geant4, CLHEP, Mokka

- Configure a steering file mokka.steer

  mokka.steer is an Geant4 like ASCII file which contains UI commands

- Mokka [options] mokka.steer

$./bin/Mokka mokka.steer

- Run in batch mode

Put in the steering file

/Mokka/init/BatchMode true

Getting help with Mokka code (1)

- Official Mokka web pages

http://mokka.in2p3.fr/

- Basic information for geometry driver developers

https://llrforge.in2p3.fr/viewvc/Mokka/trunk/doc/Kernel/geometry_drivers.html

- Mokka Common Geometry Access (CGA) Application Programming Interface

https://llrforge.in2p3.fr/viewvc/Mokka/trunk/doc/CGADoc/CGAIndex.html

- Check releases notes

https://llrforge.in2p3.fr/viewvc/Mokka/trunk/ReleaseNotes/

- How to use on line mokka with options

$ Mokka –H

Exploring Mokka dbs

- The Mokka Detector Model Database (Web) Browser by Adrian Vogels

http://www-flc.desy.de/ldcoptimization/tools/mokkamodels.php

Very helpful for finding quickly a particular information

- Mokka Geometry Database Cross-Reference by Adrian Vogels

http://www-flc.desy.de/ldcoptimization/tools/mokkaxref.php#model

- Command line interface (or browsers) to Mokka database

Very useful when you need to understand the relashionship between data

Getting help with Mokka code and Mokka dbs (2)

- Ask for help to Mokka users

ild-software-discussion@desy.de

For subscribing send an email to Frank Gaede <frank.gaede@desy.de>

Conclusion

- Mokka is a full simulation framework for the **International Linear Collider detectors** based on Geant4
- Uses MySQL DB + C++ geometry drivers
- Flexible and detailed geometry of ILD

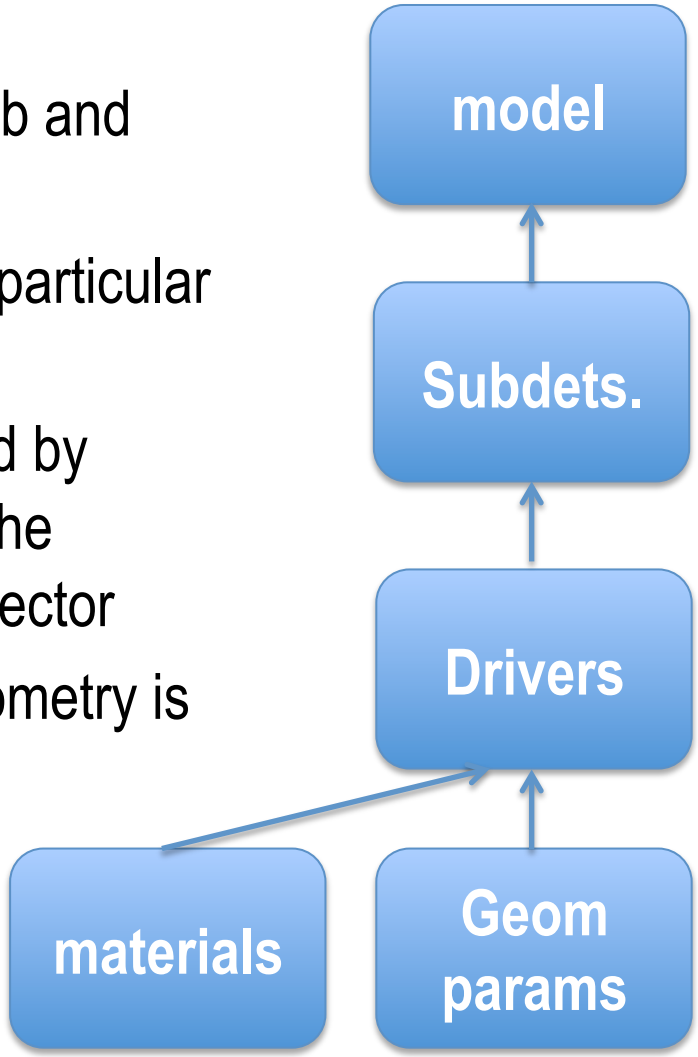
Si Vertex Detector, Ecal-Electromagnetic Detector HCAL, TPC/TPC endplate and electronics FTD- Forward tracking Disks ETC-Endcap Tracking Detector,

SET-External Tracking Detector

- Mokka is also used for testbeam prototypes simulation: Calice, EUTelesescope: pixel telescope for silicon tracking

Reminder of the main points

- Mokka main() reads the model from the db and construct the detectors
- The model is a set of sub_detectors with particular parameters values
- The subdetectors geometry is constructed by drivers using the parameters stored into the database, there is one driver per sub_detector
- All the information for the ILD models geometry is in the « models03 » mysql database
- The materials are stored in the mysql database « materials02 »



References

Presentations: Paulo Mora de Freitas, Gabriel Musat, Henry Videau

- **Workshop on Geometry Toolkit for the Linear Collider**, 24 February 2010 – CERN
- **AIDA kick-off meeting**, 17 Feb 2011 – CERN, P. Mora de Freitas, G. Musat, LLR-Ecole polytechnique

- **Marlin et al – A Software Framework for ILC detector R&D**, Dec 17, 2007, F. Faede, J. Engels


cnrs
advancing the frontiers


ÉCOLE
POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

MOKKA


DESY


東京大学
THE UNIVERSITY OF TOKYO


UNIVERSITY OF
CAMBRIDGE

Backup slides

Comparison

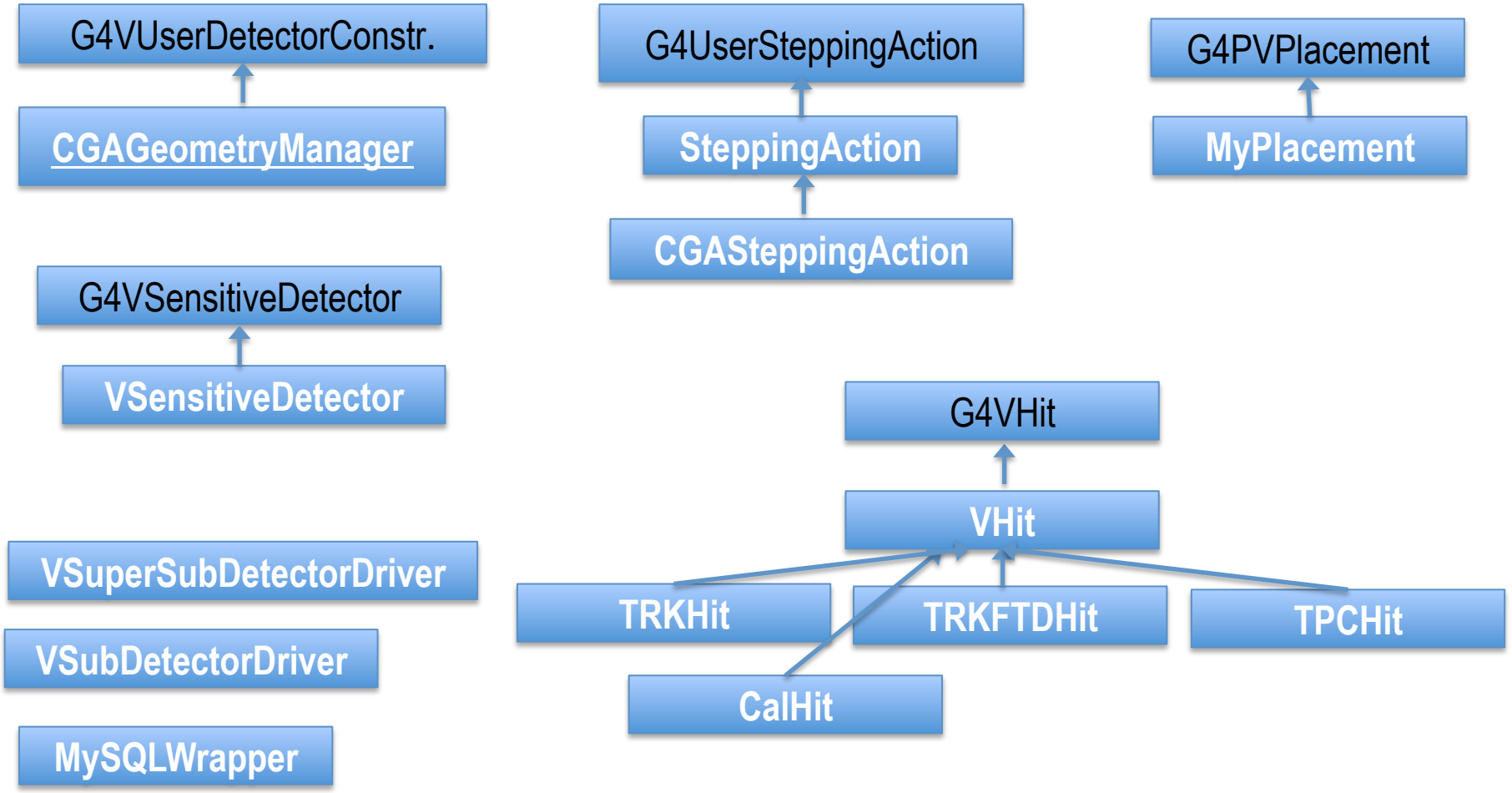
XML based scheme

- Self describing at some level
- Includes shape information (at some level)
- Tags sensitive detectors (but needs drivers to translate them into GEANT4)
- Not easy to switch between different detector / subdetector versions
- Scaling to complex detectors difficult to see without dedicated software to provide an interface to the user

MYSQL based scheme

- Vaguley self describing
- Does not include shape information (at the moment)
- Does not tag sensitive detectors
- Uses C++ drivers to translate into GEANT4 objects
- Easy to maintain and switch between different versions of the detector
- Scaling to complicated detector “easy”
- Supports every GEANT4 shape

Mokka Kernel Classes



Geometry/CGA

VSensitiveDetector.hh: public G4VSensitiveDetector
VSubDetectorDriver.hh
VSuperSubDetectorDriver.hh

MyPlacement.hh: public G4PVPlacement
MySQLWrapper.hh class DBResult And class Database

CalHit.hh: public VHit
CGADefs.h
CGAGeometryEnvironment.hh
CGAGeometryManager.hh: public G4VUserDetectorConstruction
CGASteppingAction.hh: public SteppingAction

PrimaryContribution.hh
// PDGContribution_type, maps the PDG in the E contribution for a given primary

TPCHit.hh: public VHit
TRKFTDHit.hh: public VHit
TRKHit.hh: public VHit
VHit.hh: public G4VHit

Mokka Kernel Classes 1

Particle generator :

VPrimaryGenerator.hh

GPSGenerator.hh: public VPrimaryGenerator

ParticleGunGenerator.hh: public VPrimaryGenerator

ParticleGunMessenger.hh: public G4UImessenger

Physique list :

DummyPhysicsList.hh: public G4VModularPhysicsList

PhysicsListFactory.hh

PhysicsListUserLimits.hh

ExtraParticles.hh: public G4VPhysicsConstructor

GeneralPhysics.hh: public G4VPhysicsConstructor

Mokka Kernel Classes 2

Actions :

EventAction.hh: public G4UserEventAction

PrimaryGeneratorAction.hh: public G4VUserPrimaryGeneratorAction

PrimaryGeneratorMessenger.hh: public G4UImessenger

RunAction.hh : public G4UserRunAction

RunManager.hh: public G4RunManager

StackingAction.hh: public G4UserStackingAction

SteppingAction.hh: public G4UserSteppingAction

SteppingActionMessenger.hh: public G4UImessenger

Tracking :

TrackSummary.hh

TrackingAction.hh : public G4UserTrackingAction

Trajectory.hh: public G4VTrajectory UserInit.hh

UserTrackInformation.hh: public G4VUserTrackInformation

Mokka Kernel Classes 3

Control :
Control.hh
ControlMessenger.hh: public G4UImessenger

Visualisation :
Visu.hh
VisuMessenger.hh: public G4UImessenger

Mokka Kernel Classes 4

GuineaPigInterface.hh: public G4VPrimaryGenerator

HepLCIOInterface.hh:public G4VPrimaryGenerator

/** Implementation of G4VPrimaryGenerator that reads in an StdHep file * and creates an LCIO MCParticle collection and uses that to * create the G4PrimaryParticles needed by geant4. * This results in a complete 'HepEvt' recors in the MCparticle collection.

HepLCIOInterfaceNew.hh: public G4VPrimaryGenerator/** Implementation of G4VPrimaryGenerator that reads in an StdHep file * and creates an LCIO MCParticle collection and uses that to * create the G4PrimaryParticles needed by geant4. * This results in a complete G4PrimaryParticle tree.

InputFileGenerator.hh: public VPrimaryGenerator

LCAscHepRdr.h

*/**Basic utility for reading a ASCII HEPEvt file and filling * a LCCollectionVec with MCParticles containing the HEPEvt * file information.*

TPCStepLimiterLowPt.hh: public G4VProcess // class description // // A "process" to be registered to the process manager of each particle, // in the UserPhysicsList, in order to take into account the MaxAllowedStep // defined by the steering parameter the G4UserLimits object attached to a logical volume.

IHEPEvtInterface.hh:public G4VPrimaryGenerator

IStdHep.hh: public IXDR IXDR.hh

Tables

```
| Tables_in_models03 |
+--------------------------+
| detector_concept     |
| ingredients          |
| model                |
| model_parameters     |
| parameters           |
| scripts              |
| setup                |
| setup_parameters     |
| sharing              |
| sub_detector         |
| tmp_databases        |
+--------------------------+
```

Ingredients

```
+-----------------+------------+
| Field           | Type       |
+-----------------+------------+
| id              | bigint(4)  |
| model           | char(80)   |
| sub_detector    | char(80)   |
| build_order     | bigint(4)  |
+-----------------+------------+-
```

model

```
+----------------------+------------------------------+
| Field                | Type                         |
+----------------------+------------------------------+
| name                 | varchar(80)                  |
| description          | varchar(255)                 |
| detector_concept     | varchar(100)                 |
| model_status         | enum('unstable','frozen')    |
+----------------------+------------------------------+
```

model_parameters

Field	Type
model	varchar(80)
parameter	varchar(80)
default_value	varchar(80)

parameters

Field	Type
name	varchar(100)
description	varchar(200)
default_value	varchar(100)

setup_parameters

Field	Type
setup	varchar(100)
parameter	varchar(100)
value	varchar(100)

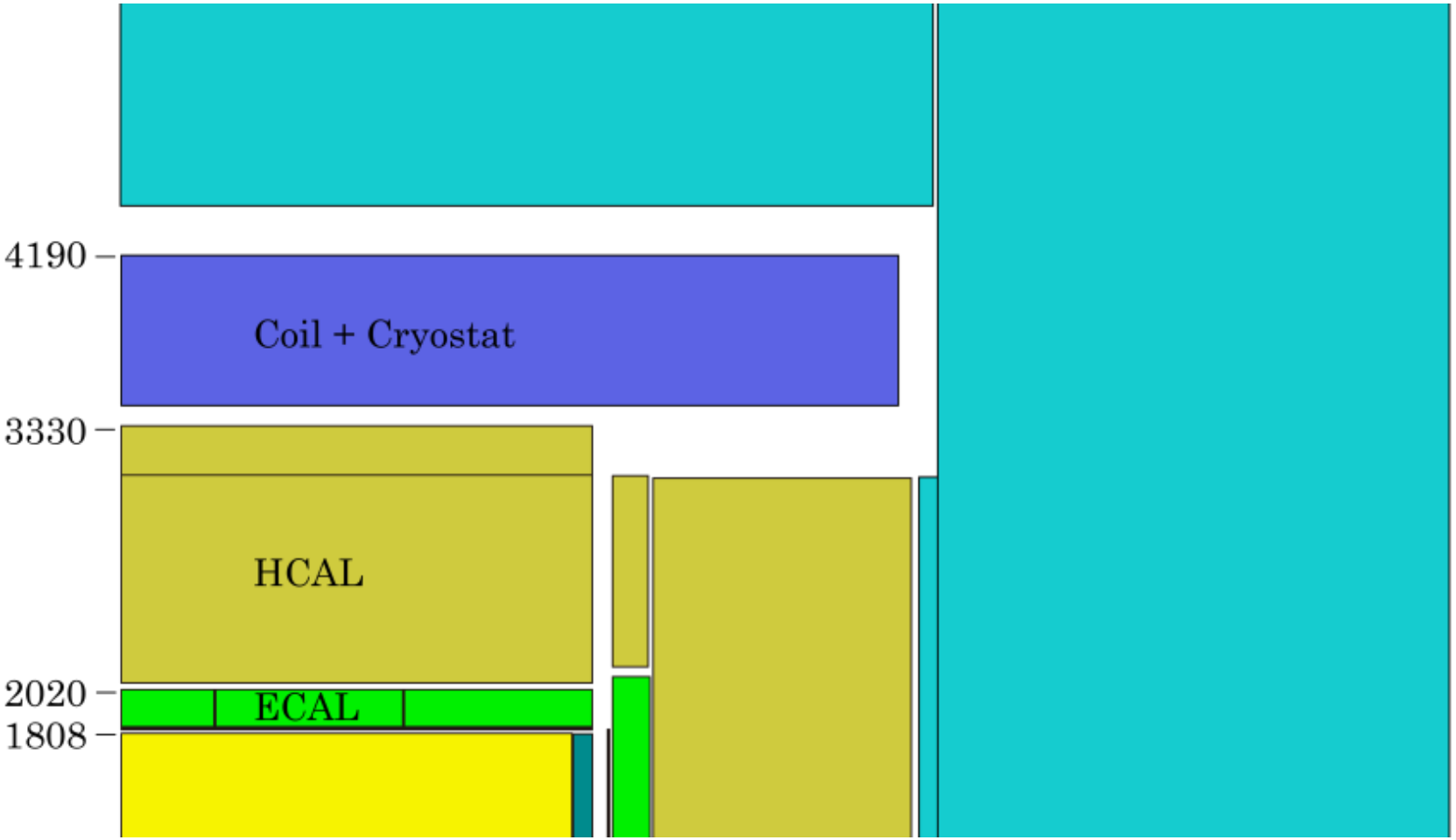
Sub_detector

Field	Type
id	int(4)
name	varchar(80)
db	varchar(80)
driver	varchar(20)
description	varchar(255)
subdriver	varchar(100)

sharing

Field	Type
driver	varchar(100)
parameter	varchar(100)
driver_default_value	varchar(100)

ILD_00



Quadrant view of the ILD detector model.
The interaction point is in the lower left corner.