Status of new ILD software tools

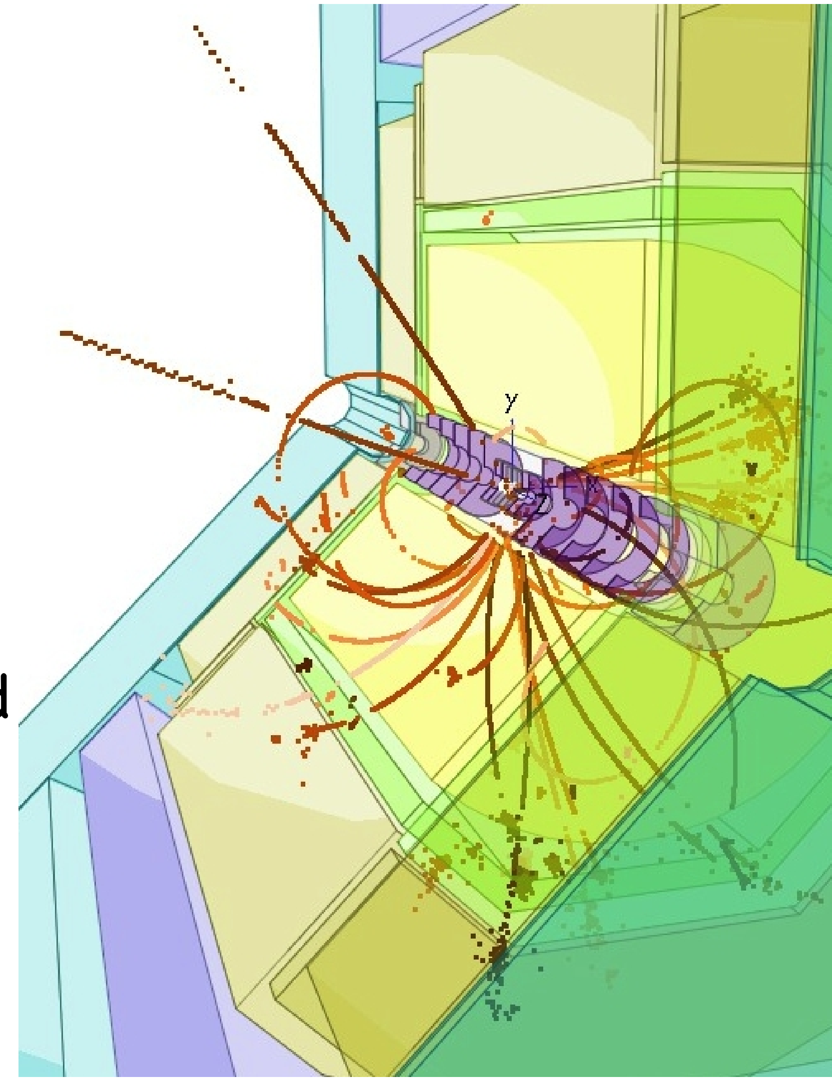
Frank Gaede, CERN/DESY

ILD Optimization Meeting

April, 8 2015

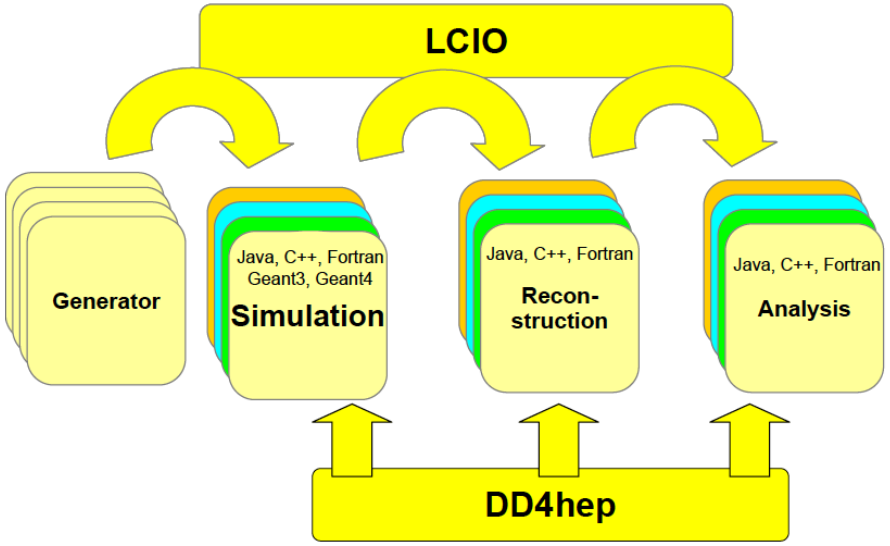
Outline

- Introduction
- Simulation
  - DDG4, lcgeo,...
- Reconstruction
  - DDRec, DDKalTest,...
- Towards new simulation models and a timeline
- Summary & Outlook



Introduction

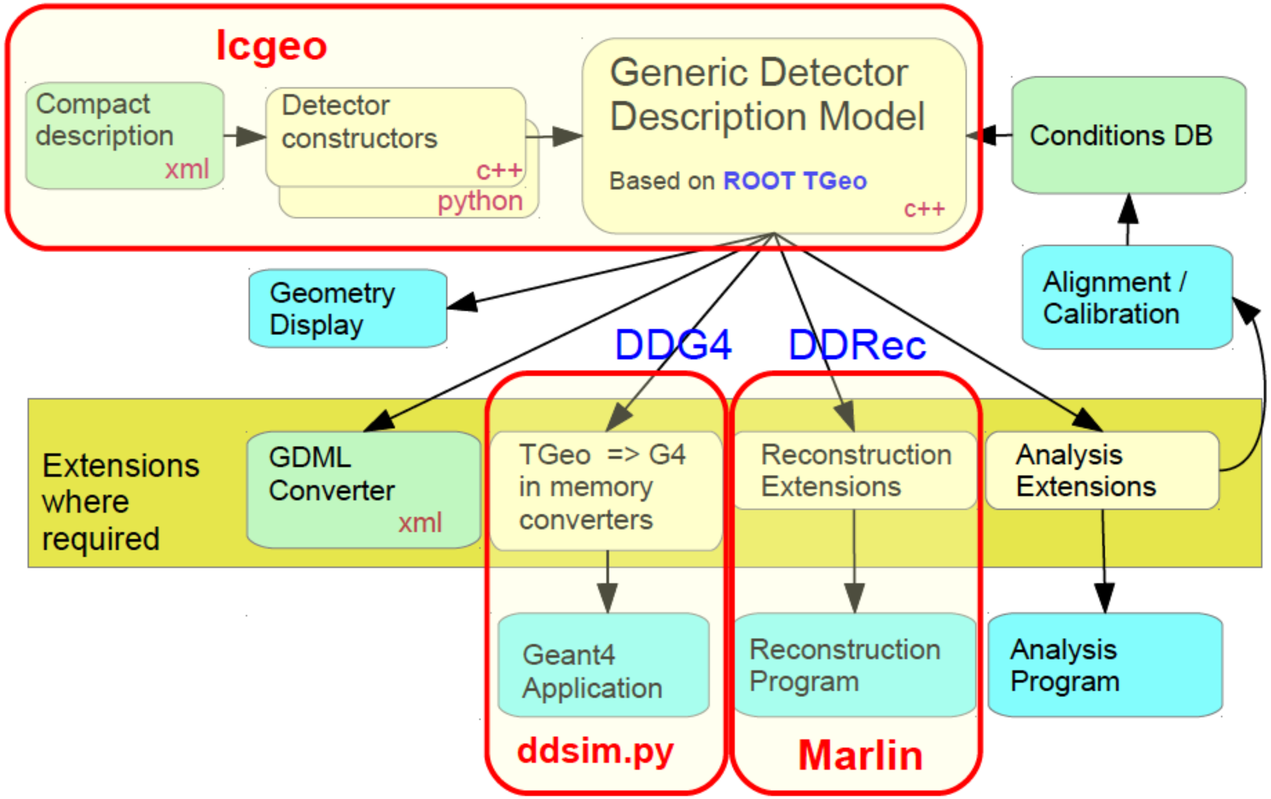
- in Linear Collider Software Meetings 2012/2013 decided to use new detector geometry description **DD4hep** as basis for a new common LC simulation package



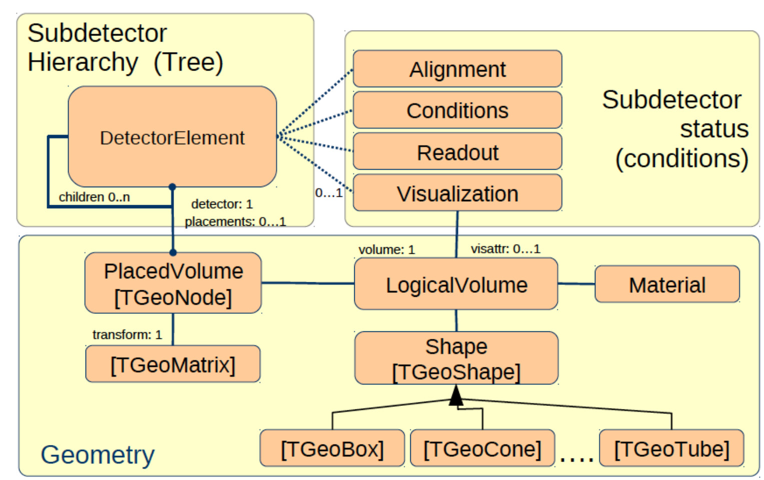
- at last ILD meeting at the LCWS in Belgrade we have decided to use DD4hep based simulation models for the next round of ILD detector optimization

- defining a common geometry API is the second step – after the common EDM (LCIO) – that is needed to have an open and modular software framework

DD4hep - overview



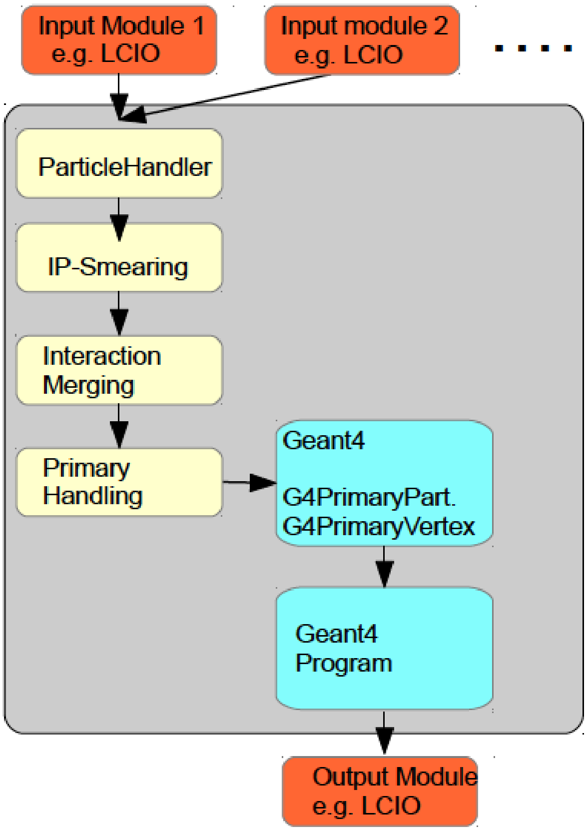
- consistent description of detector geometry from one unique source
  - for complete life cycle of experiment
- detailed (simulation) geometry model - extended with user defined data for reconstruction/analysis
- implemented in ROOT-TGeo - interface to Geant4



Simulation

DDG4: DD4hep-Geant4 gateway

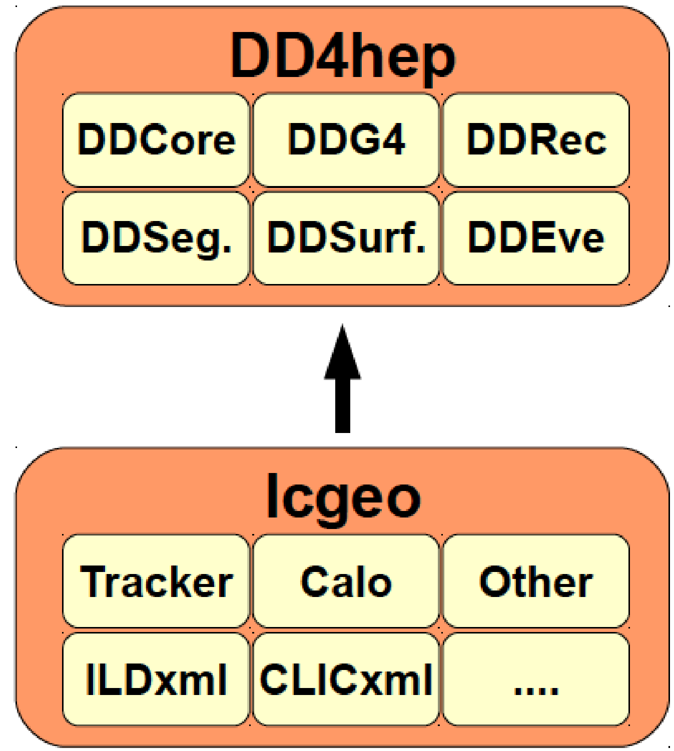
- in memory conversion of TGeo geometry to Geant4 geometry
- modular design using plugin mechanism for
  - sensitive detectors, Geant4 user actions : stepping, tracking, handling of MCTruth link for hits,...
  - input (LCIO, stdhep, HepMC,...) and output (LCIO,...)
- configure mechanism
  - xml, python or CINT:
  - physics lists, limits, fields,...
  - define sequences for
    - input, sensitive detectors, user actions, output,...



- supported by CERN for CLICdp, ILD and FCC

lcgeo: detector description

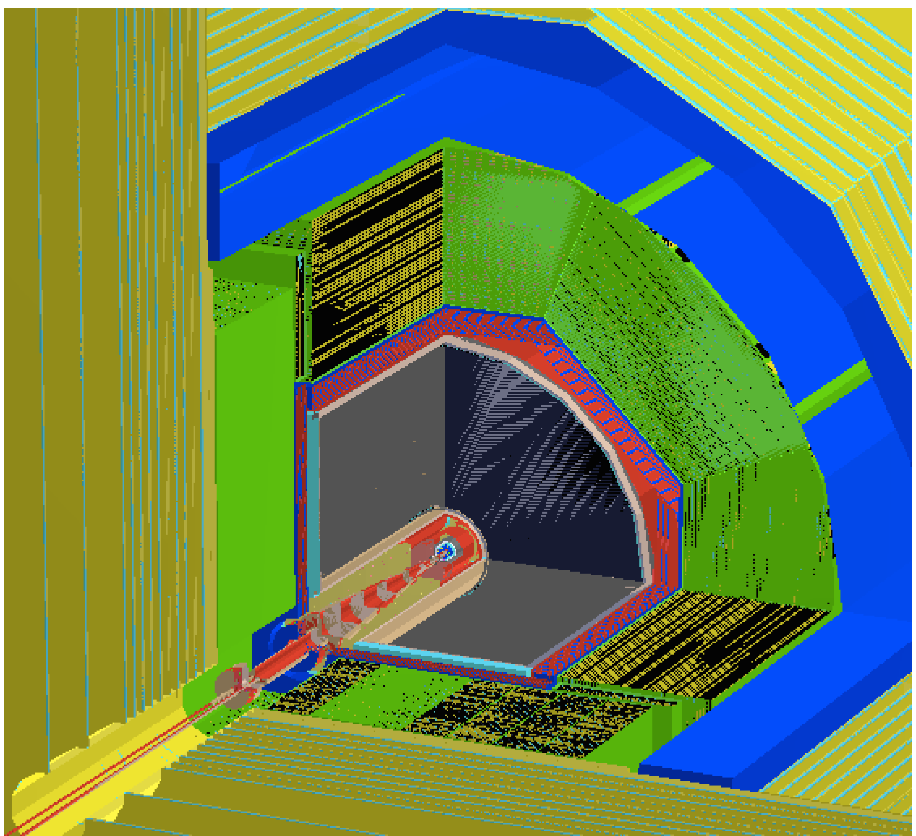
- created package **lcgeo** as a common LC detector description package for ILD and CLIC (and SiD) simulation and reconstruction
- simulation is (more or less automatically) provided via DDG4
- use simply python script for configuring and running the simulation: **ddsim.py**
  - (the actual Geant4 simulation application)
- where possible use the common (CLICdp/ILD) geometry drivers for sub-detectors, e.g. beamcal, ECal, Hcal,...
- for ILD: ported current Mokka model ILD_o1_v05 to DD4hep
- started to modify and improve these

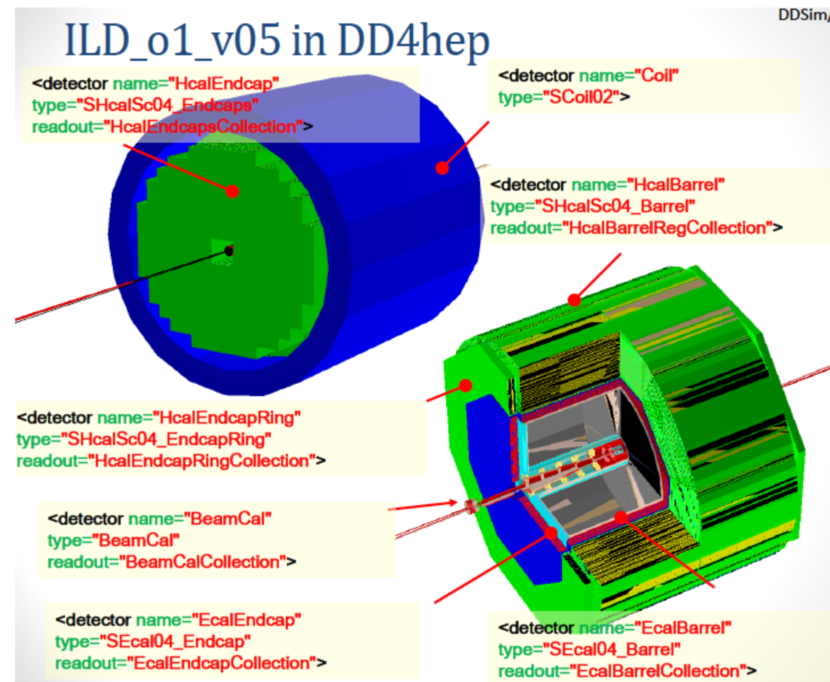


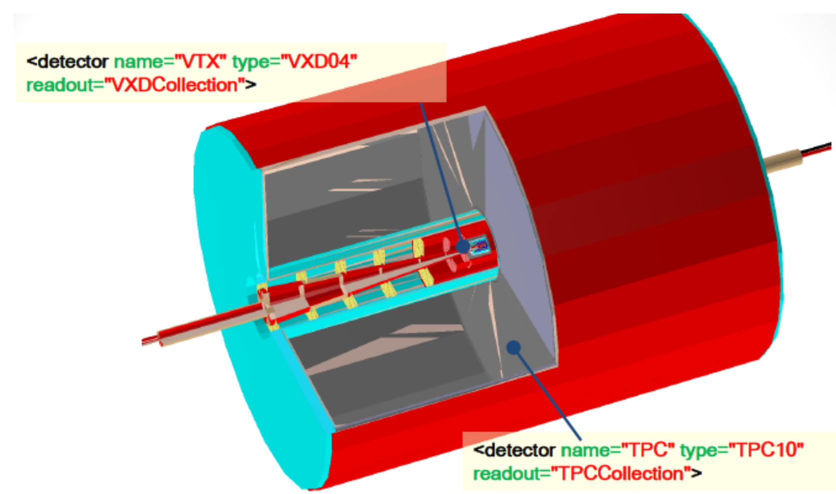
https://svnsrv.desy.de/viewvc/ddsim/lcgeo/trunk

ILD_o1_v05 in lcgeo

- complete Mokka model ILD_o1_v05 ported:
- VXD, FTD, SIT, TPC, SET, beam pipe
- Ecal, Hcal, Yoke (S.Lu)
- Beamcal, Lcal, LHcal (A.Sailer, M.Petric)
- so far using 'canonical' sensitive detectors

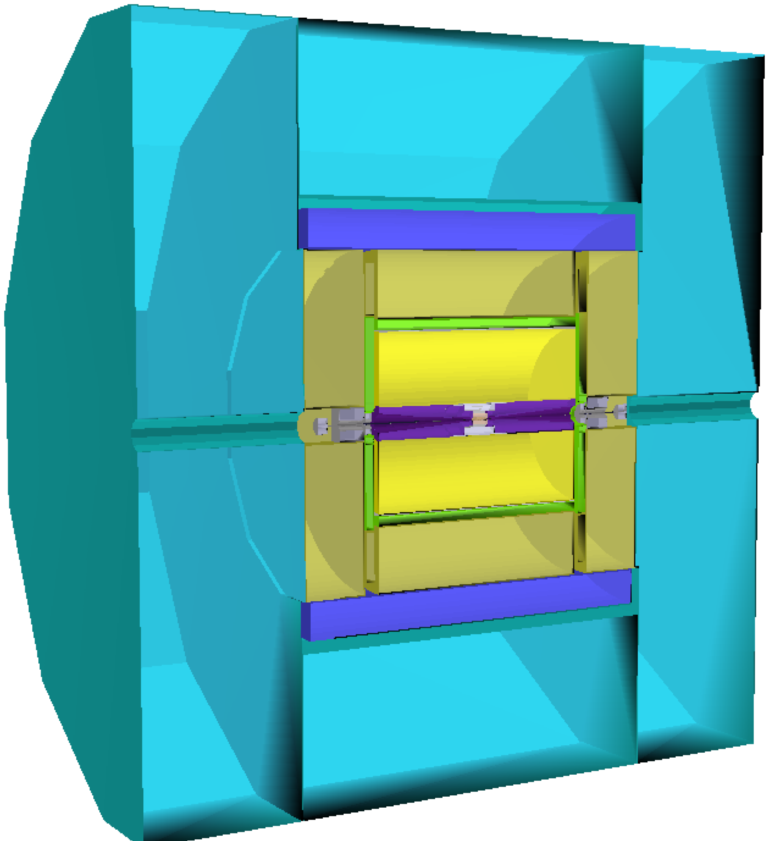






Envelopes in ILD simulation model I

- introduced 'mandatory' envelopes into the ILD simulation model, in order to



- speed up the simulation (navigation)
- have well defined 'real estate' for detectors
- synchronize more easily with CAD models (place holder volumes)
- facilitates development of new detector drivers and models
- eventually allow for some well defined scaling behavior

S.Lu, FG

xml:

```
<envelope vis="ILD_ECALVis">
  <shape type="PolyhedraRegular" numsides="Ecal_symmetry"  rmin="Ecal_inner_radius - env_safety"
         rmax="Ecal_outer_radius + env_safety" dz="2.*Ecal_half_length + env_safety"  material = "Air" />
  <rotation x="0*deg" y="0*deg" z="90*deg-180*deg/Ecal_symmetry"/>
</envelope>
```

code:

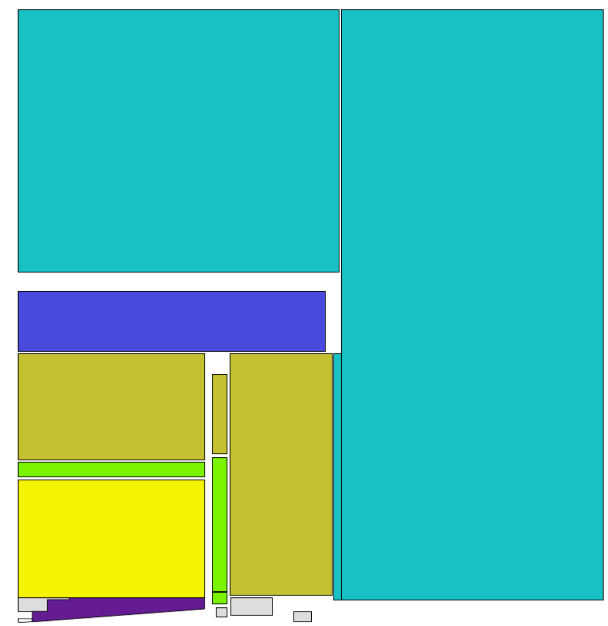
```
// --- create an envelope volume and position it into the world ---------------------

Volume envelope = create_placed_envelope( lcdd,  element , sdet ) ;

if( lcdd.buildType() == BUILD_ENVELOPE ) return sdet ;
//------------------------------------------------------------------------------------
```

Envelopes in ILD simulation model II

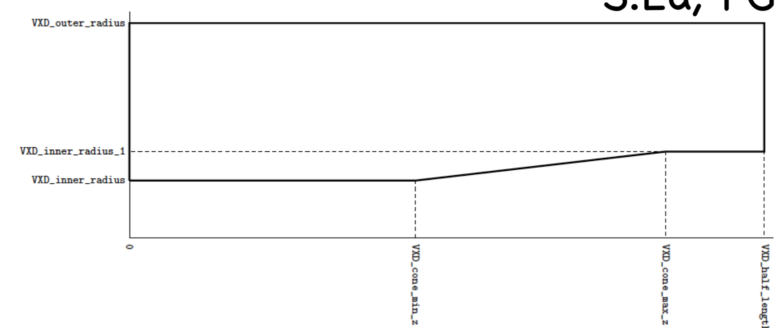
- started to compile 'canonical' set of envelope parameters
- document shapes and values (semi-automatically)

Envelope parameters for ILD_o1_v05

detector	inner radius	outer radius	half length min z, max z	additional parameters	
VXD	16.0	60.0	177.6	VXD_cone_min_z	80.0
				VXD_cone_max_z	150.0
				VXD_inner_radius_1	24.1
FTD	25.1	328.9	2350.0	FTD_outer_radius_1	152.8
				FTD_outer_radius_2	299.7
				FTD_min_z_0	177.7
				FTD_min_z_1	368.2
				FTD_min_z_2	644.2
				FTD_cone_min_z	230.0
				FTD_cone_radius	184.1
SIT	152.9	324.6	644.1	SIT_outer_radius_1	299.8
				SIT_half_length_1	368.1
TPC	329.0	1808.0	2350.0		
SET	1808.1	1827.9	2350.0		
Ecal	1843.0	2028.0	2350.0	Ecal_Hcal_symmetry	8.0
				Ecal_symmetry	8.0
EcalEndcap	400.0	2088.8	2450.0, 2635.0		
EcalEndcapRing	250.0	390.0	2450.0, 2635.0		
Hcal	2058.0	3395.5	2350.0	Hcal_inner_symmetry	8.0
HcalEndcap	350.0	3395.5	2670.7, 3957.7	EcalEndcap_symmetry	8.0
HcalEndcapRing	2138.8	3137.0	2450.0, 2635.0	HcalEndcapRing_symmetry	8.0
Coil	3425.0	4175.0	3872.0		
Yoke	4424.0	7725.0	4047.0	Yoke_symmetry	12.0
YokeEndcap	300.0	7725.0	4072.0, 7373.0	YokeEndcap_symmetry	12.0
YokeEndcapPlug	300.0	3395.5	3981.5, 4072.0	YokeEndcapPlug_symmetry	12.0
BeamCal	20.0	150.0	3475.0, 3695.0	BeamCal_thickness	220.0
				BeamCal_tubeIncoming_radius	15.0
LHCal	100.0	325.0	2680.0, 3200.0	LHCal_thickness	520.0
LumiCal	80.0	195.2	2500.0, 2630.7	LumiCal_thickness	130.7

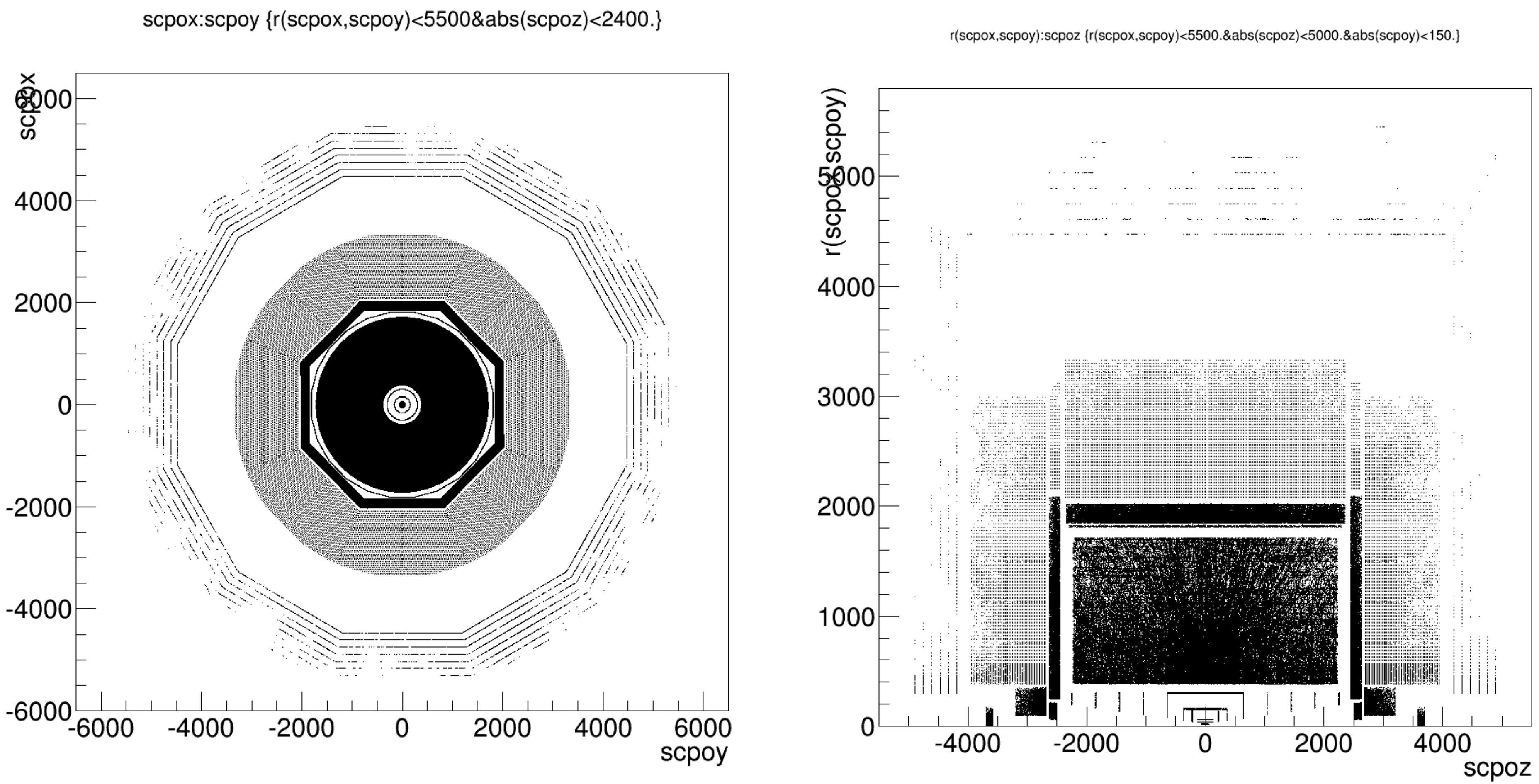


S.Lu, FG



to do: check values - introduce proper gap parameters ...

SimHits in lcgeo ILD_o1_v05



- can now fully simulate new ILD_o1_v05 model
- using canonical sensitive detectors (issue for TPC and Hcal, see next slide)

Status of subdetector drivers

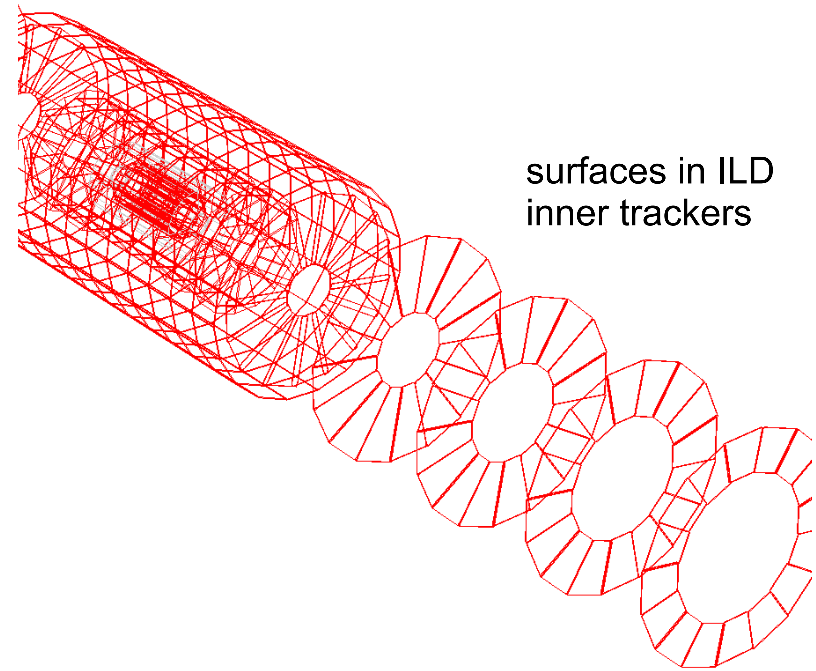
sub detector	description	comment
VXD	direct port from Mokka	need review
SIT, SET	direct port from Mokka	need review (still rather simplistic)
FTD	direct port from Mokka	need review
TPC	direct port from Mokka	need sensitive detector
Ecal	currently rewritten (D.Protopopescau)	would like to use common model with CLIC
Hcal	re-implemented (S.Lu)	need sens. detector w/ proper segmentation (tiling algorithm)
BeamCal, LumiCal, LHcal	re-implemented (A.Sailer, M Petric)	common w/ CLIC
Yoke	direct port from Mokka	
Beampipe	direct port from Mokka	common w/CLIC
Services	re-implemented (S.Lu)	ongoing

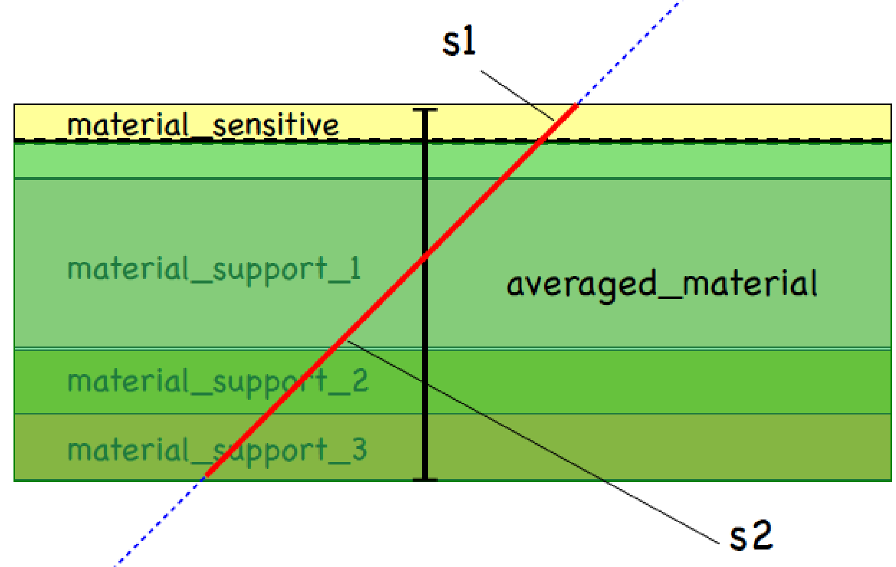
goal: have set of drivers that use a well defined set of parameters
and have a well defined scaling behavior

Reconstruction

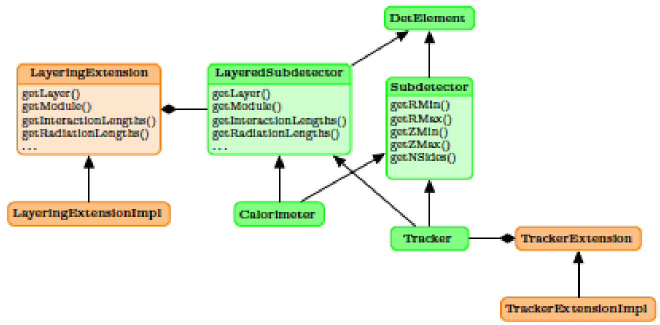
DDRec: surfaces for tracking

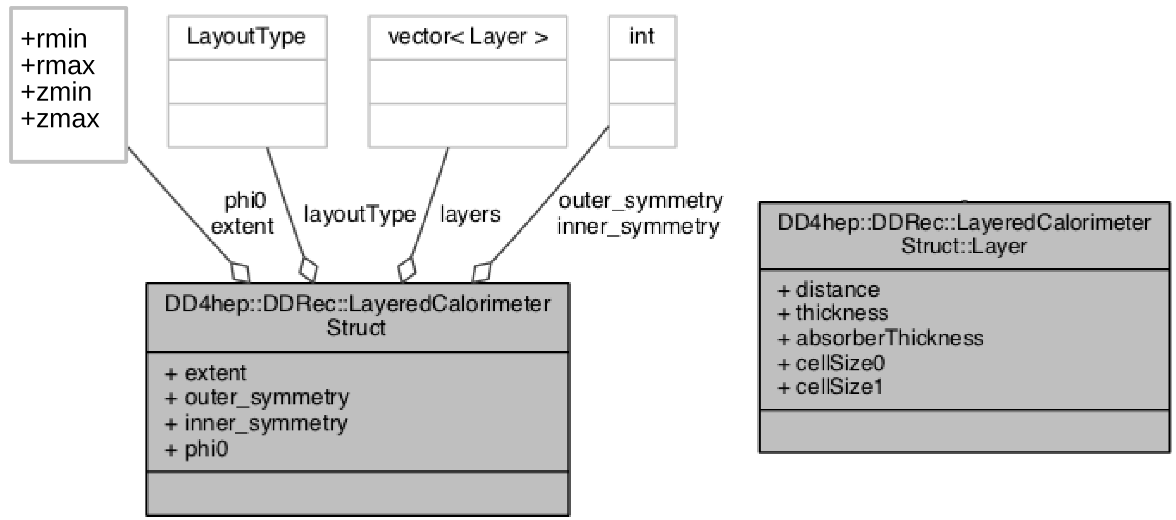
- tracking code needs a special interface to geometry:
- measurement and dead material surfaces (planar, cylindrical)
- surfaces are attached to volumes (defining boundaries) and provide:
  - u,v, normal, origin
  - inner and outer thicknesses and material
    - material is automatically averaged from detailed model
  - global to local and local to global coordinate transforms:
    - (u,v) ↔ (x,y,z)





DDRec: - interface for calorimeters





- two options:
- use extension mechanism to define the high level interface for the calorimeter reconstruction in a hierarchy of LayeredSubdetector classes
  - currently no longer pursued (prototype exists)
- attach simple data structure LayeredCalorimeter (similar to GEAR CalorimeterParameters) to describe the calorimeter information (as well as other detectors) to the DetElements (-> will use this for now)
- both options will work for PandoraPFA as it needs only little geometry information (a la GEAR)

DDRec versus GEAR

- need to be able to run existing GEAR based reconstruction code MarlinReco, MarlinTrk, PandaraPFA,... with new simulation model
- need an (intermediate) interface from DDRec to GEAR

- defined detector data structures, filled on detector construction
- added these to all tracker and calorimeter drivers (S.Lu)
- wrote 'quick and dirty' throw away code to generate a GEAR file from this
- in order to run existing reconstruction

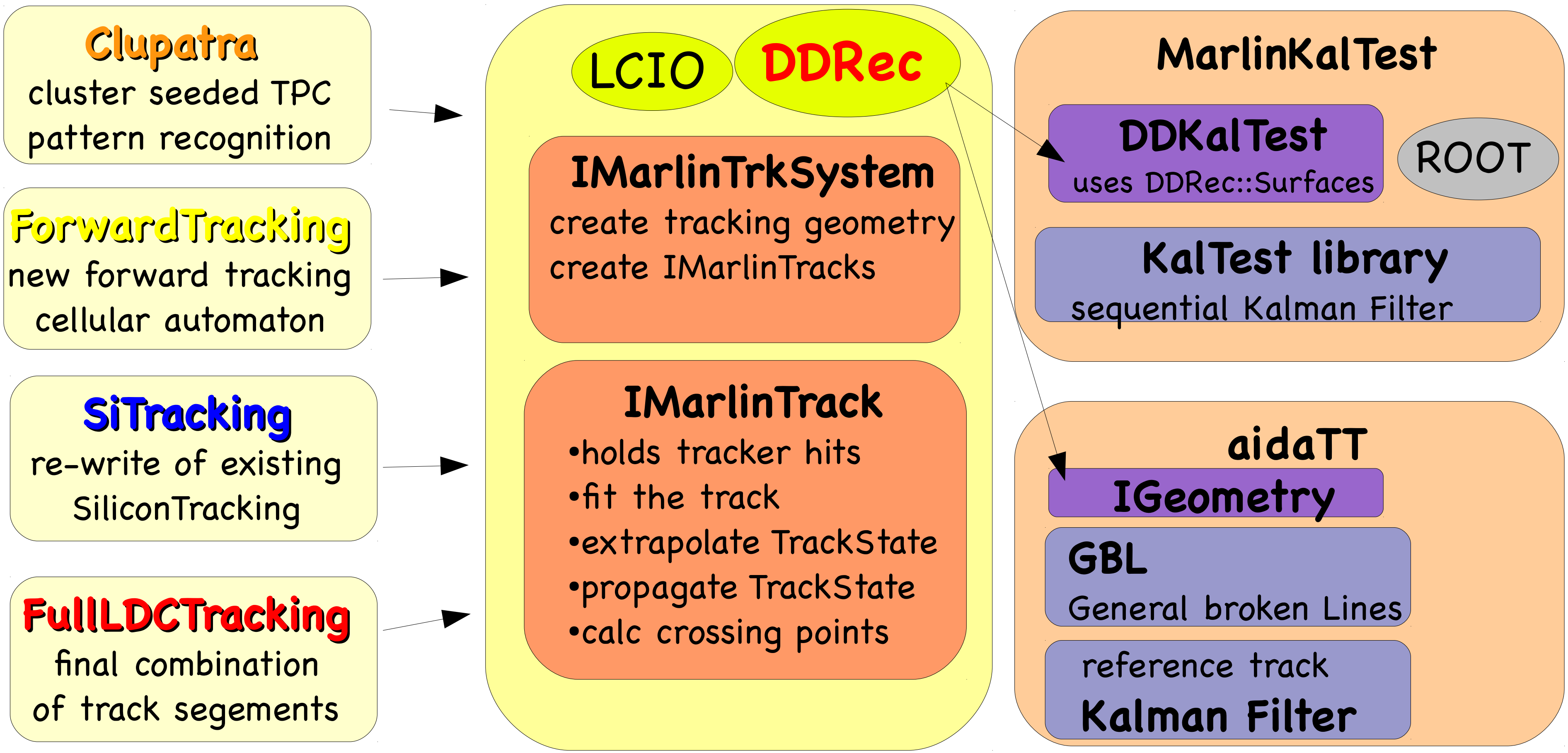
Detector Data Structures	
FixedPadSizeTPCData	TPC
ZPlanarData	VXD, SIT, SET
ZDiskPetalsData	FTD
ConicalSupportData	BeamPipe
LayeredCalorimeterData	Ecal, Hcal, BeamCal, Lumical, LHCal, ...

```
convertToGear GearForILD ILD_o1_v05.xml gear_ILD_o1_v05_dd4hep.xml
```

- continue to further develop the DDRec interface and eventually completely replace GEAR w/ DDRec in Marlin based reconstruction ...

IMarlinTrk interface for LC tracking

- the surfaces and materials from DD4hep/DDRec will replace the pre-existing GEAR geometry description in iLCSoft
- existing pattern recognition tools can be used (almost) unmodified
- new (generic) pattern recognition tools can be developed for DD4hep based detectors



DDKalTest

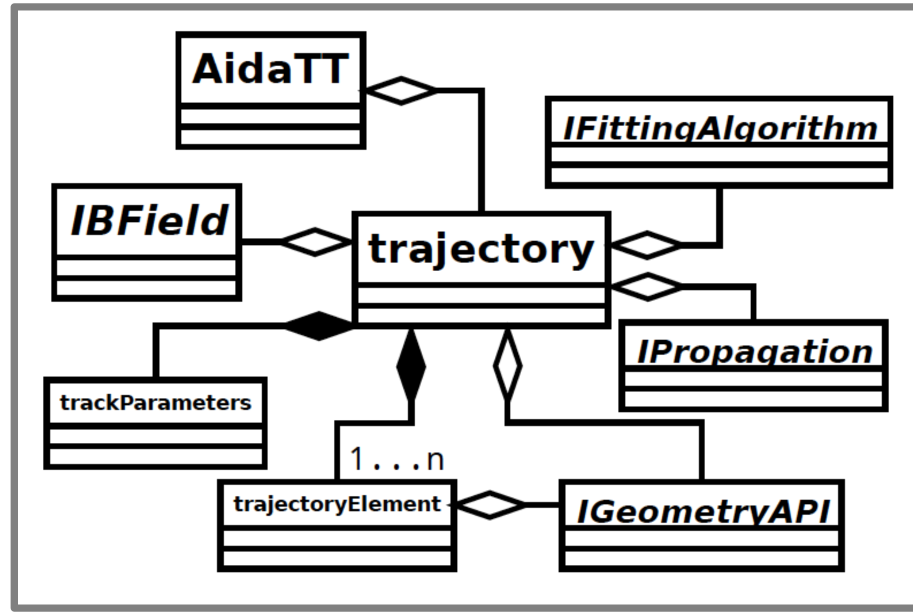
- DDKalTest: implementation of measurement surface and hits classes needed for KalTest Kalman filter used in MarlinTrk
  - re-implement some classes from KalDet that used Gear to instantiate the surfaces now using the DDRec::Surfaces from the DD4hep model
  - no GEAR file is needed !
- DD(Parallel)PlanarMeasLayer:
  - planar measurement layers (parallel and orthogonal to z)
  - works for 1D and 2D hits: VXD, SIT, SET, FTD, (all silicon tracker)
- DDCylinderMeasLayer:
  - cylindrical measurement layers parallel to z: TPC

- with DDKalTest we can run the KalTest Kalman filter on any tracking detector that has the DDRec::Surfaces implemented without additional glue code!

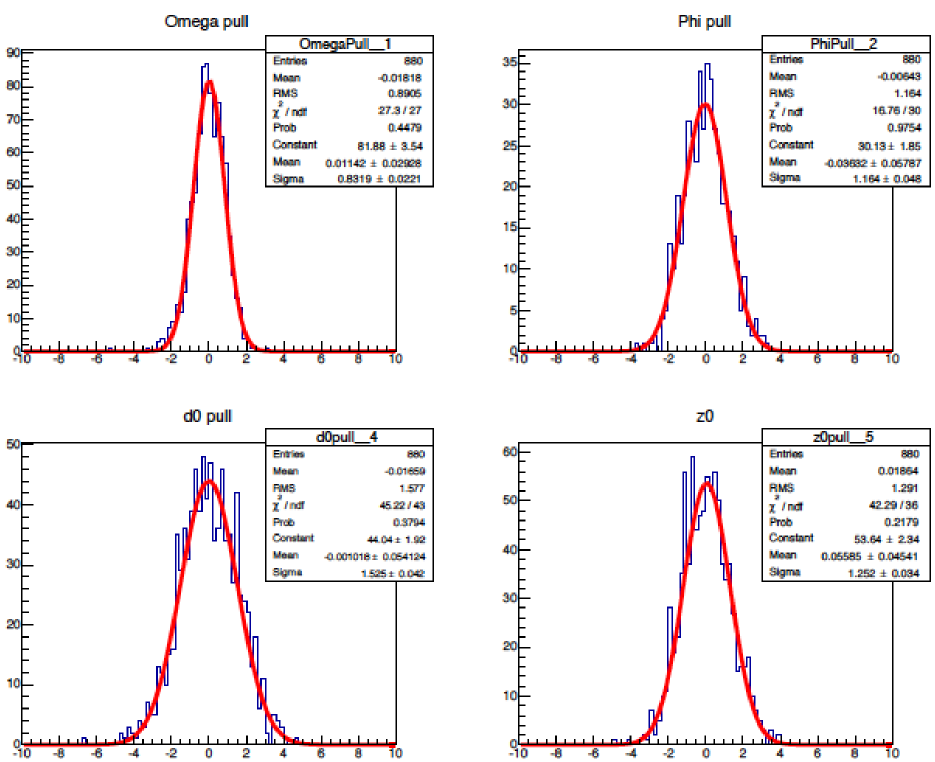
aidaTT

C.Rosemann

- tracking tool developed in AIDA WP2
  - (C.Rosemann, Y. Voutsinas)
- transparently use Kalman-Filter or GeneralBrokenLines GBL
  - needed for Millipede alignment tool
- interface to DDRec::Surfaces



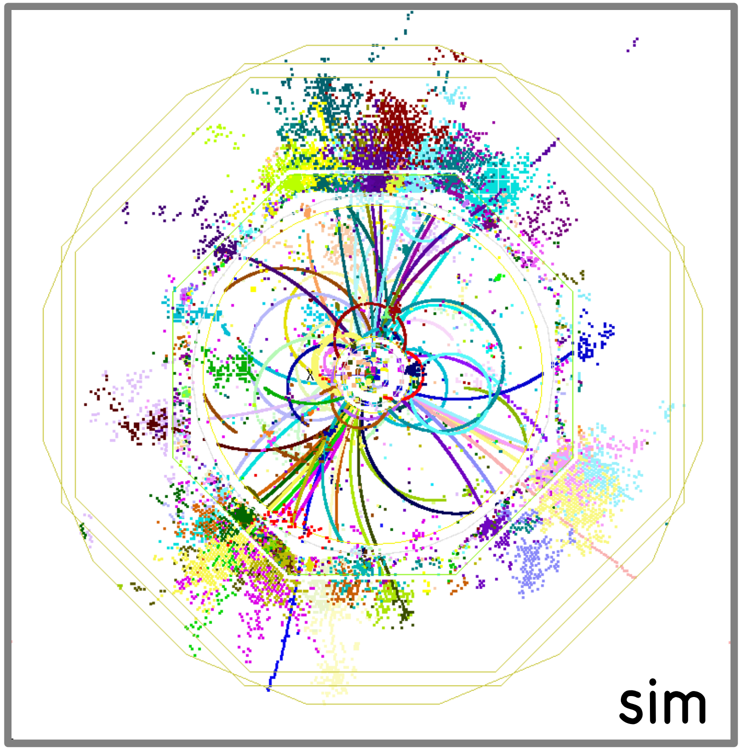
https://svnsrv.desy.de/desy/aidasoft/aidaTT/trunk

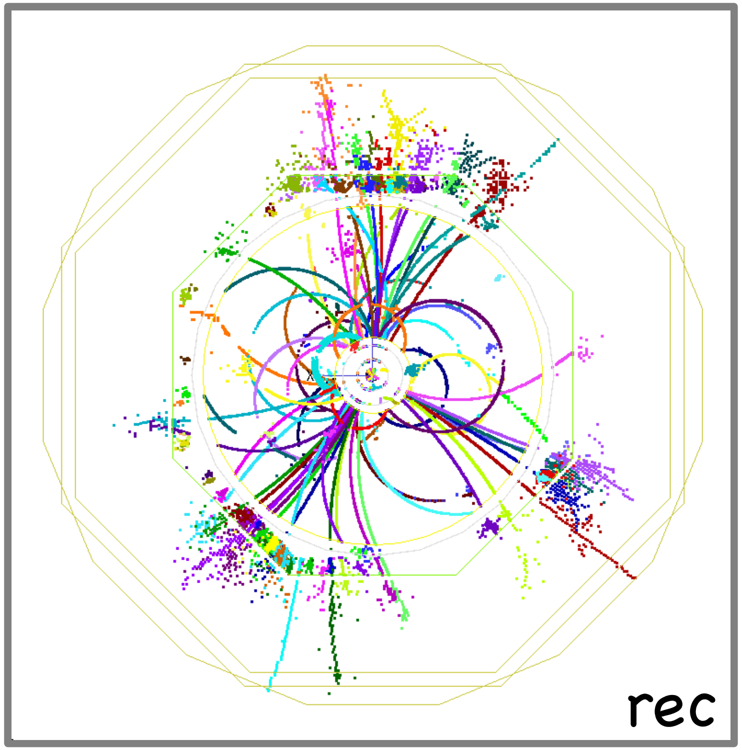


example: pull distributions for ILD VXD/SIT

- Status:
  - implementation of GBL exists
  - testing and inclusion of material effects ongoing
  - next step: implementation of MarlinTrk interface

Running existing reconstruction





- simulated ttbar event in the ILD_o1_v05 DD4hep simulation model reconstructed with the standard Marlin based reconstruction (using the interface from DDRec to GEAR)
- -> prove of concept
- still many things to do ...

Towards a timeline for software I

- ingredients and missing items needed for defining a timeline for ILD software development:
  - need first functional version of ILD_o1_v05 in DD4hep/DDSim **~done**
  - need functional interface to existing reconstruction (GEAR/DDRec) **~done**
  - need testing and validation
  - define the ILD optimization models - how many (2-3) ?
    - reference detector + smaller detectors ...
  - implement these models
  - need testing and validation
  - define the physics benchmarks/data samples that need to be processed
    - 250 GeV, 350 GeV, 500 GeV full SM ?
  - finalize the Grid production infrastructure w/ ILCDirac **~done**
  - adapt reconstruction to new models
  - need testing and validation
  - estimate the CPU (and storage) needs
  - the actual Monte Carlo mass production
  - ...

Towards a timeline for software II

- a first very rough estimate of the effort involved:

item:	estimated effort*	comment
first version of ILD_o1_v05 in DD4hep	1 pm	~done
interface to reconstruction	1 pm	~done
testing and validation	2 pm	
define ILD optimization models	1 pm	start at ILD meeting ?
implement these models	3 pm	# models ?
testing and validation	3 pm	# models
define physics benchmarks	1 pm	
Grid production infrastructure	1 pm	~done
Grid Monte Carlo simulation	1 pm (3 months)	# channels/processes, # CPUs
adapt reconstruction (incl. testing)	2 pm	
Grid reconstruction	1 pm (1 month)	
Total	17 pm	

*pm: full time person month

- calendar time depends on number of (experienced) people
- Not included here: producing generator files and analyses !

Summary & Outlook

- lcgeo - new DD4hep detector geometry description and simulation package has a first complete simulation model ILD_o1_v05
  - ported/re-implemented from Mokka
  - have defined envelopes for all detectors
- DDRec - interface to reconstruction exists: surfaces for tracking, data structs for calorimetry
- interface to GEAR is basically ready -> can run 'old' reconstruction
- **To Do:**
- address some open issues (sens. detectors) and start validating the existing ILD model
- adopt existing reconstruction to use DDRec consistently
- **Outlook**:
- define and implement 2-3 models for detector optimization
  - start at ILD meeting in Tsukuba
- try to find people that can help so we can have the new models in the ~~first half of 2015~~ end of summer 2015