

DD4hep and Shareable Detector Geometry Description

André Sailer

CERN

Software and Physics Requirements for e^+e^- Colliders
January 17, 2020

Table of Contents



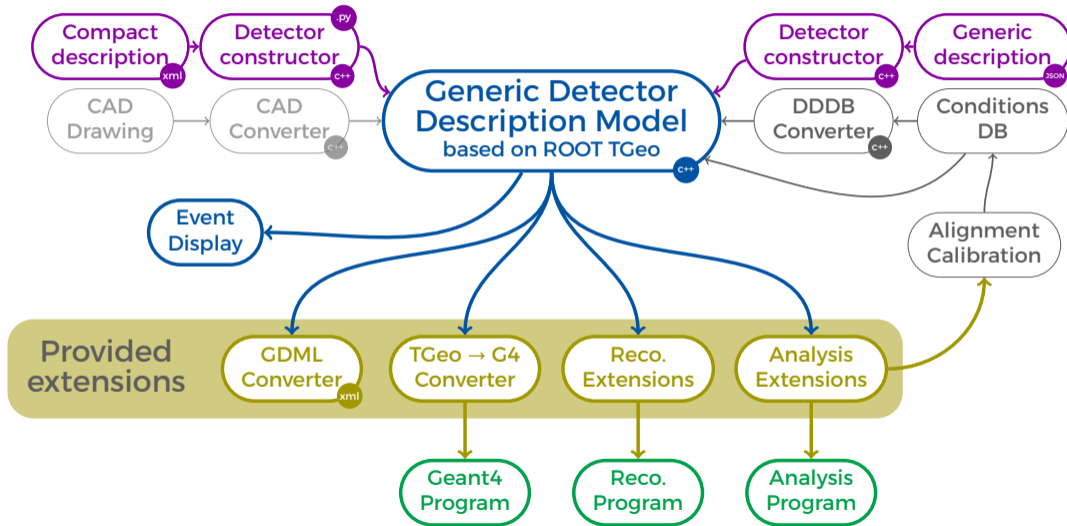
DD4hep

Simulation

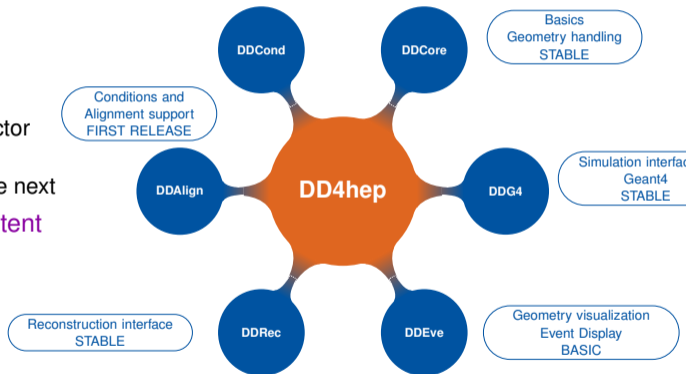
Describing a Sub-Detector

Reconstruction

The Big Picture



- ▶ **Complete Detector Description**
 - ▶ Providing geometry, materials, visualization, readout, alignment, calibration. . .
- ▶ **Supports full experiment life cycle**
 - ▶ Detector concept development, detector optimization, construction, operation
 - ▶ Facile transition from one stage to the next
- ▶ **Single source of information → consistent description**
 - ▶ Use in simulation, reconstruction, analysis, etc.
- ▶ **Ease of Use**
- ▶ **Few places for entering information**
- ▶ **Minimal dependencies**



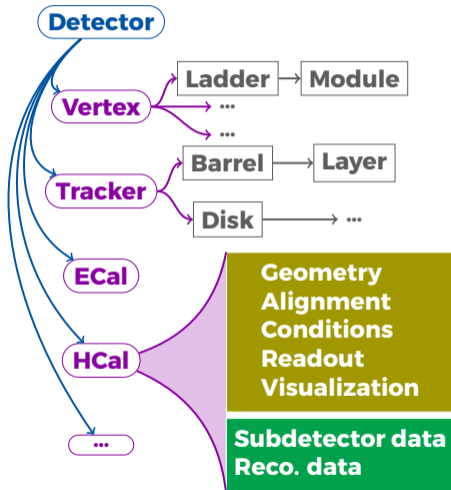
What is DD4hep Detector description?



AIDA²⁰²⁰

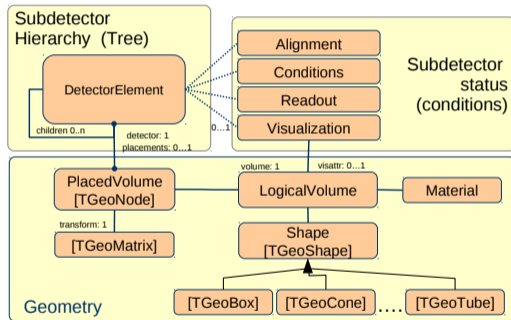


- ▶ Description of a tree-like hierarchy of 'detector elements'
 - ▶ Sub-detectors or parts of subdetectors
- ▶ **Detector Element describes:**
 - ▶ Geometry
 - points to placed logical volumes
 - ▶ Environmental conditions
 - ▶ Properties required to process event data
 - ▶ **Extensions (optionally):** experiment, sub-detector or activity specific data, measurement surfaces. . .



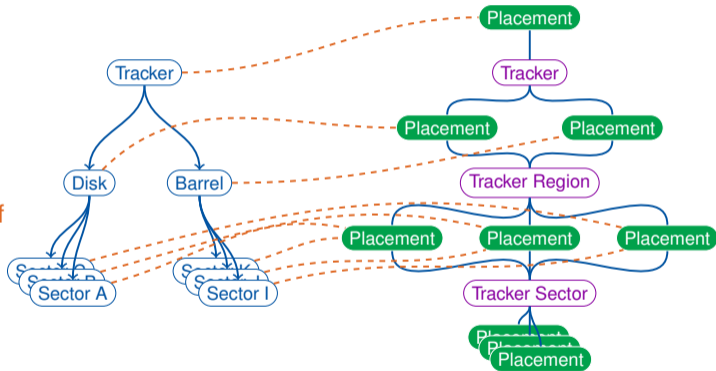
In DD4HEP detectors are a tree of DetElements

- ▶ A DetElement is a sub-detector or a significant part of a sub-detector
- ▶ DetElement points to placed logical volumes
- ▶ Extensions can be attached to DetElements to **provide additional views of the geometry information**





- ▶ Logical Volumes used to build geometrical hierarchy
 - ▶ geometry part delegated to the ROOT classes
- ▶ Relationship between DetElement and placements through full path from top of geometry
- ▶ DetElement tree is fully expanded
 - ▶ DetElement is placed only once in the DetElement tree



- ▶ In-memory translation of geometry from TGeo to Geant4
 - ▶ Materials, Solids, Limit sets, Regions
 - ▶ Logical volumes, placed volumes and physical volumes
- ▶ External configuration:
 - ▶ Plug-in mechanism
 - ▶ Property mechanism to configure plug-in instances
 - ▶ Supports configuration via **XML**, **Python** or **ROOT-AClick**
- ▶ Use plug-in mechanism to configure: Generation, Event Action, Tracking Action, Stepping Action, SensitiveDetector, PhysicsList. . .

Or just use the geometry translation and configure `G4RunManager` by oneself

DDG4 is highly modular

- ▶ Very easily configurable through python (configure actions, filters, sequences, cuts...)

```
#...
part = DDG4.GeneratorAction(kernel, "Geant4ParticleHandler/ParticleHandler")
kernel.generatorAction().adopt(part)
part.SaveProcesses = ['Decay']
part.MinimalKineticEnergy = 1*MeV
part.KeepAllParticles = False
#...
user = DDG4.GeneratorAction(kernel, "Geant4TCUserParticleHandler/UserParticleHandler")
user.TrackingVolume_Zmax = DDG4.tracker_region_zmax
user.TrackingVolume_Rmax = DDG4.tracker_region_rmax
```

- ▶ ddsim python executable is part of the DD4hep release
- ▶ Get steering file `ddsim --dumpSteeringFile > mySteer.py`
 - ▶ Steering file includes documentation for parameters and examples
 - ▶ The python file contains a `DD4hepSimulation` object at global scope
 - ▶ Configure simulation directly from commandline

```

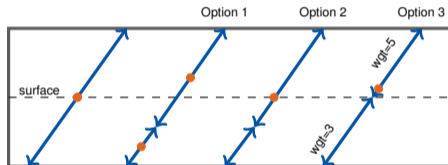
from DDSim.DD4hepSimulation import DD4hepSimulation
from SystemOfUnits import mm, GeV, MeV, keV
SIM = DD4hepSimulation()
SIM.compactFile = "CLIC_o3_v06.xml"
SIM.runType = "batch"
SIM.numberOfEvents = 2
SIM.inputFile = "electrons.HEPEvt"
SIM.part.minimalKineticEnergy = 1*MeV
SIM.filter.filters ['edep3kev'] =
dict (name="EnergyDepositMinimumCut/3keV" ,
      parameter={"Cut" : 3.0*keV} )
    
```

```

$ ddsim
--action.calo                --filter.tracker            --part.keepAllParticles
--action.mapActions          -G                          --part.minimalKineticEnergy
--action.tracker             --gun.direction            --part.printEndTracking
--compactFile                --gun.energy               --part.printStartTracking
--crossingAngleBoost         --gun.isotrop              --part.saveProcesses
--dump                        --gun.multiplicity        --physics.decays
--dumpParameter              --gun.particle             --physics.list
--dumpSteeringFile           --gun.position             --physicsList
--enableDetailedShowerMode   -h                          --physics.rangecut
--enableGun                  --help                     --printLevel
--field.delta_chord           -I                          --random.file
--field.delta_intersection    --inputFiles               --random.luxury
--field.delta_one_step        -M                          --random.replace_gRandom
--field.eps_max               --macroFile                --random.seed
--field.eps_min               -N                          --random.type
--field.equation              --numberOfEvents           --runType
--field.largest_step          -0                          -S
--field.min_chord_step        --outputFile               --skipNEvents
--field.stepper               --output.inputStage        --steeringFile
--filter.calo                 --output.kernel            -v
--filter.filters              --output.part              --vertexOffset
--filter.mapDetFilter         --output.random            --vertexSigma
    
```

- ▶ Providing input handlers, sensitive detectors for most cases...
- ▶ Hard to provide Geant4 Sensitive Detectors for all cases
 - ▶ Couples detector 'construction' to reconstruction, MC truth and Hit production
 - ▶ Too dependent on technology and user needs

e.g. several possibilities
for tracker



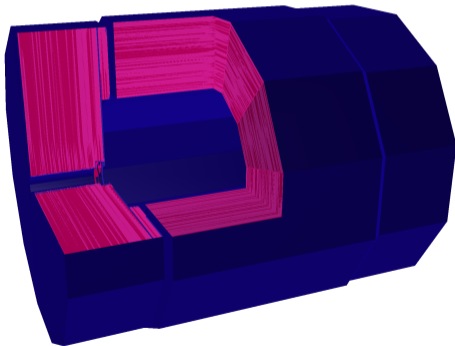
- ▶ Providing palette of most 'common' sensitive components for trackers and calorimeters
- ▶ Physics lists, Physics/particle constructors etc.
 - ▶ Wrapped factory plug-ins directly taken from Geant4
 - ▶ Users extend physics list (e.g. QGSP)
- ▶ Several IO handlers (LCIO, ROOT, StdHep, HepEvt, HepMC)
- ▶ MC truth handling w/ or w/o record reduction

- ▶ XML structure to set parameters for detectors
- ▶ C++ driver to interpret XML parameters and create DetElements and Volumes
 - ▶ Define sensitive parts (attached with SensitiveDetector) and radiator, which has to be known for reconstruction purposes
- ▶ Attach sensitive detector readout, defined elsewhere in XML

```
<readout name="ECB">  
  <segmentation type="CartesianGridXY"  
    grid_size_x="ECal_cell_size"  
    grid_size_y="ECal_cell_size" />  
  <id>...x:32:-16,y:-16 </id>  
</readout>
```

```
<detector  
  name="ECalBarrel"  
  type="GenericCalBarrel_o1_v01"  
  id="42" readout="ECB">  
  <dimensions  
    numsides="ECalBarrel_symmetry"  
    rmin="ECalBarrel_inner_radius"  
    z="ECalBarrel_half_length*2" />  
  <layer repeat="25" >  
    <slice material="Tungsten"  
      thickness="2.40*mm"  
      radiator="yes"/>  
    <slice material="Air"  
      thickness="0.25*mm" />  
    <slice material="Silicon"  
      thickness="0.50*mm"  
      sensitive="yes"/>  
  </layer>  
</detector>
```

- ▶ C++ model of separation of 'data' and 'behaviour'
- ▶ Drivers return single 'reference' to the DetElement object



```
static dd4hep::Ref_t create_element(  
    dd4hep::Detector& description,  
    xml_h element,  
    dd4hep::SensitiveDetector sens) {  
    xml_det_t e = element;  
    DetElement aDetector(e.nameStr(), e.id());  
    //...  
    sens.setType("calorimeter");  
    //...  
    return aDetector;  
}  
DECLARE_DETELEMENT(AName, create_element)
```



- ▶ Combining several sub-detectors to create a detector
- ▶ Define some constants for use in XML
 - ▶ avoid use of global constants in drivers
- ▶ Easy to replace on sub-detector by another

```
<lccdd>
  <info name="CLIC_o3_v14" ...></info>

  <includes>
    <gdmlFile ref="elements.xml"/>
    <gdmlFile ref="materials.xml"/>
  </includes>

  <define>
    <constant name="world_side" value="30000*mm"/>
    <constant name="world_x" value="world_side"/>
    <!-- .... -->
  </define>
  <include ref="Beampipe_o1_v01_02.xml"/>

  <include ref="Vertex_o2_v06_01.xml"/>

  <include ref="InnerTracker_o2_v06_01.xml"/>
  <!-- .... -->

</lccdd>
```

Sharing Drivers

- ▶ DD4HEP's plugin manager can load drivers at runtime
- ▶ Expected XML structure needs to be known to other users
- ▶ Existence of driver needs to be known to other users

Existing Palettes:

- ▶ DD4hep: <https://github.com/AIDAsoft/DD4hep/tree/master/DDDetectors>
- ▶ lcgeo: <https://github.com/iLCSoft/lcgeo/tree/master/detector>
- ▶ FCCSW <https://github.com/HEP-FCC/FCCSW/tree/master/Detector>
- ▶ CEPC?

Place common drivers in DD4hep or different package?

Sharing Detectors



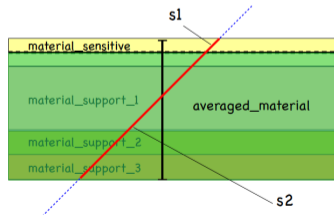
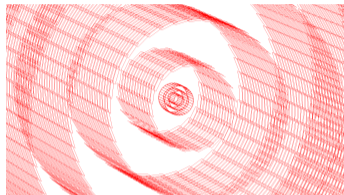
- ▶ Probably very straight-forward for simulation: have to be able to attach sensitive detector of our choice
- ▶ Also want to share reconstruction algorithms, and mix and match geometry drivers (e.g., calorimeters) and reconstruction algorithms (particle flow clustering)
 - ▶ For `iLCSoft` using `Surfaces` and `DDRec::DataStructures` for this purpose
 - ▶ I think these have stronger implications than what is required by simulation
 - ▶ Have to avoid divergence

Information for track reconstruction in iLCSofT

- ▶ measurement directions of hits, local-to-global coordinate transforms, material properties
- ▶ DD4HEP surfaces can be **automatically** added (plugin) or explicitly in the driver

```
<plugin name="DD4hep_GenericSurfaceInstallerPlugin">  
  <argument value="TrackDet"/>  
  <argument value="dimension=2"/>  
  <argument value="u_x=-1."/>  
  <argument value="v_y=-1."/>  
  <argument value="n_z=1."/>  
</plugin>
```

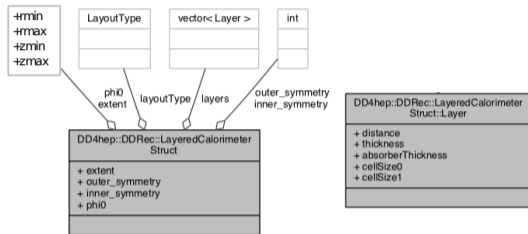
- ▶ Plugin loops over all DetElements of given sub-detector and adds surfaces for sensitive elements
- ▶ Configure surface type and direction in volume
- ▶ Surfaces automatically average materials



High level view onto the detectors through `DDRec::DataStructures` extensions for `DetElements`

- ▶ Drivers fill `DDRec::DataStructures`
 - ▶ Dimensions, positions, parameters for layers,...
- ▶ `DataStructures` allow one to decouple detector implementation from reconstruction algorithms
- ▶ Detector drivers responsible to fill and attach appropriate `DataStructures` to sub-detectors

`DataStructures` contain sufficient information to provide geometry information to particle flow clustering via PandoraPFA



Data Structure	Detector Type
<code>ConicalSupportData</code>	Cones and Tubes
<code>FixedPadSizeTPCData</code>	Cylindrical TPC
<code>LayeredCalorimeterData</code>	Sandwich Calorimeters
<code>ZPlanarData</code>	Planar Silicon Trackers
<code>ZDiskPetalsData</code>	Forward Silicon Trackers

Conclusion

- ▶ **The DD4HEP detector description tool-kit offers a flexible and easy to use solution for the consistent and complete description of particle physics detectors in one single system.**
- ▶ Plugin manager and granular XML configuration of geometry allows sharing of detector drivers
- ▶ Careful consideration for reconstruction also needed