Simulation and Reconstruction for the new CLIC detector model

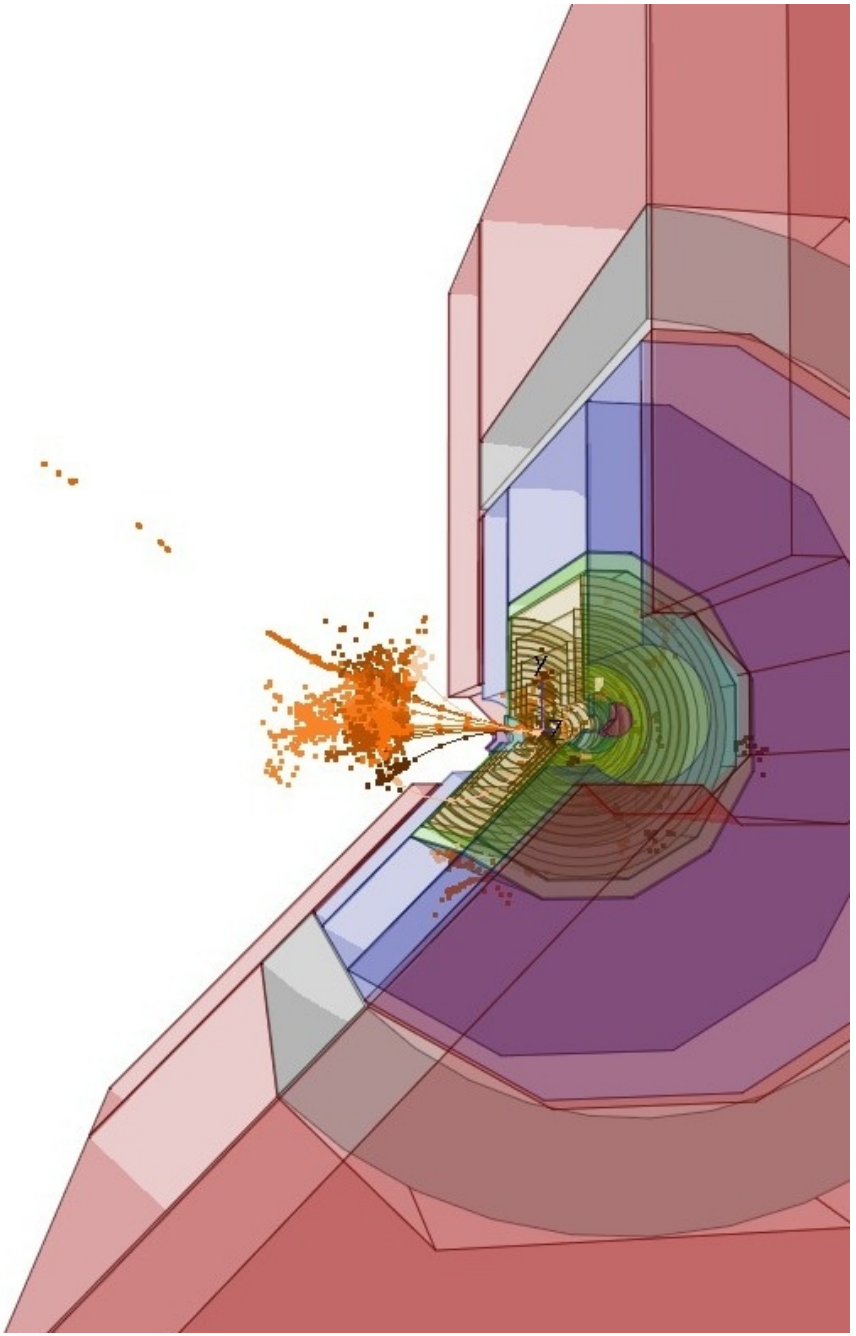
Frank Gaede, CERN/DESY

CLIC Workshop 2015

CERN, 26-30 January 2015

Outline

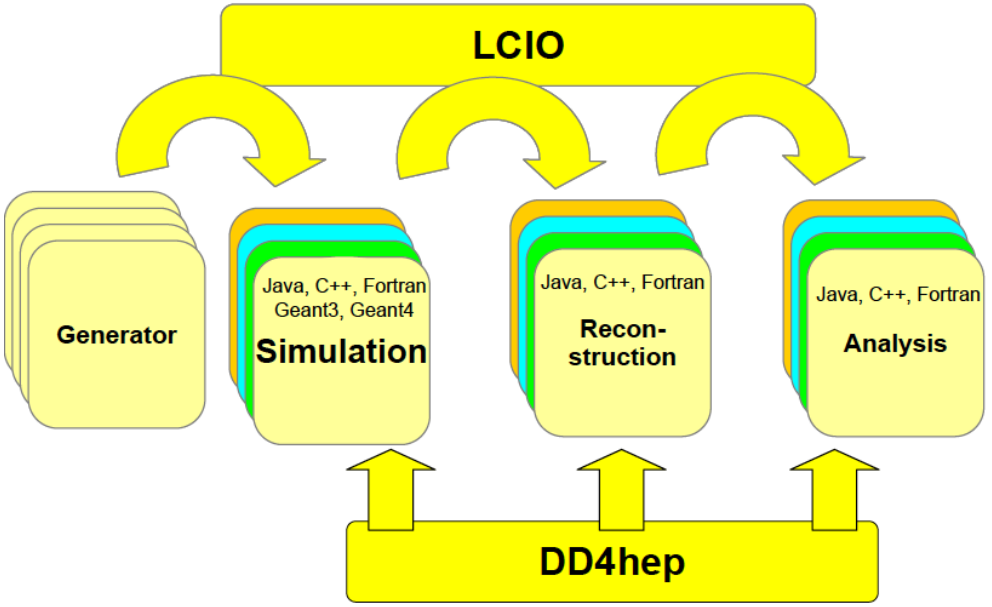
- Introduction
- DD4hep and lcgeo
- first version of CLIC model
- overview iLCSoft reconstruction
- reco interface: DDRec
- track reconstruction
- calorimeter reconstruction
- Summary/Outlook



people involved:
M.Frank, C.Grefe, A.Sailer, N.Nikiforou, S.Lu, M.Petric, R.Simoniello, F.G.

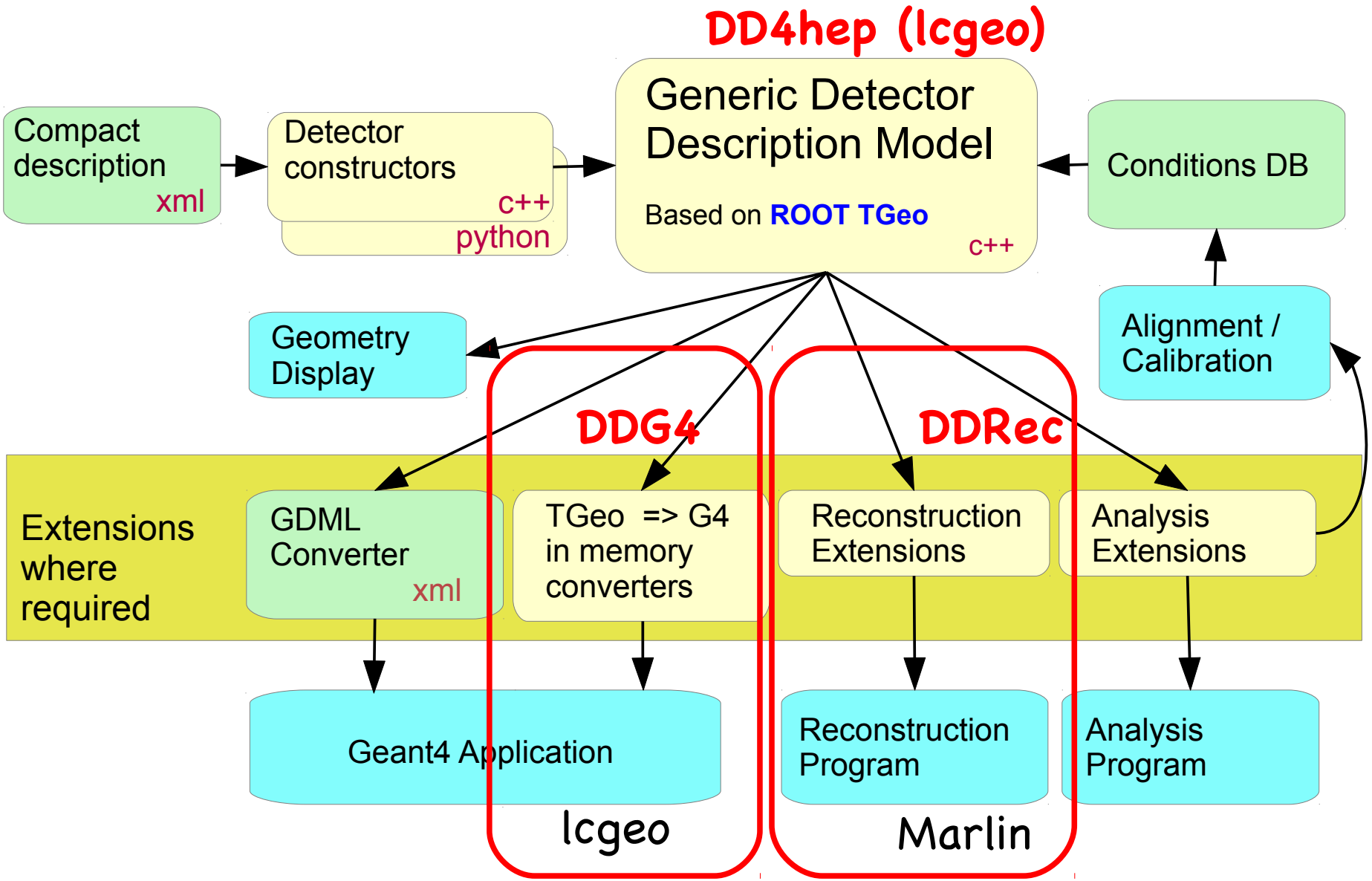
Introduction

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



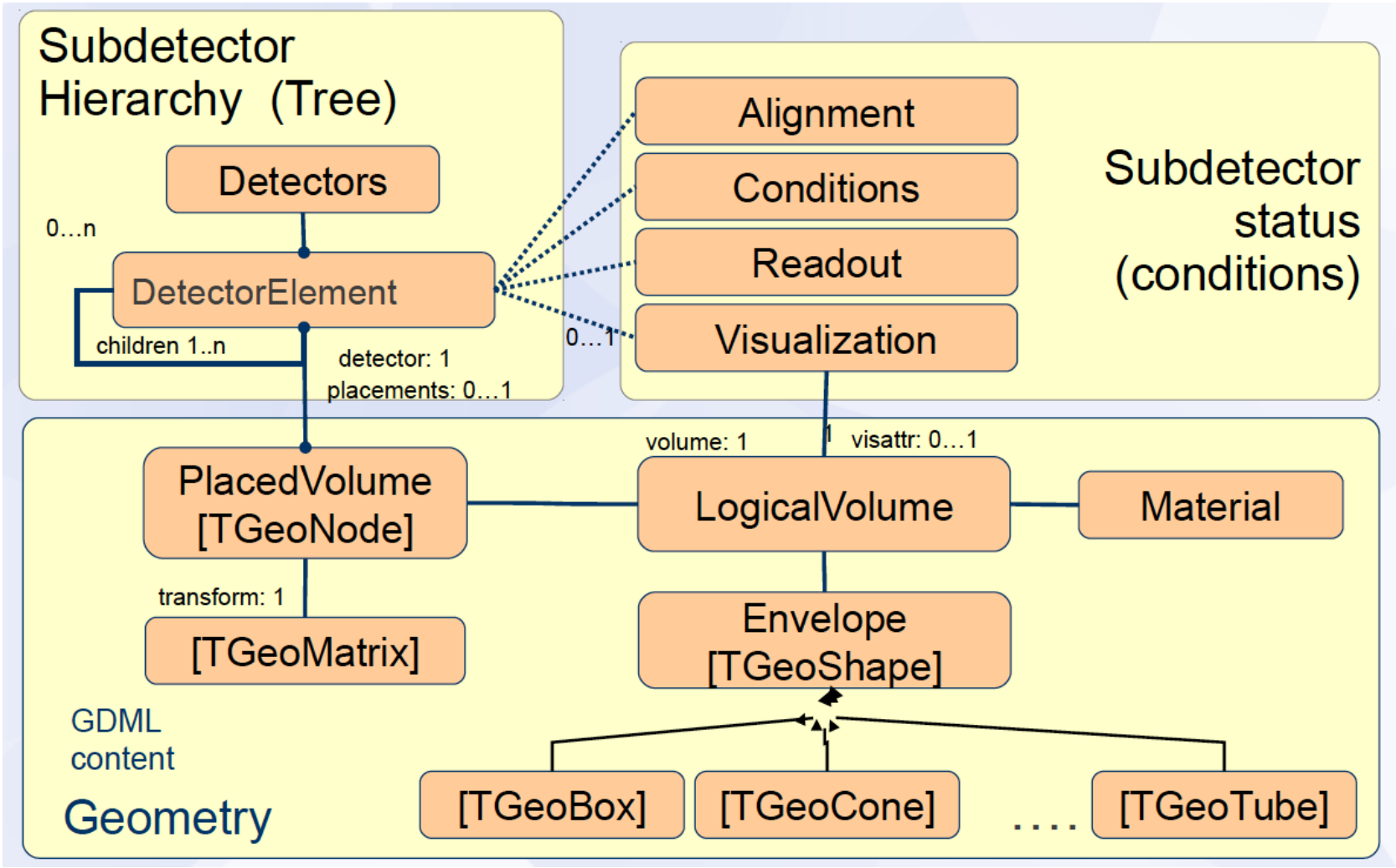
  - defining a common geometry API is the second step - after the common EDM: LCIO - that is needed to have an open and modular software framework
- allows to share existing common software tools between ILC and CLIC and also develop new common simulation and reconstruction tools
- simulation uses **DDG4** and lcgeo
- reconstruction uses **DDRec** and existing Marlin reco tools

DD4hep: schematic overview



one source of geometry to feed into all HEP applications

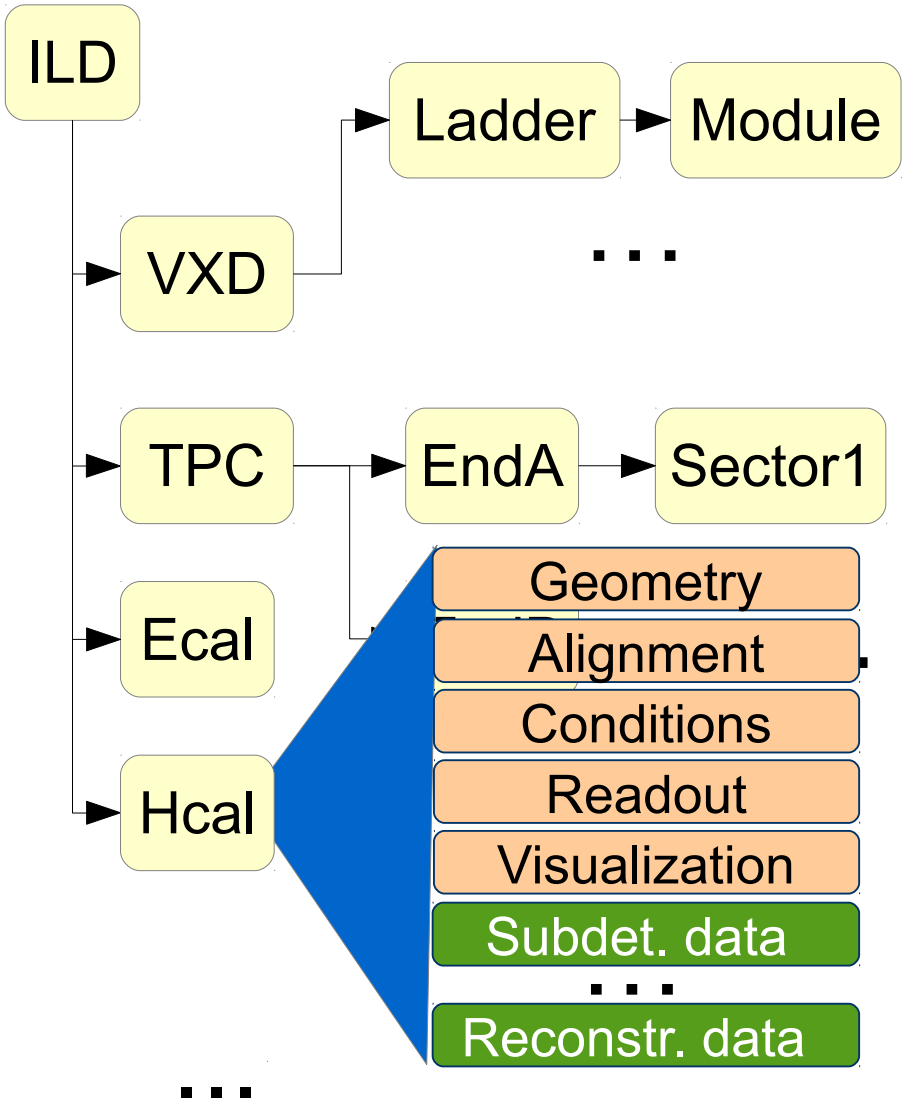
DD4hep geometry implementation



- DD4hep uses the TGeo geometry classes to instantiate geometry tree -> can use all TGeo features directly
- additional (user) code added to detector element class

DD4hep: detector elements

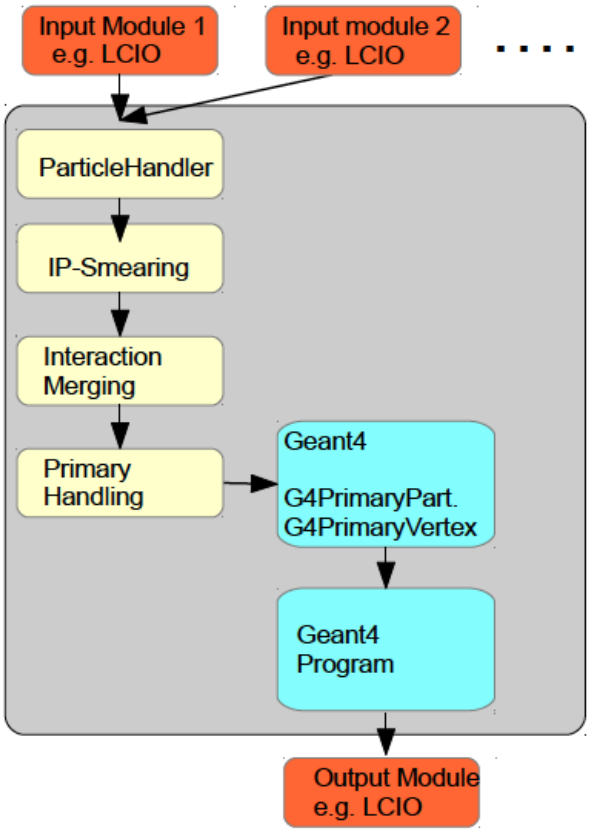
- detector is described in a tree-like hierarchy of DetectorElements:
  - sub detectors or parts thereof:
    - modules, sensors, ...
  - DetectorElement describes:
    - Geometry
      - Shape, material, ...
    - detector properties:
      - readout, alignment, conditions
    - visualization
- allows additional user/ experiment specific data



Simulation

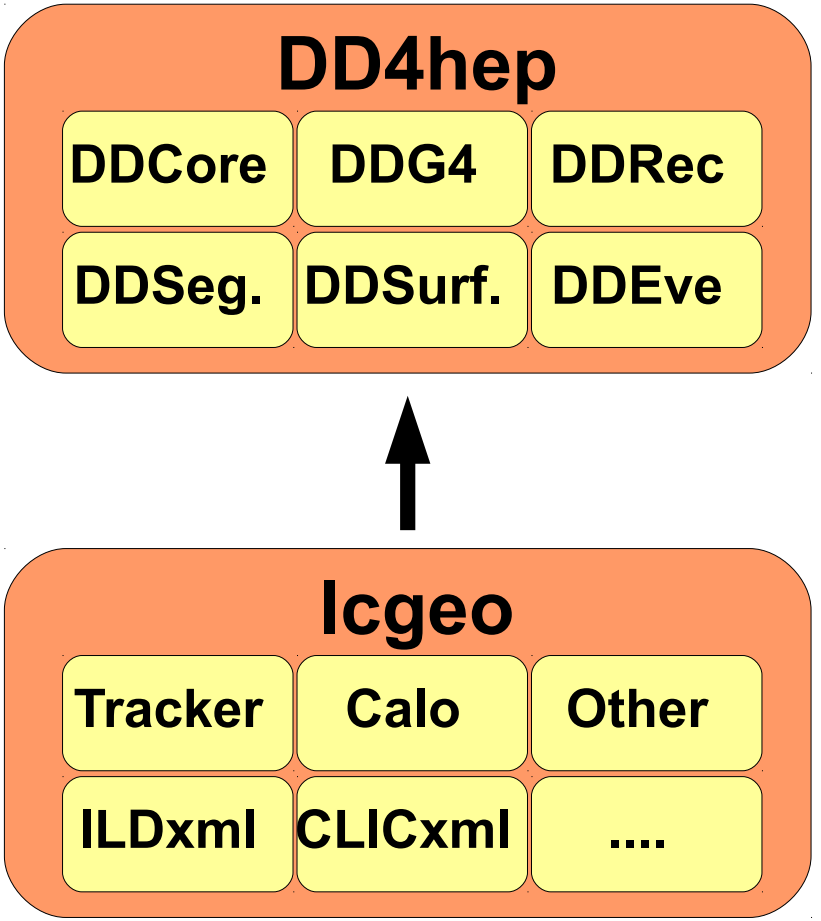
DDG4 - built in Geant4 gateway

- 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,...
- features :
  - full flexibility in sensitive detectors
  - can use extension code in simulation and reconstruction
  - supported by CERN for CLICdp, ILC and FCC



lcgeo: detector description

- created package **lcgeo** as a common LC detector description package for ILD and CLIC (and SiD)
- simulation is (automatically) provided via DDG4
- where possible use the common (CLICdp/ILD) geometry drivers for sub-detectors, e.g. beamcal, ECal, Hcal (?),...
- for ILD: started with porting current Mokka model ILD_o1_v05 (others to follow)
- for CLIC: started with porting the existing CLIC_SiD model to DD4hep
  - fairly straight forward as original design of DD4hep inspired by SiD compact format



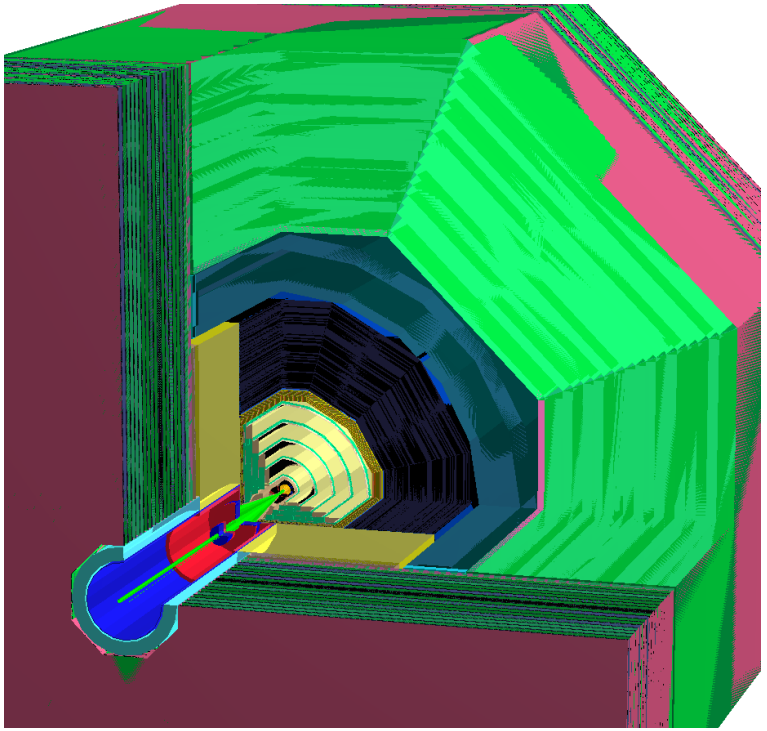
https://svnsrv.desy.de/viewvc/ddsim/lcgeo/trunk

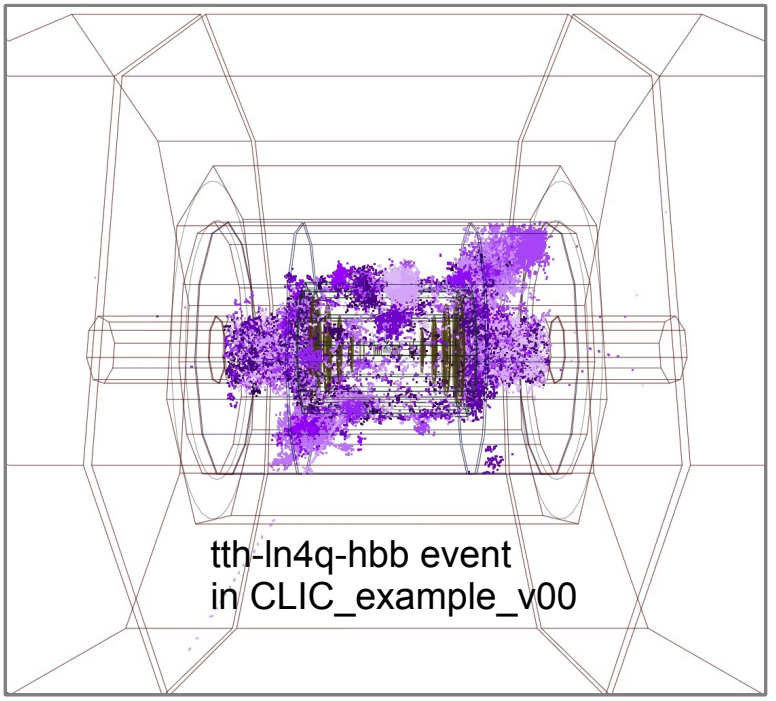
lcgeo naming and versioning

- detector model: DETECTORNAME_OPTION_RELEASEVERSION
  - ILD_o1_v05
  - CLIC_example_v01
- driver names: SUBDETECTOR_OPTION_RELEASEVERSION_geo.cpp
  - HCalBarrel_analogue_v01_geo.cpp
  - VertexBarrel_doublelayer_v01_geo.cpp
- sub-detector xml files:
  SUBDETER_OPTION_RELEASEVERSION_VERSION
  - HCalBarrel_analogue_v01_07.xml
  - VertexBarrel_doublelayer_v01_42.xml
- in most code sub detectors are refered to by their base name:
- VTX, SIT, FTD, TPC, SET, HCal, ECal, BeamCal, LCal, LHcal,..

- above scheme allows to combine software versions and different parameter versions in a well defined way
- using svn copy will allow to keep track of changes and differences

CLIC simulation model in DD4hep

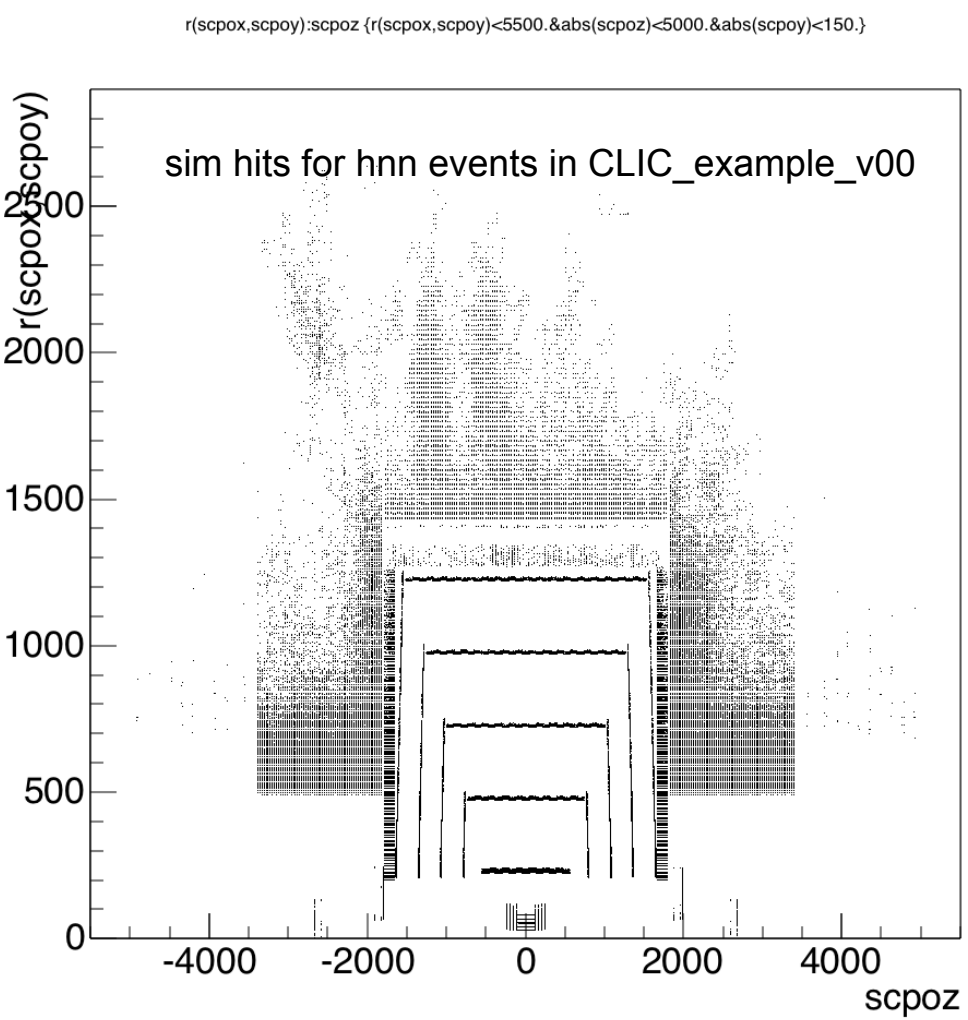




- CLIC_example_v00: port of existing CLIC_SiD model to DD4hep/lcgeo
- basic simulation functionality exists: create SimTracker/CalorimterHits
- serves as starting point for the development of the new CLIC simulation model

CLIC simulation next steps

- check functionality of the model:
  - e.g. study hit maps
- start to modify the parameters according to the new CLIC model document
- develop new drivers - if needed - based on technology/layout decisions:
  - Vertex & Silicon tracker layout
    - single vs. double layers
  - Hcal: # layers, cell sizes
  - forward region: coverage, QD0

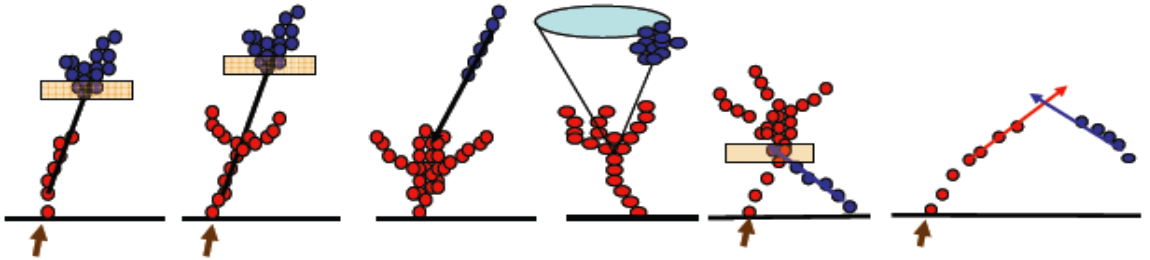


- identified responsible person for every sub detector
- overall coordination: M.Petric
- having a running version this summer should be feasible

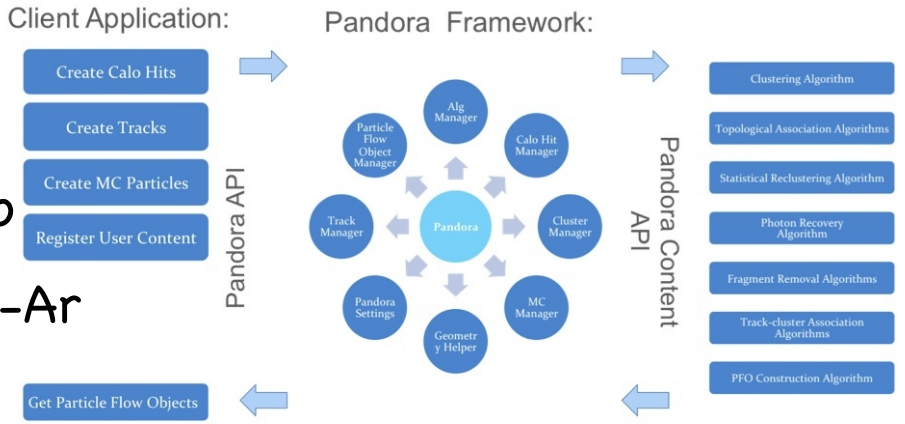
Reconstruction

main reconstruction tools in iLCSoft

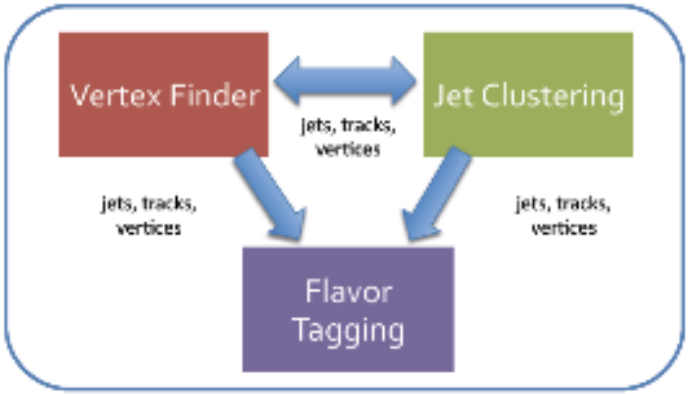
- **MarlinTrk**
  - complete tracking toolkit
  - more on next slides



- **PandoraPFA**
  - Particle Flow Algorithm
  - originally developed for ILD, now applied to
  - CLICdp, Calice, SiD, neutrino physics (Liquid-Ar TPC), LHC...
  - redesigned to be framework independent
    - -> minimal glue code needed (geometry)

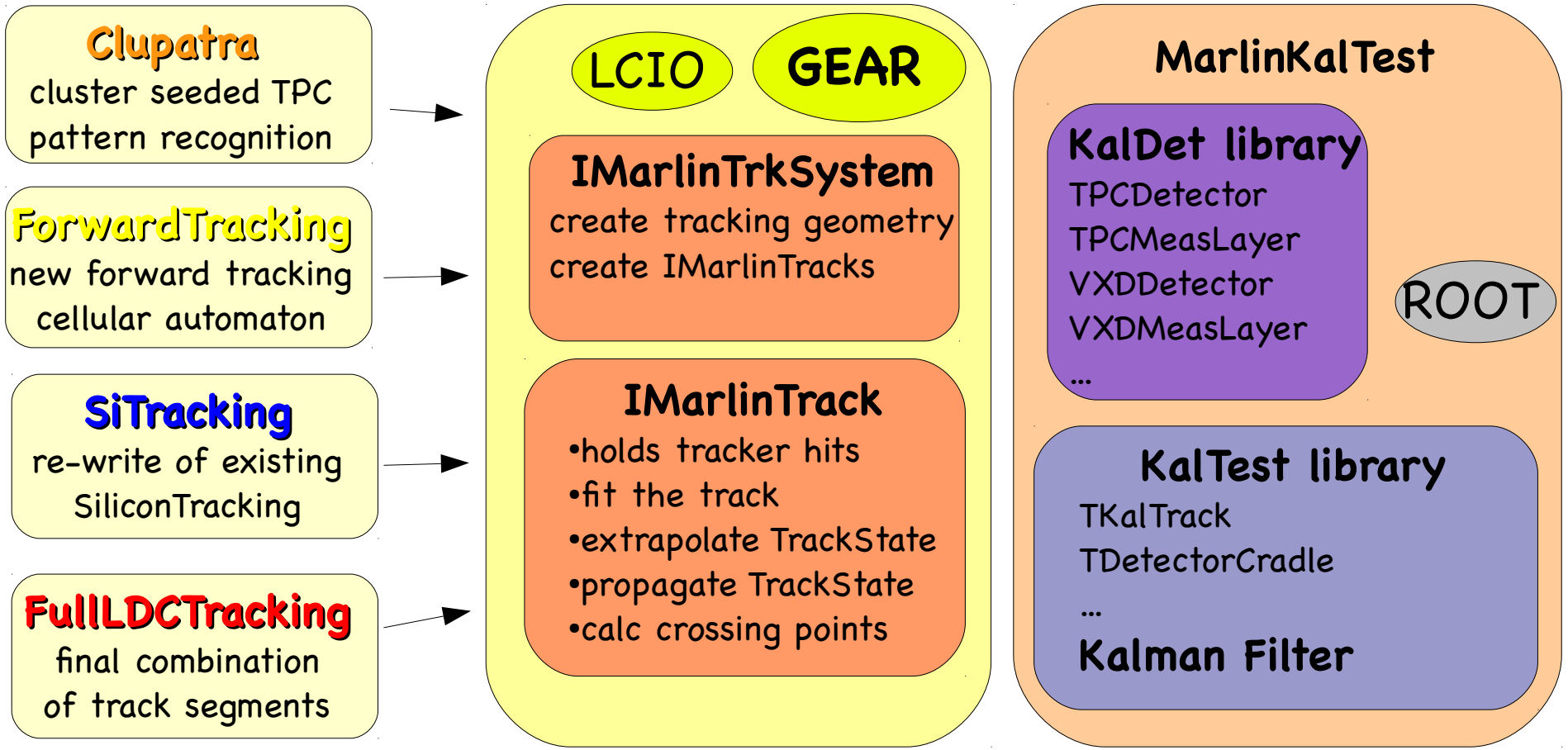


- **LCFIPlus**
  - flavor tagging - based on vertex reconstruction (ZVtop,...) and Neural Networks
  - used by all LC concepts (no geometry)

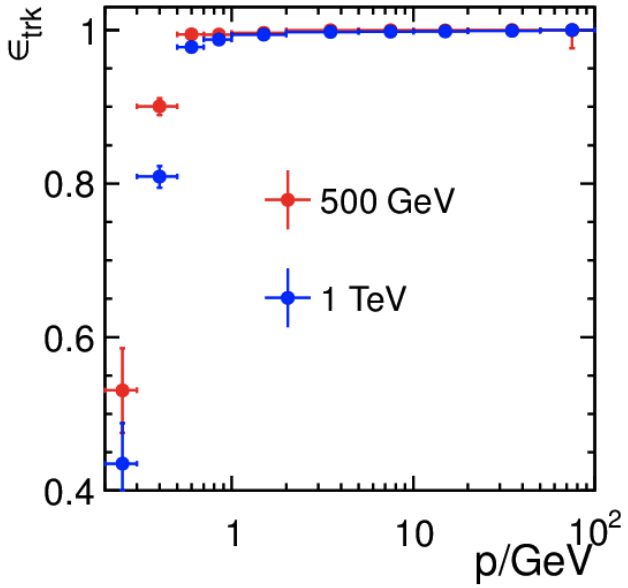


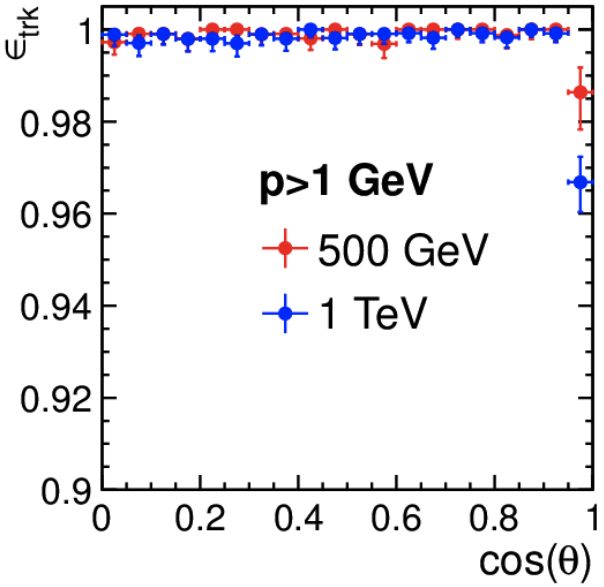
IMarlinTrk interface for (ILD) tracking

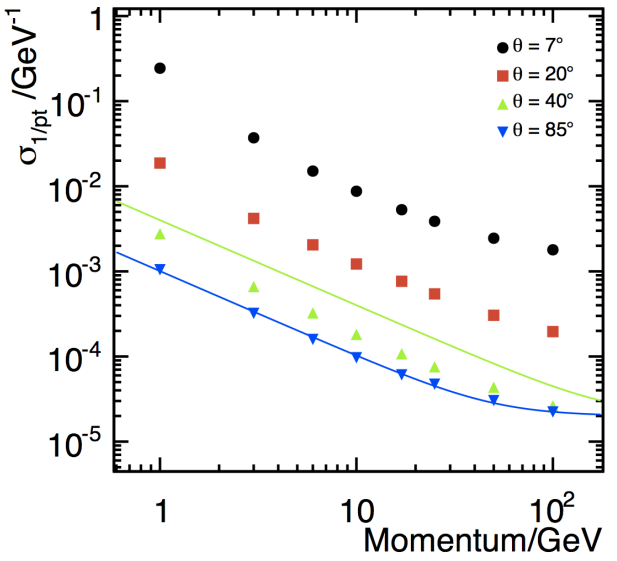
- common API for developing tracking code (in Marlin)
- provides loose coupling between pattrec and fitting
- defined abstract interface IMarlinTrk and implement using KalTest/KalDet
- originally developed for new ILD tracking used in the DBD
- will be adapted to DDRec to be also usable for new CLIC simulation



MarlinTrk Performance for ILD







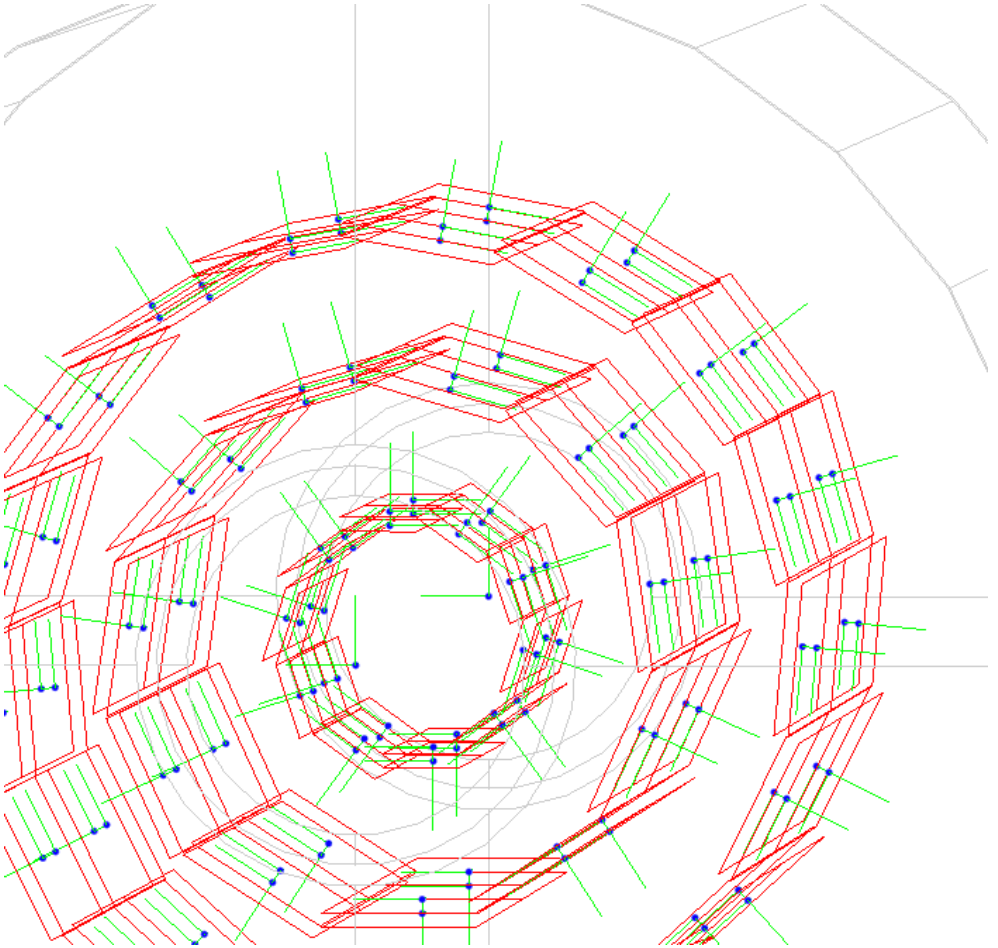
eff_trk > 99.8%, p>1GeV, cos(th)<0.95

$$\sigma_{1/p_T} = \frac{2 \times 10^{-5}}{GeV} \oplus \frac{1 \times 10^{-3}}{p_T \sin\theta}$$

- the new ILD tracking based in IMarlinTrk shows excellent tracking efficiency and transverse momentum resolution
- used for large scale MC production for TDR (DBD) -> "battle proven"
- should be able to adopt the track fitting and some of the pattern recognition also for new CLIC simulation (not Clupatra ;-( )
- provided the code is changed to use DDRec rather than GEAR

DDRec: surfaces for tracking

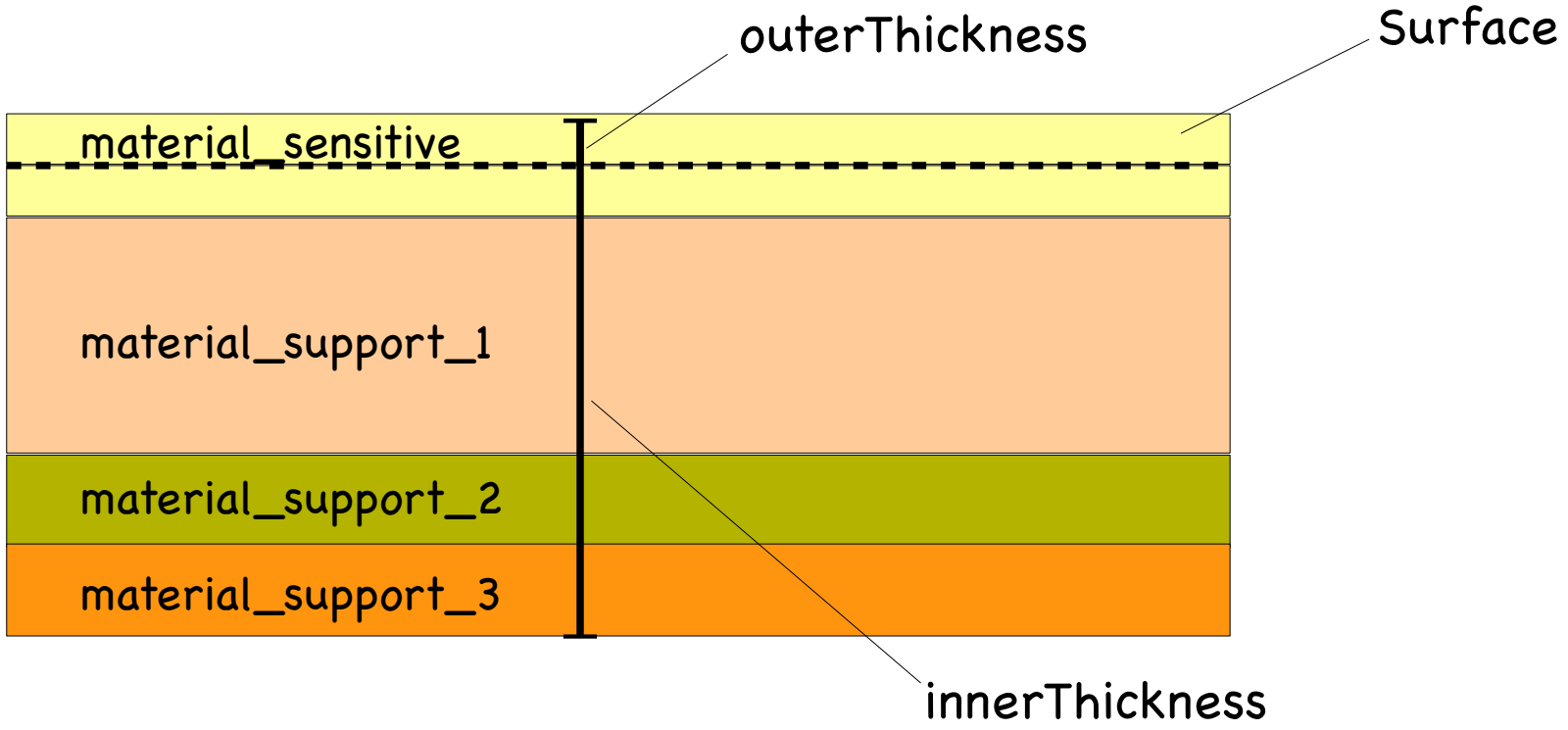
- tracking code needs a special interface to geometry:
- measurement and dead material surfaces (planar, cylindrical)
- surfaces are attached to volumes (defining boundaries) and provide:
  - u,v, normal, origin
  - inner and outer thicknesses and material
    - material is automatically averaged from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



example: surfaces attached to ILD vertex detector in new DD4hep model ILD ported from Mokka

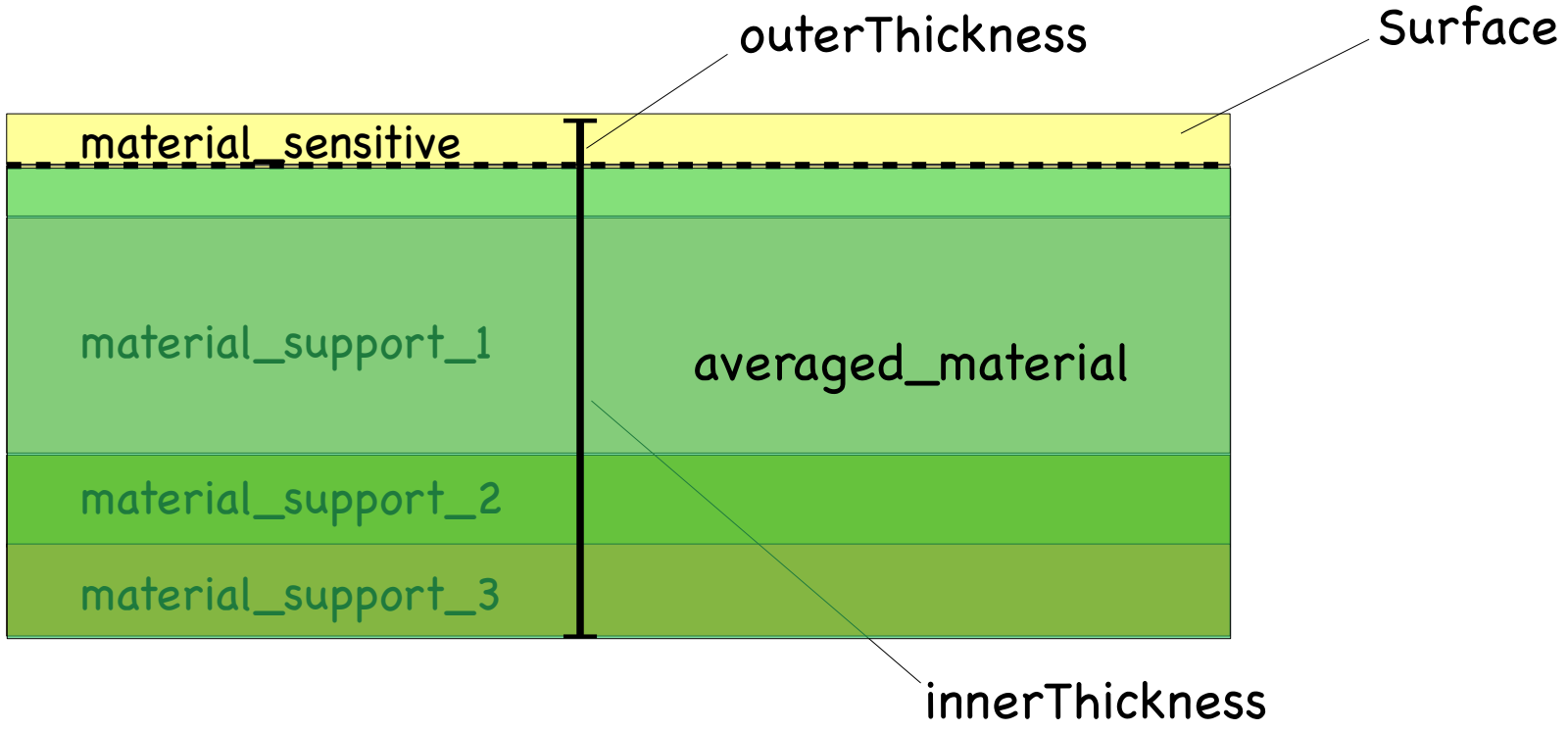
DDRec: materials and surfaces

example: Si-waver for tracking in simulation model



DDRec: materials and surfaces

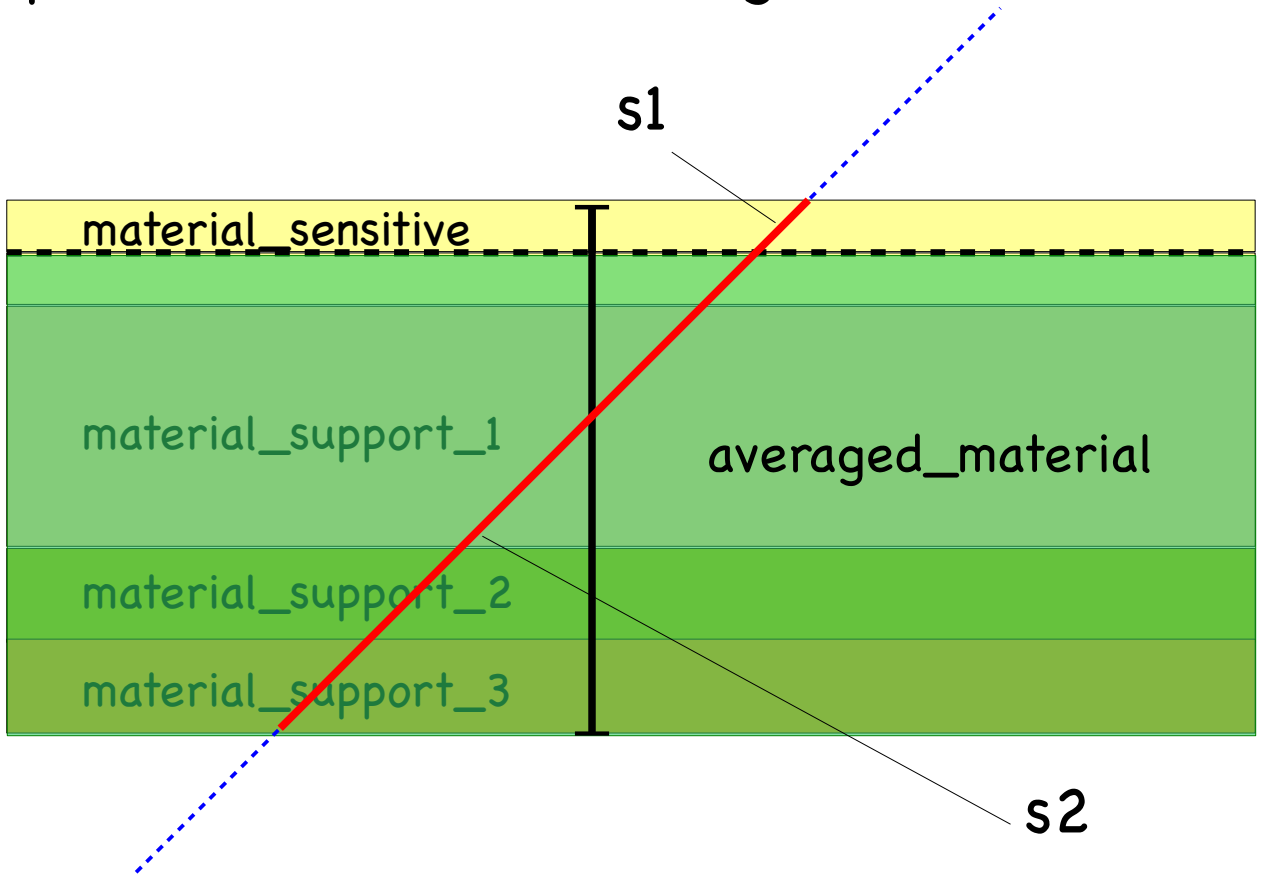
example: Si-waver for tracking in simulation model



averaged material is automatically computed from detailed simulation model

DDRec: materials and surfaces

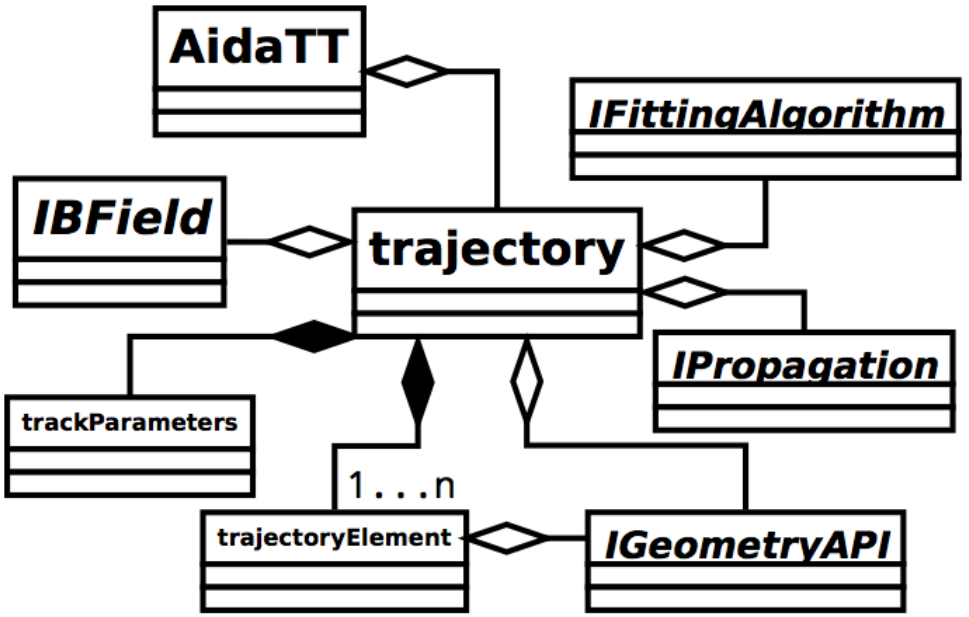
example: Si-waver for tracking in simulation model



use material properties A, Z, rho, radLen, intLen to compute effect of energy loss and multiple scattering along path lengths "through the surface" s1,s2

aidaTT

- Tracking Toolkit
  - track fitting (and finding)
    - GBL, Kalman, ...
  - track propagation, extrapolation
  - intersection calculation
- developed in AIDA WP2



- geometry implementation from dd4hep::DDSurfaces
  - coordinates, measurement directions, normal, material, insideBounds,...
  - tracking provides intersection with surfaces
- track parameters: L3/LCIO perigee parameters:
  - omega, D0, Phi0, Z0, tanL
- will eventually implement the IMarlinTrk interface

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 needed !

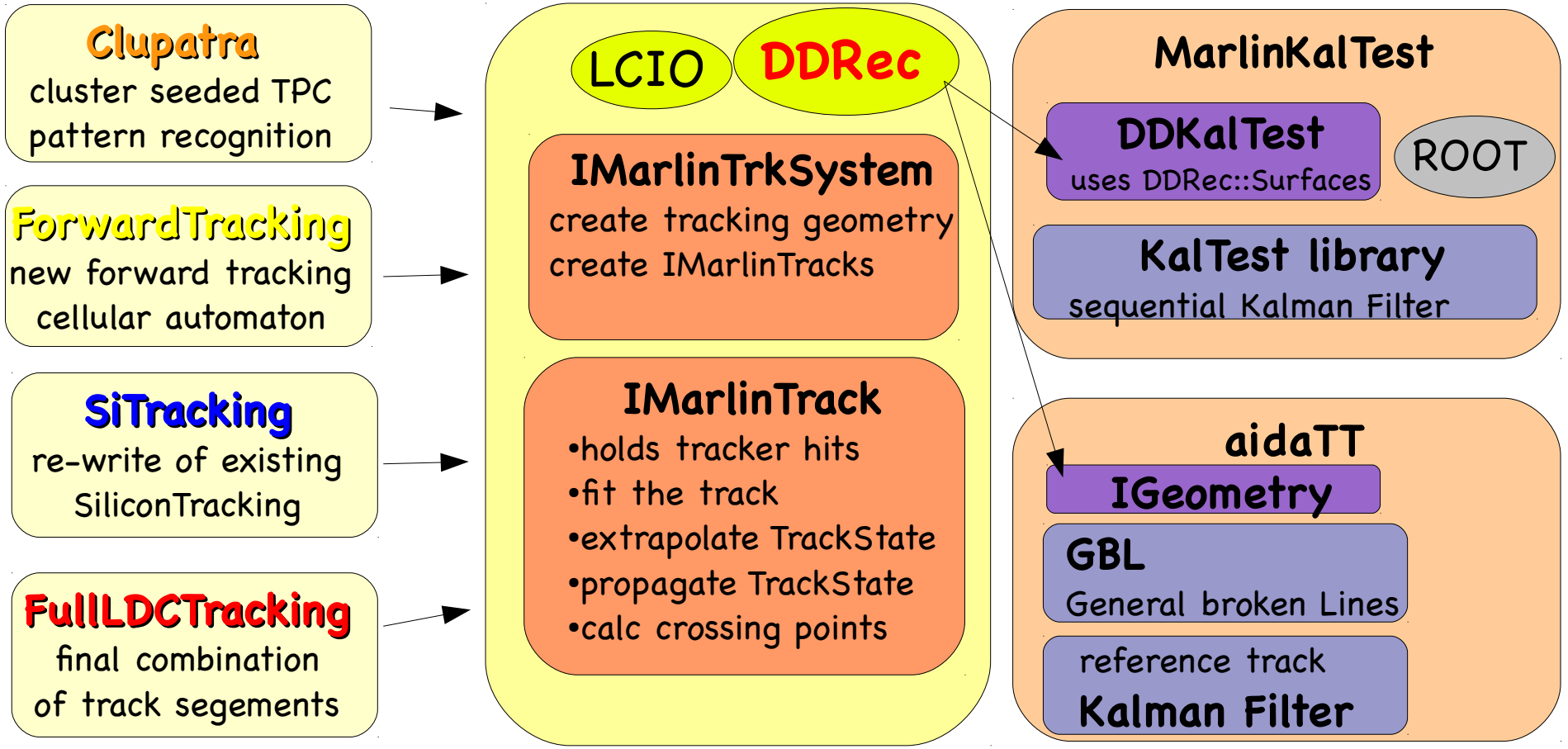
- DD(Parallel)PlanarMeasLayer:
  - planar measurement layers (parallel to Z)
  - global to local coordinate transforms (using the DDRec::Surface)
  - derivatives: @u,v @ omega, phi, tanL, d0, z0
  - intersection with helix (track):
    - from aidaTT for parallel to z (analytical)
    - from KalTest w/ newtonian method for arbitrary planes
  - works for 1D and 2D hits

aidaTT and DDkalTest status

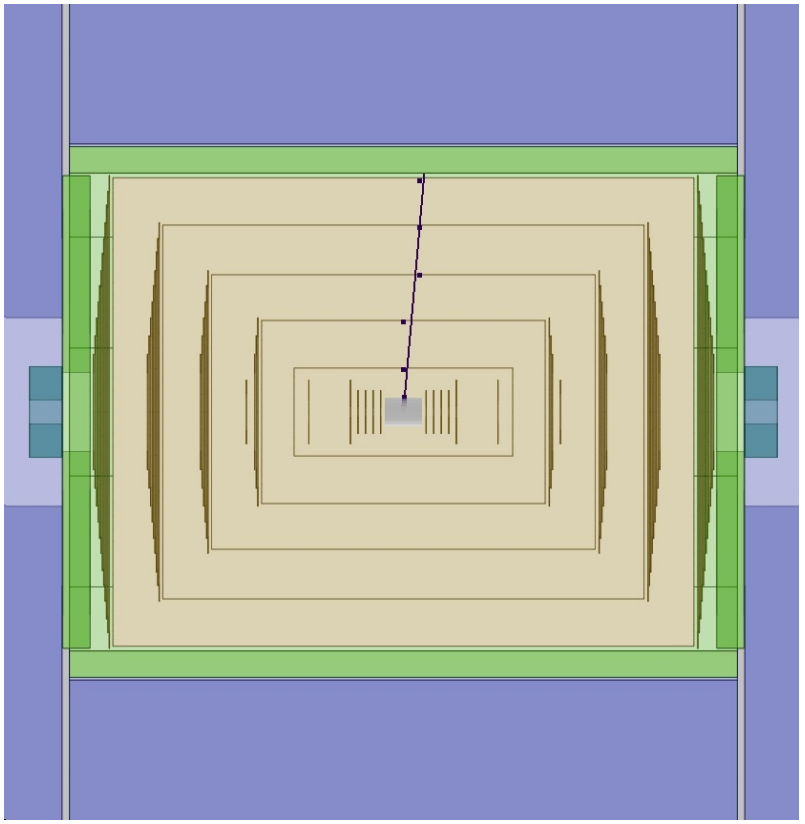
- DDKalTest
  - replaced GEAR geometry description with DDRec::Surfaces
  - implemented planar measurements for 1-d and 2-d hits
  - implemented energy loss and multiple scattering using DDRec::Material
  - to do:
    - cylindrical and disk measurement layers
    - then can run complete MarlinTrk tracking code with DD4hep based simulation
- aidaTT
  - implemented complete core functionality for track fitting with GBL
  - planar and disk measurement layers (using DDRec)
  - simple example for fitting tracks from Si-Trackers DD4hep models
  - to do:
    - add cylindrical layers
    - energy loss and multiple scattering
    - implement IMarlinTrk interface

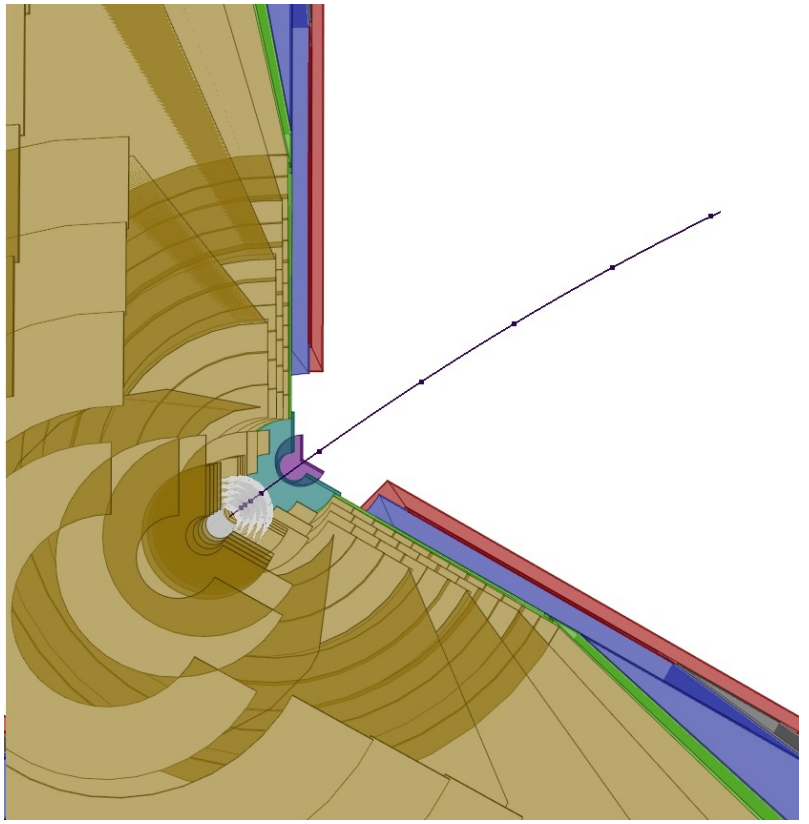
IMarlinTrk interface for LC tracking

- 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



first track reconstruction for CLIC

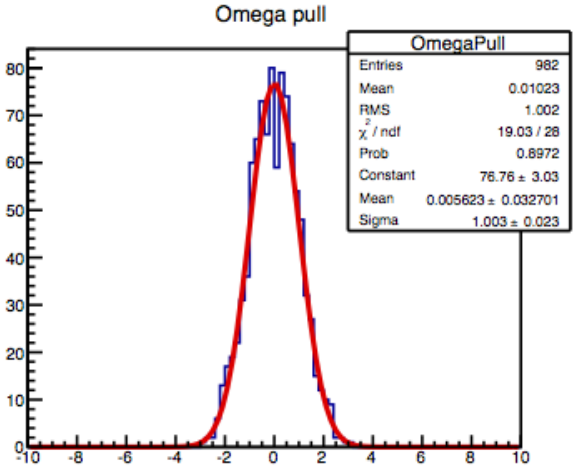


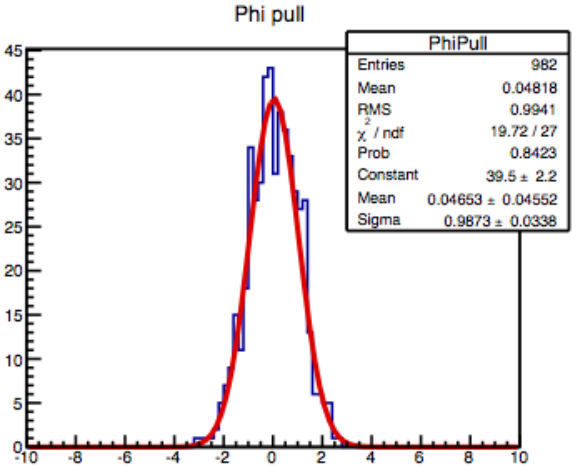


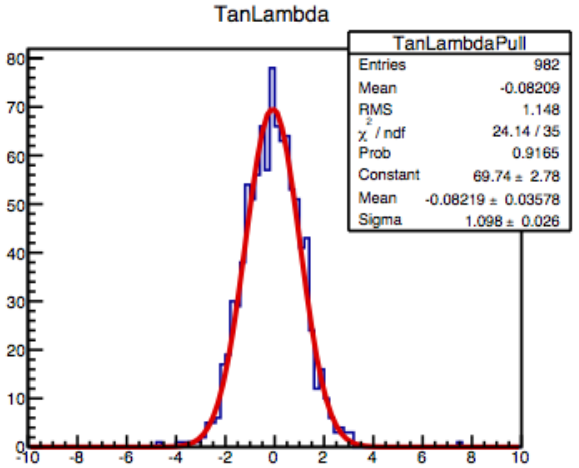
- reconstructed single muon 10 GeV in the CLIC example detector
- running SiTracking (for ILD) to find tracks in the CLIC vertex detector
- extrapolate vertex track to Silicon-Barrel-Tracker and pick up hits
  - using IMarlinTrk interface and DDKalTest

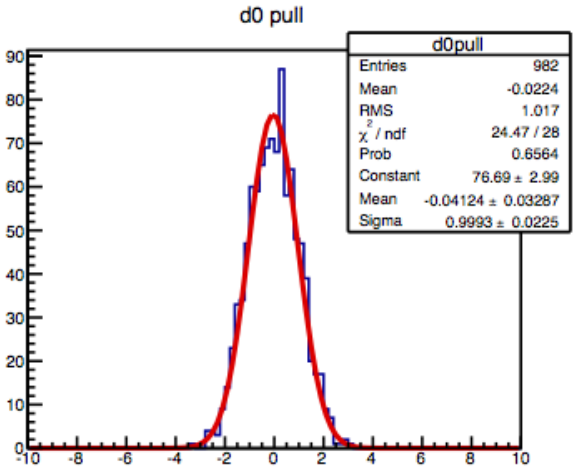
R. Simoniello
N. Nikiforou

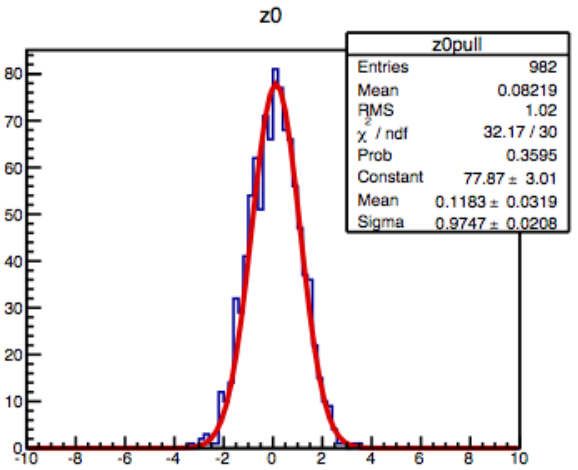
pull distributions for tracks in CLIC









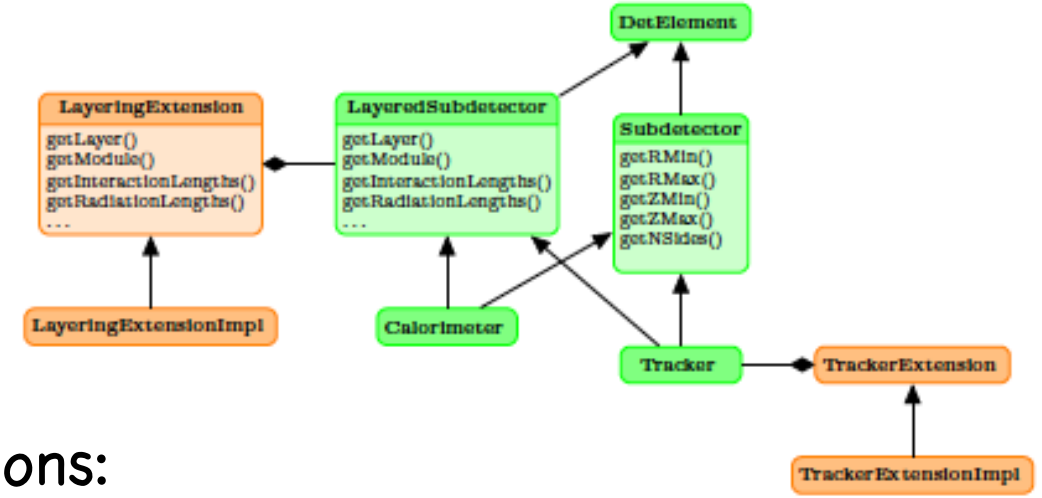


- pull distributions for track parameters in CLIC_example_v00 w/ DDKalTest
- single muons 10 GeV @ 85 deg (5 layer Vertex detector only)
- very preliminary -> need more detailed checks

tracking next steps

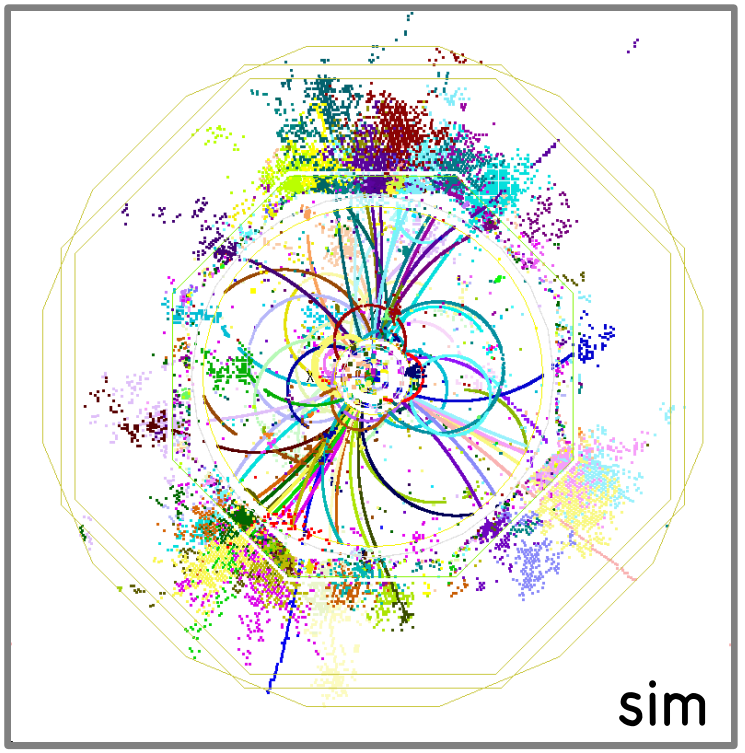
- need to add disk measurement layers to DDKalTest
- -> have complete track fitting available also for new CLIC simulation model
- pattern recognition:
- step 1: can use track cheater (MCTruth) to develop rest of reconstruction chain
- step2: use the new CA-based Vertex detector pattern recognition that has been developed for ILD (Y.Voutsinas) also for CLIC (double layer Vertex) and extrapolate hits outwards to Silicon Tracker
- step3: develop a CLIC specific pattern recognition by adopting and modifying the CA based pattrec to the specific layout of the new CLIC detector model

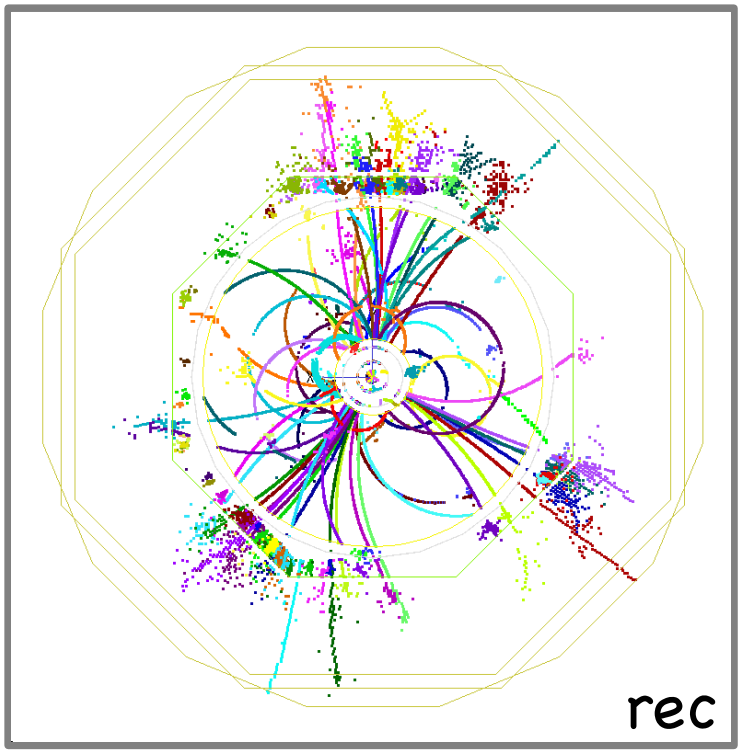
DDRec - interface for calorimeters



- two options:
  - use extension mechanism to define the high level interface for the calorimeter reconstruction in a hierarchy of LayeredSubdetector classes
    - provides access to layering structure and material properties per layer thickness, radiation and interaction lengths, cell sizes, ...
  - use simple data structs - similar to GEAR - to describe the calorimeter information
- both options will work for PandoraPFA as it needs only little geometry information (a la GEAR)
  - need to find someone to work on this ...

Running existing reconstruction





- example: simulated ttbar event in ILD DD4hep simulation model reconstructed with the standard Marlin based reconstruction (using an interface from DDRec to GEAR) -> prove of concept

- one the complete tracking code is functional for the CLIC model, including PandoraPFA and other tools (LCFIPlus) into the chain will be quite straight forward

Summary & Outlook

- the simulation and reconstruction for the new CLIC detector model will be based on DD4hep
- a first version of the simulation model in DD4hep/lcgeo exists: CLIC_example_v00 (based on CLIC_SiD)
- fully functional, will be adopted to the new CLIC model as decisions on technology and parameters become available
- the reconstruction will run as much as possible existing Marlin based reconstruction (MarlinTrk, Pandora, LCFIPlus)
- main work now focuses on the tracking:
- port the interface to the geometry to DD4hep
- develop a working pattern recognition for CLIC

goal: have a running version of the simulation and reconstruction running this summer:
ambitious but feasible