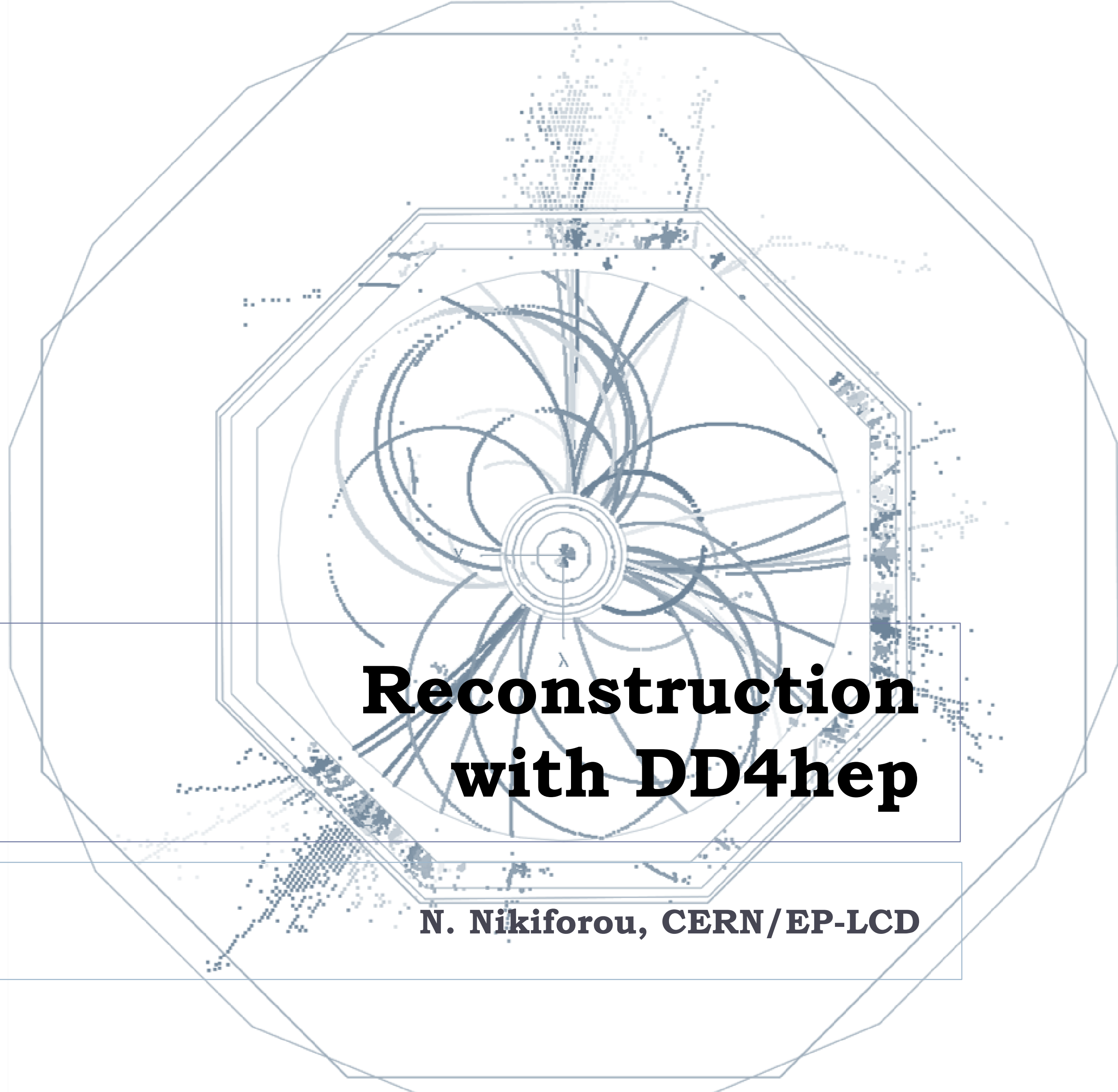
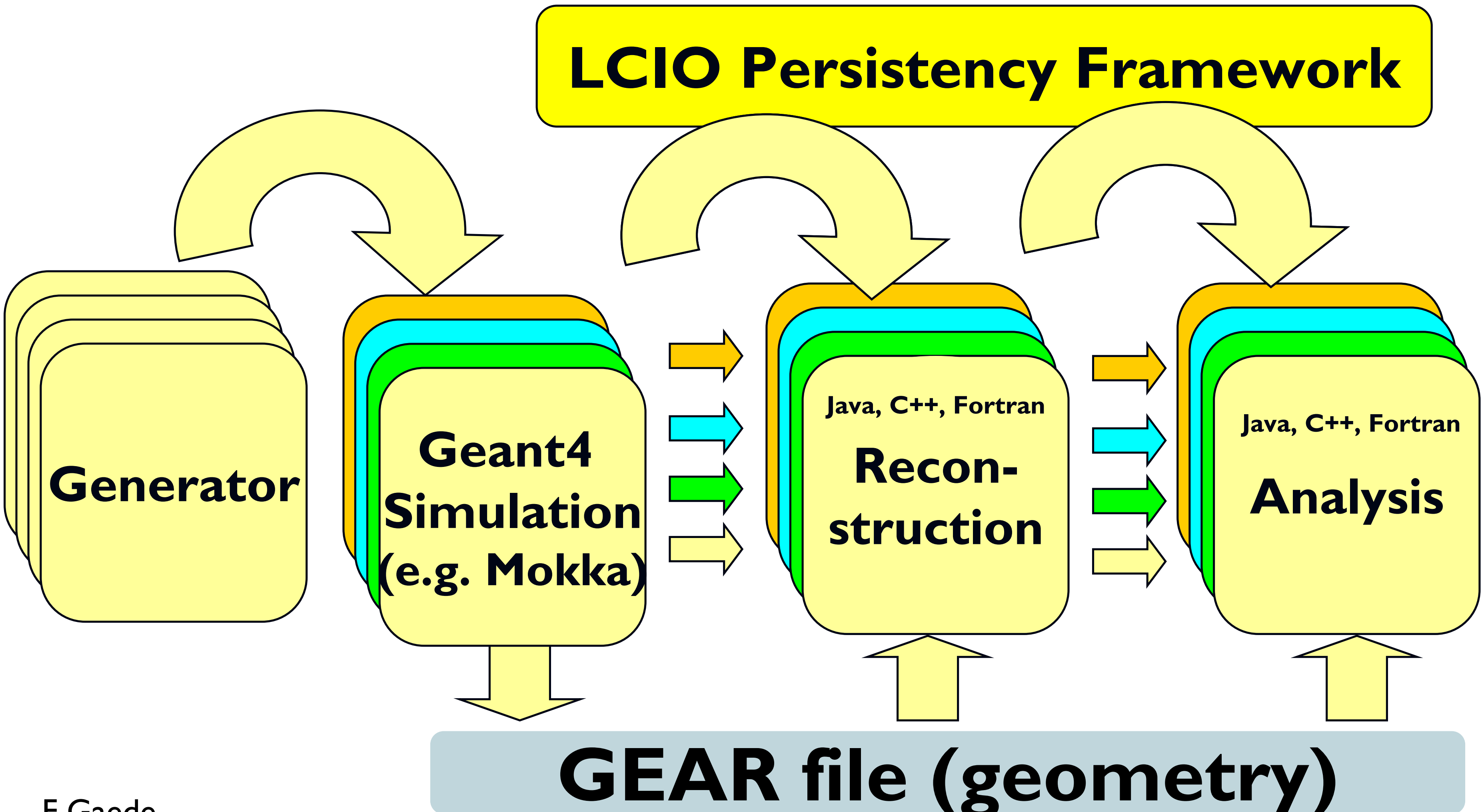


Reconstruction with DD4hep

N. Nikiforou, CERN/EP-LCD

Introduction: chain currently in use



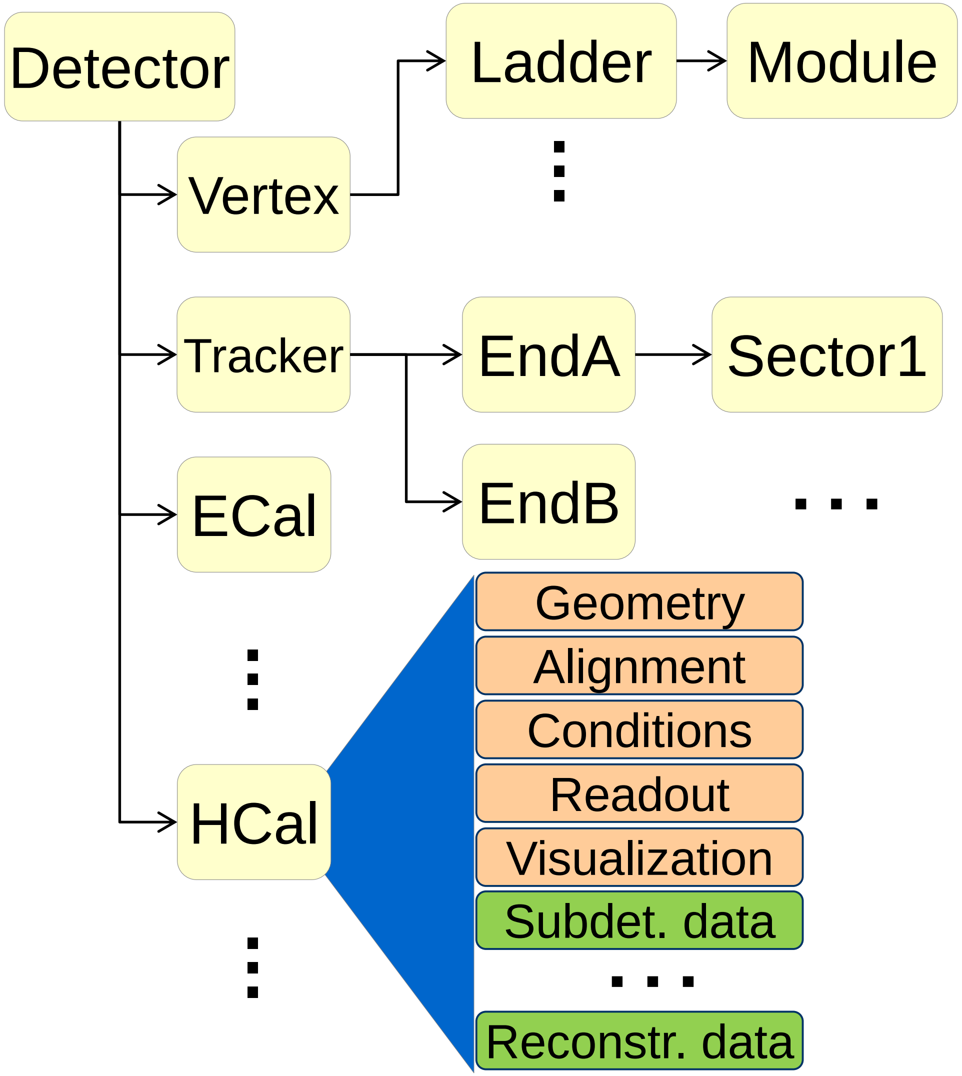
F. Gaede

Introduction

- The GEAR toolkit has served us well over the years
  - Nice, human readable, slimmed-down description of detector geometry
- But tied to ILD geometry and evolution of supported structures is not trivial
  - For a non ILD-type geometry, need “hacks” to create stuctures that GEAR understands
  - Or have to add extra string constants
    - Can explode very quickly
- Always have to pass along information using a gear file from stage to stage in the chain
- We are now building our Simulation and Reconstruction software around **DD4hep**
  - Aims to alleviate some of these problems

Describing a detector in DD4hep

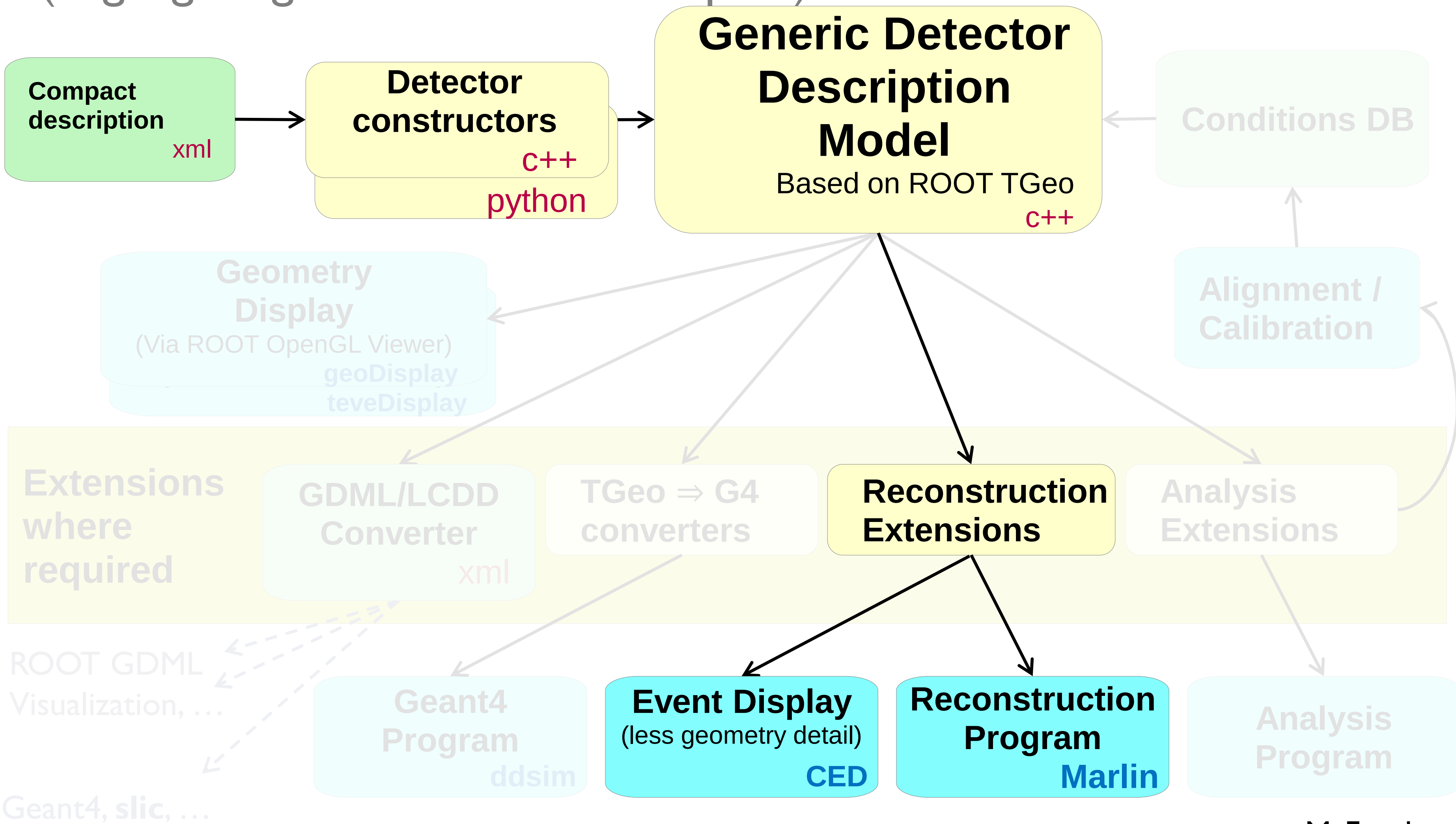
- Description of a tree-like hierarchy of **"detector elements"**
  - Subdetectors or parts of subdetectors
- Detector Element describes
  - Geometry
  - Environmental conditions
  - Properties required to process event data
  - Extensions (optionally): experiment, sub-detector or activity specific data, measurement surfaces, …



M. Frank

DD4hep – The big picture

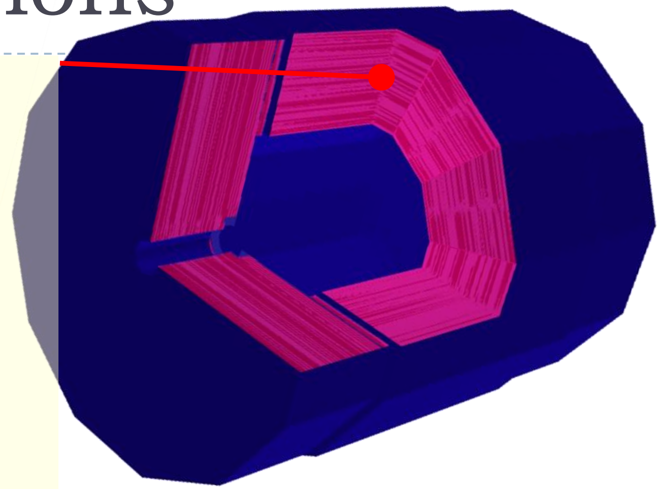
(Highlighting the reconstruction path)



M. Frank

Detector drivers and extensions

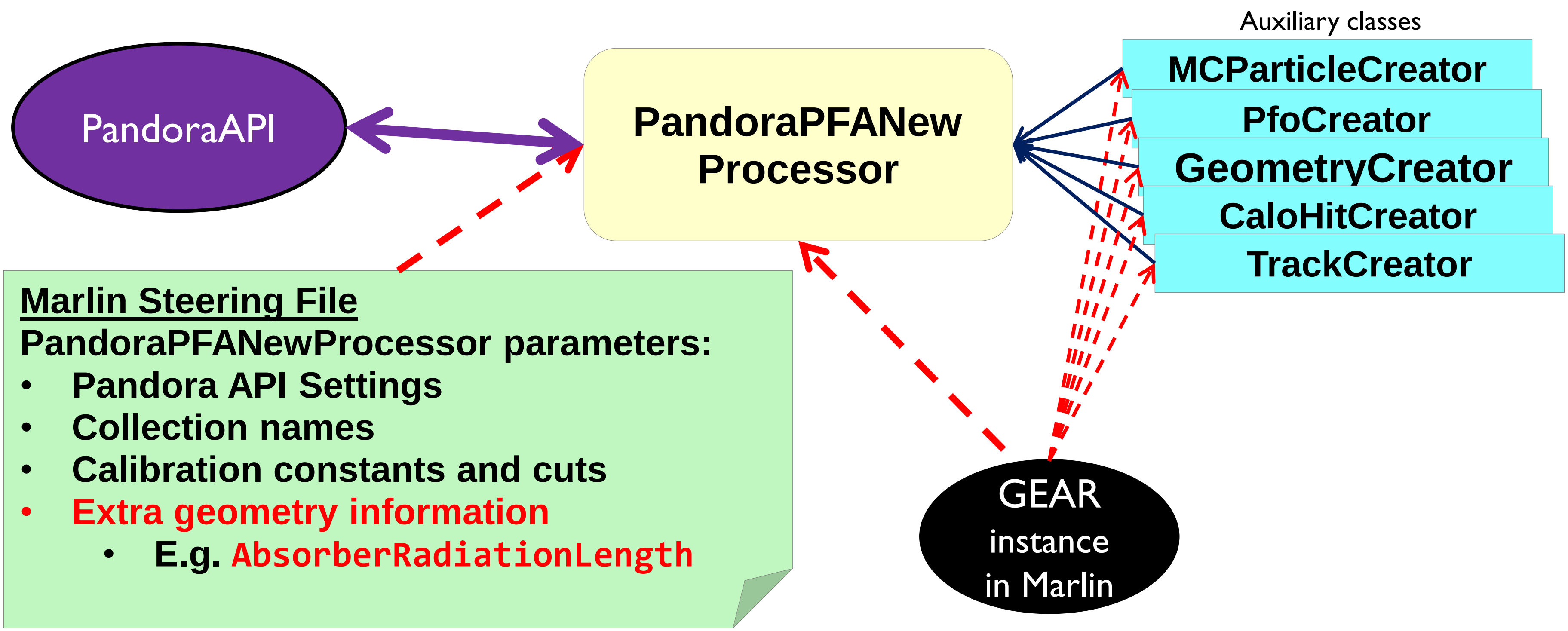
```xml
<detector id="DetID_HCAL_Barrel" name="HCalBarrel" type="HCalBarrel_o1_v01"
readout="HCalBarrelHits" vis="HCALVis" >
 <dimensions nsides="HCal_symm" rmin="HCal_Rin" z="HCal_Z" />
 <layer repeat="(int) HCal_layers" vis="HCalLayerVis" >
  <slice material="Steel235" thickness="0.5*mm" vis="AbsVis"/>
  <slice material="Steel235" thickness="19*mm" vis="AbsVis"/>
  <slice material="Polysterene" thickness="3*mm" sensitive="yes"/>
  <slice material="PCB" thickness="0.7*mm"/>
  <slice material="Steel235" thickness="0.5*mm" vis="AbsVis"/>
  <slice material="Air" thickness="2.7*mm"/>
  </layer>
</detector>
```



- Fairly scalable and flexible drivers (Generic driver palette available)
  - Example C++ code in backup
- Visualization, Radii, Layer/module composition in compact xml
  - Example above
- Volume building in C++ driver
- User decides balance between detail and flexibility
- **Once you have the detector geometry, you can *extend* it, i.e. add more information using the *Reconstruction Extensions*** (more on this later)

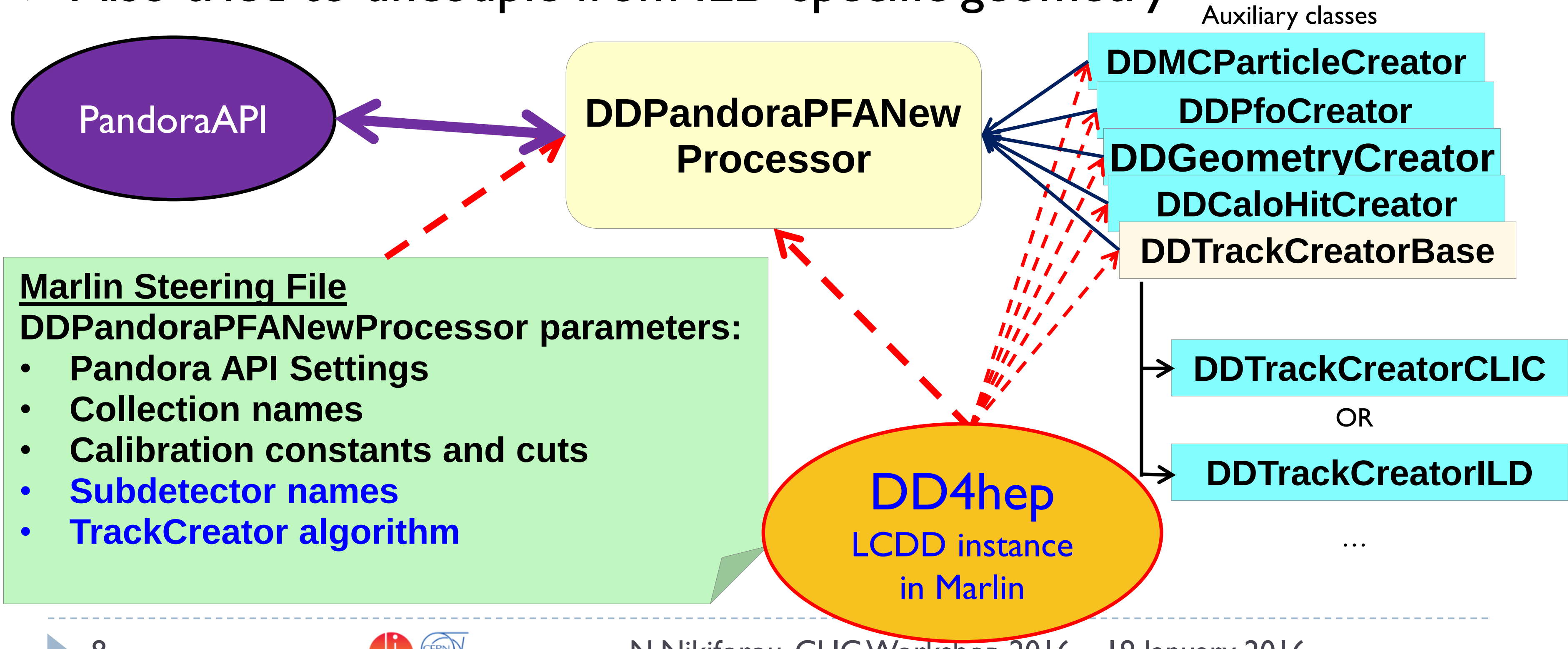
Currently: PandoraPFA and GEAR

- Pandora is the main user of the high-level geometry information provided by GEAR
  - Package **MarlinPandora** translates the GEAR geometry (and LCIO Calorimeter hits/tracks) to the format required by the Pandora API
    - It's also significantly tied to the ILD detector concept



DDMarlinPandora

- New package **DDMarlinPandora**, direct copy of **MarlinPandora**
  - Appended "**DD**" to all class names to avoid clashes
- DD4hep as single source of information
  - No material or other geometry info in processor parameters
- Also tried to uncouple from ILD-specific geometry



But What are "*Reconstruction Extensions*"?

- The user can ***attach*** any object that could help in reconstruction to a DetElement
  - Uses the DD4hep extension mechanism
- We identify a couple of possible options:
  - Objects that directly manipulate the in-memory geometry to **dynamically** calculate quantities requested by the reconstruction algorithms (did not really catch on yet)
  - **GEAR-like simple data structures** that get filled by the detector constructor at creation time (simplest way to start)
  - **Surfaces:** special type of extension foreseen mainly for tracking
    - Kind of a mix of the above: provides static as well as dynamic info

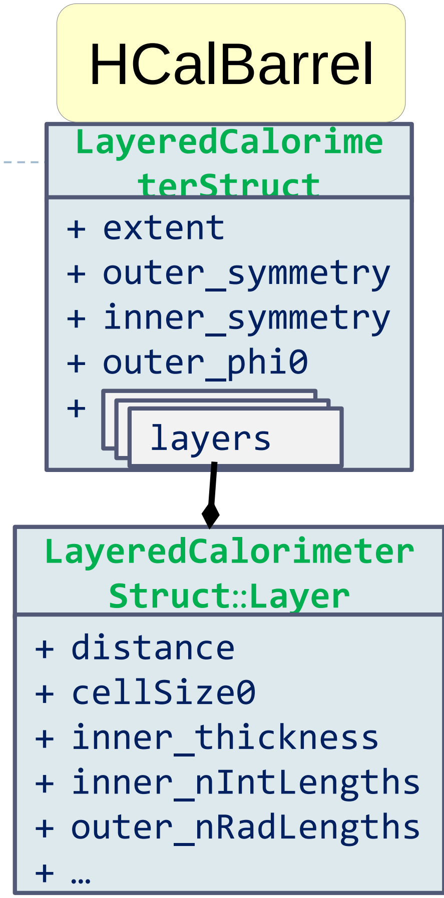
DDRec Data Structures

Extend subdetector driver with arbitrary user data

- Summary of more *abstract* information useful for **reconstruction**
- **Mainly serve** `DDMarlinPandora`, but other use cases:
  - Auxiliary information for **tracking**
    - E.g. "global" information like **number of layers** which you don't want to keep calculating on the fly from surfaces
  - Slimmed-down geometry for a faster event display (e.g. CED)

- Populate during driver construction
  - Driver has access to all the information
  - Take advantage of material map
- **OR** Could even write a ***plugin*** that operates on the subdetector to extend it **without modifying it**
  - Promotes subdetector driver sharing

LayeredCalorimeterStruct



e.g: attach a **LayeredCalorimeterStruct** to the **DetElement** for HCalBarrel

- Developed with needs of Pandora in mind
- Fill all the dimension, symmetry and other info (almost definitely known to the driver)
- Fill a vector of substructures with info on the layers
  - Sum/average material properties from each slice:

```
nRadLengths += slice_thickness/(2*slice_material.radLength());
nIntLengths += slic_thickness/(2*slice_material.intLength());
thickness_sum += slice_thickness/2;
```

- After you are done, add the extension to the detector:

```
sdet.addExtension<DDRec::LayeredCalorimeterData>(caloData);
```

More DDRec Structures

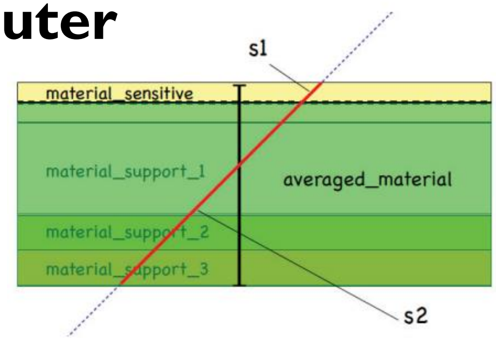
- More *simple* data structures available in `DD4hep/DDRec/DetectorData.h`:
  - `FixedPadSizeTPCData`: Cylindrical TPC with fixed-size pads
  - `ZPlanarData`: Si tracker planes parallel to z
  - `ZDiskPetalsData`: Si tracker disks
  - `ConicalSupport`: e.g. beampipe

- Please consult documentation for conventions on the relevant quantities

Assuming the structures are filled according to the conventions, `DDMarlinPandora` will transparently (and correctly) convert the geometry and initialize `Pandora`

Measurement surfaces

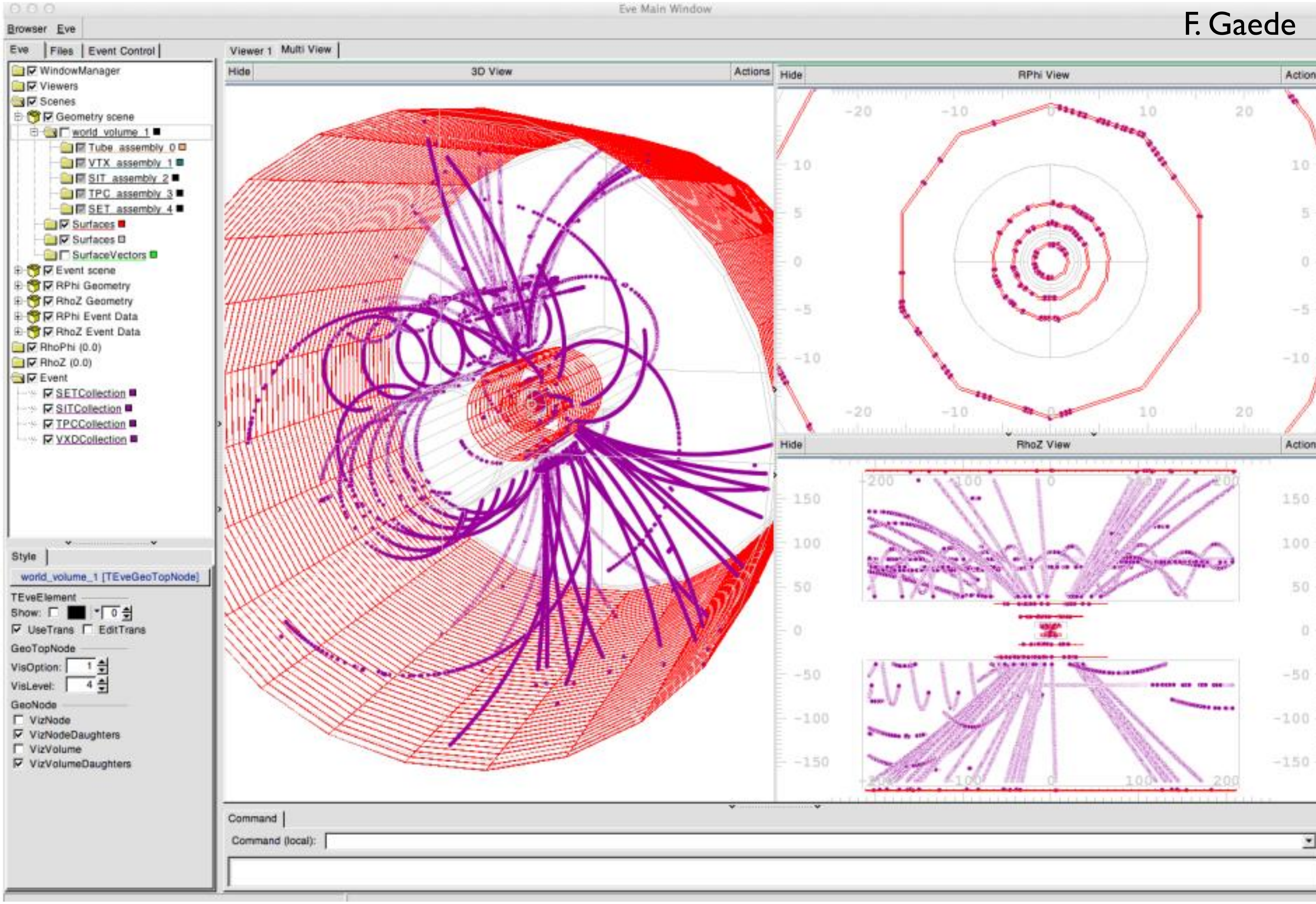
- Special type of extension, used primarily in **tracking**
  - Did not find an implementation in TGeo
  - Implemented in DDRec
- Attached to `DetElements` and `Volumes` (defining their boundaries)
  - Can be added to drivers via **plugins** without modifying detector constructor
- They hold **u**,**v**,**n**ormal and **o**rigin vectors and **inner/outer thicknesses**
- Material properties **averaged automatically**
- Could also be used for **fast simulation**



See talks by Rosa and Daniel

- Outlines of surfaces drawn in `teveDisplay` for CLICdp Vertex Barrel and Spiral Endcaps

Surfaces and Hits in teveDisplay



A word on validation

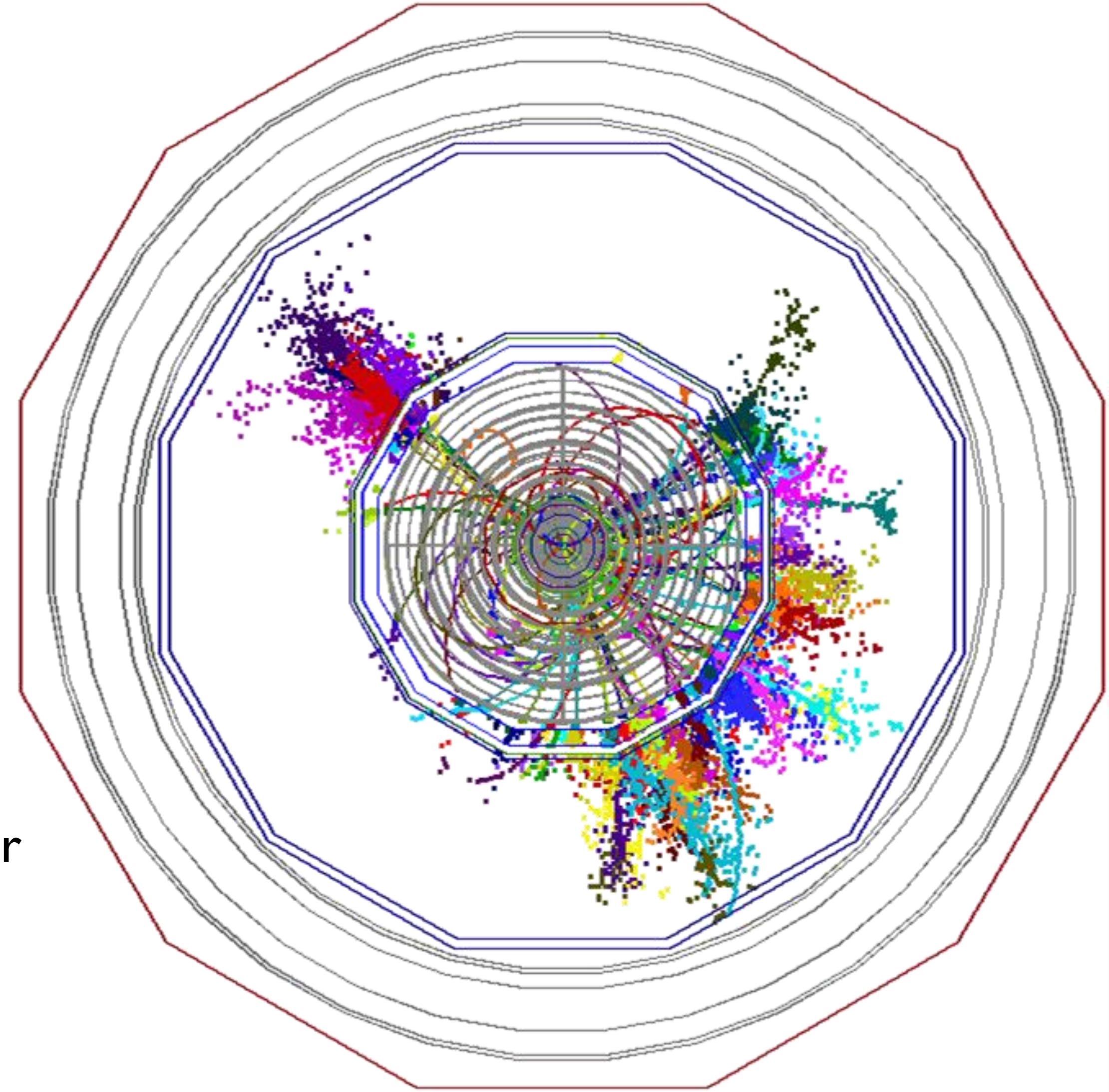
- We are validating the new method against the old one
- One way is to use a very nice monitoring/debug feature of the `Pandora` API: **you can dump the geometry data and the event data as understood by `Pandora`**
  - PandoraGeometry.xml: list of subdetectors with their dimensions, symmetry, layer makeup, etc
  - PandoraEvents.xml: list of events with their CaloHit and Track properties, MCParticles, etc
- Comparing the dumps from `GEAR`+`MarlinPandora` with the ones from `DD4hep`+`DDMarlinPandora` we obtained an almost perfect agreement
- Comparison of performance in physics events ongoing

Reco with the new CLIC detector model

- New detector model being implemented in DD4hep
  - No complete geometry equivalent in older frameworks
- Can't validate against old geometries
  - Rely on ILD validation effort and detailed low-level checks
- `DDTrackCreatorCLIC` is still very basic
  - Cuts and logic need to be optimized as soon as tracker geometry and track reconstruction are stable
- Still some bugs to work out, but already able to fully reasonably reconstruct physics events
- Ported the **calibration procedure** from S. Green to use `DDMarlinPandora`
  - Necessary to set digitization and Pandora constants
    - No other way to obtain constants for new det. model!
  - Working in principle, but not yet ready for production

Event simulated, reconstructed and visualized fully with DD4hep

- **New CLIC detector** model implemented in **DD4hep**
- $Z \rightarrow uds$ event at $\sqrt{s} = 1\ \mathrm{TeV}$ simulated in **DDSim**
- Tracks reconstructed using **DDSurfaces**
- PFOs from **DDMarlinPandora** using the **DDRec** data structures
- Event display from the **CED** viewer interfaced with **DD4hep**
  - Also uses **DDRec** and **DDSurfaces**



Summary

- DD4hep provides consistent single source of detector geometry for simulation, reconstruction, analysis
- ILD and CLICdp are moving to a DD4hep-based reconstruction

- For calorimeter and Particle Flow reconstruction a new package called `DDMarlinPandora` was created
  - Interfaces `Pandora` with geometry provided by `DD4hep`
  - Uses the `DDRec` reconstruction data structures
  - Not tied to a particular detector design

- For tracking: primarily using *surfaces* attached to the detector elements

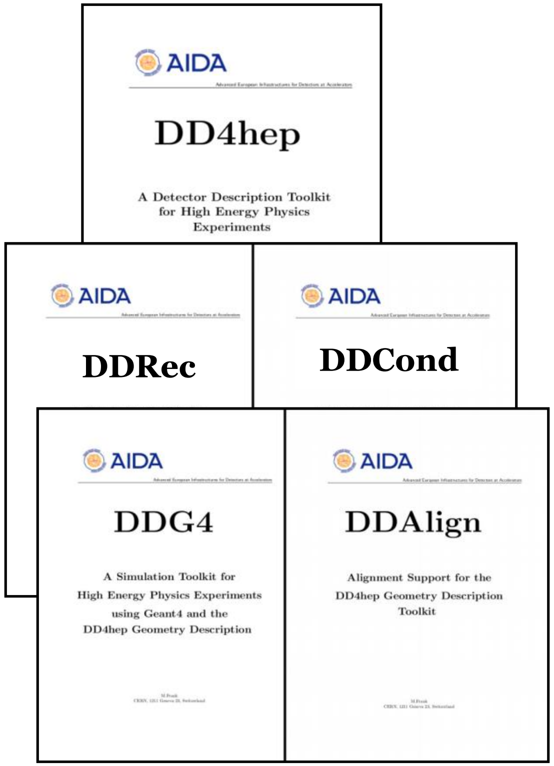
BACKUP SLIDES

DD4hep motivation and goals

- Complete detector description
  - Includes geometry, materials, visualization, readout, alignment, calibration, etc.
- Support full experiment life cycle
  - Detector concept development, detector optimization, construction, operation
  - Easy transition from one phase to the next
- Consistent description, single source of information
  - Use in simulation, reconstruction, analysis, etc.
- Ease of use
- Few places to enter information
- Minimal dependencies

DD4hep components

- **DD4hep**: basics/core
  - Basically stable
- **DDG4**: Simulation using Geant4
  - Validation ongoing
- **DDRec**: Reconstruction support
  - **Driven by LC Community**
  - Covered in this talk
- **DDAlign, DDCond** : Alignment and Conditions support
  - Being developed



- **http://aidasoft.web.cern.ch/DD4hep**

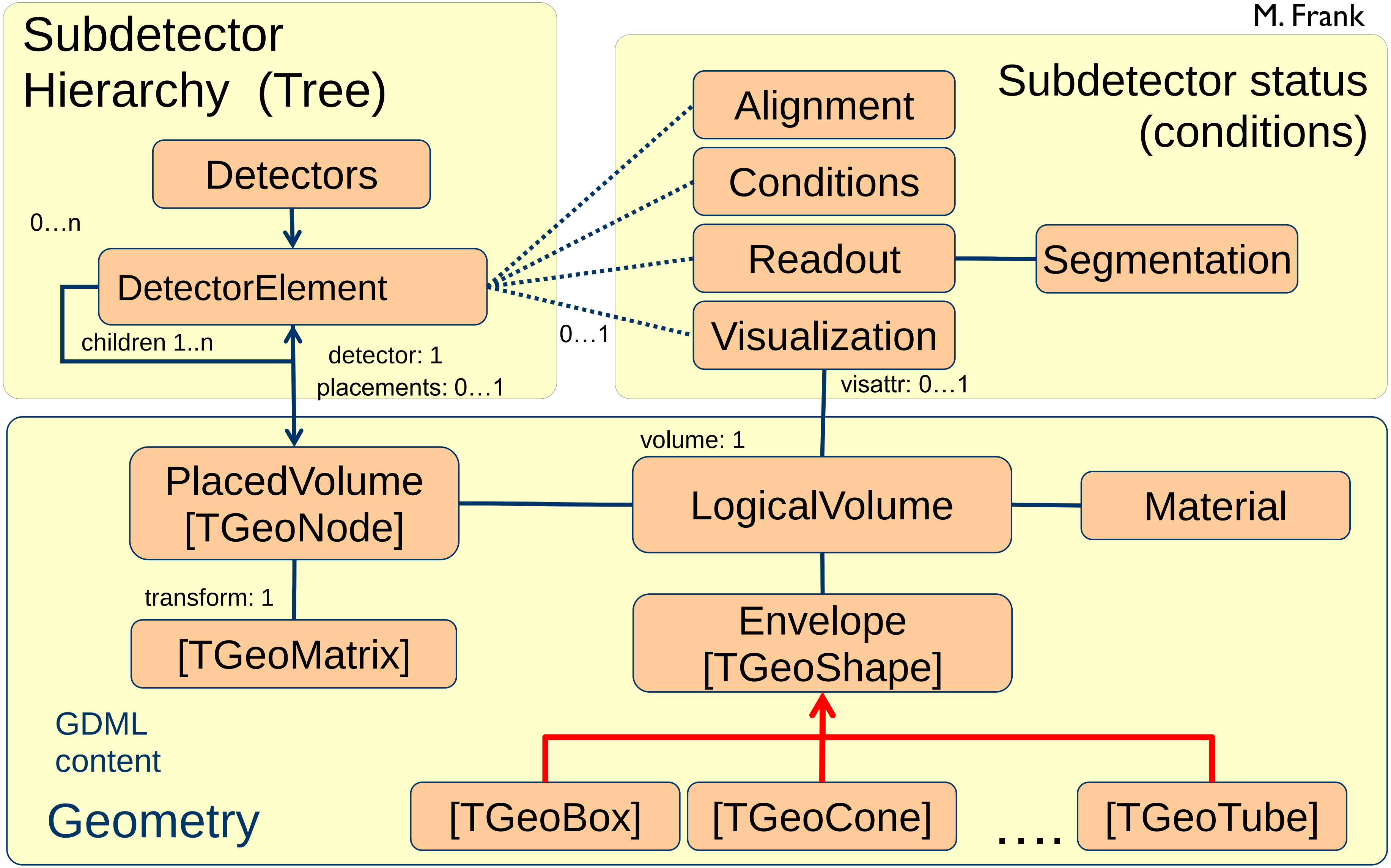
Current DD4hep Toolkit Users

		DD4hep	DDG4
ILD	F. Gaede et al., ported complete model ILD_o1_v05 from previous simulation framework (Mokka)	✓	✓
CLICdp	New detector model being implemented after CDR, geometry under optimization	✓	✓
FCC-eh	P. Kostka et al.	✓	✓
FCC-hh	A. Salzburger et al.	✓	

Feedback from users is invaluable and helps shaping DD4hep!

Geometry Implementation

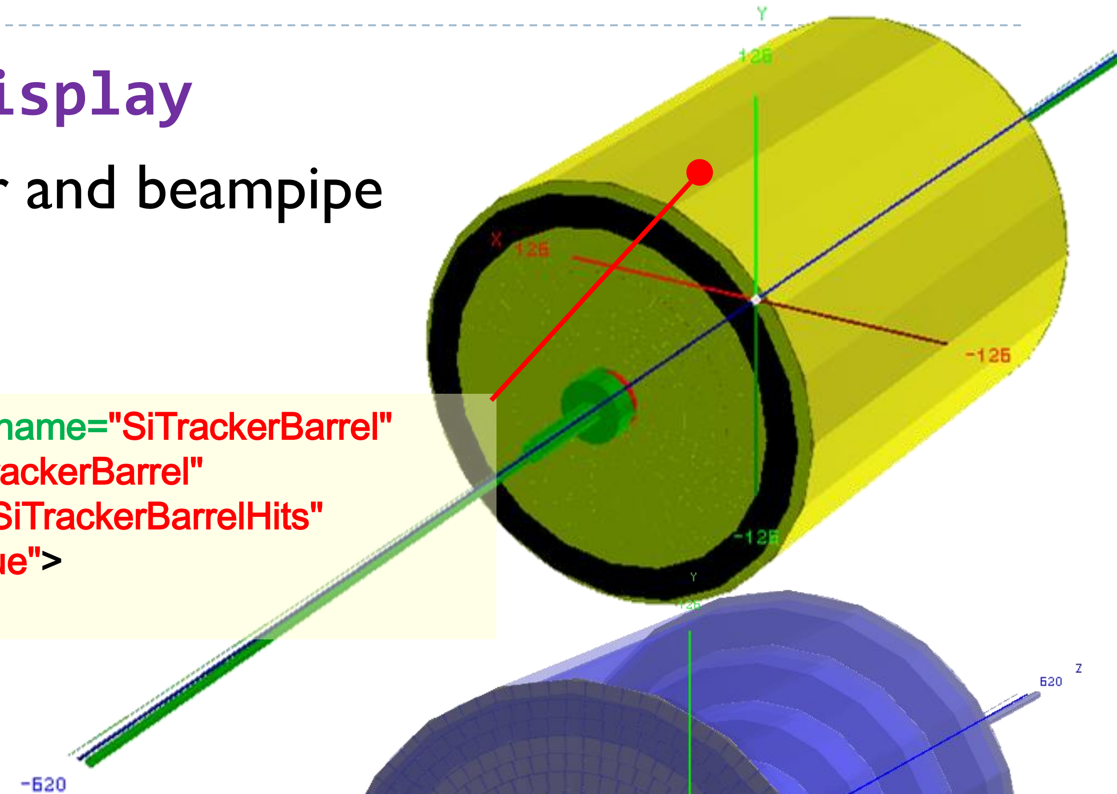
M. Frank

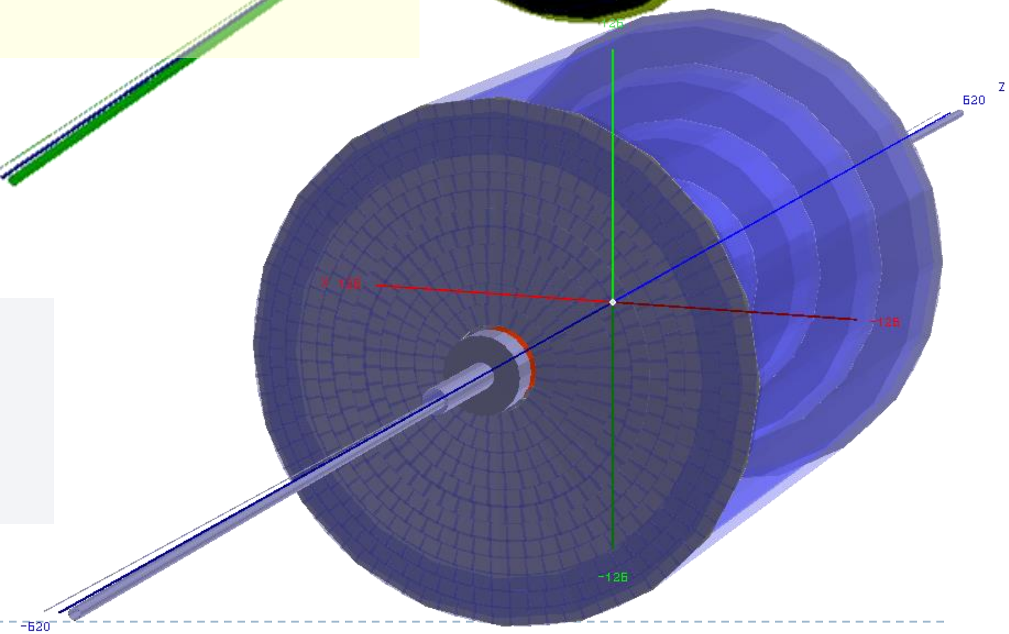


CLIC_SID_CDR Tracker

- Visualized here in **geoDisplay**
- Around Vertex Detector and beampipe

```
<detector name="SiTrackerBarrel"
type="SiTrackerBarrel"
readout="SiTrackerBarrelHits"
reflect="true">
```





The same tracker visualized with ROOT's TGeoManager using and intermediate GDML file dumped from Geant4 after loading geometry from DD4hep

```
for (xml_coll_t c(x_det, _U(layer)); c; ++c) {
  xml_comp_t x_layer = c;
  int repeat = x_layer.repeat();            // Get number of times to repeat this layer.
  const Layer* lay = layering.layer(layer_num - 1); // Get the layer from the layering engine.
  // Loop over repeats for this layer.
  for (int j = 0; j < repeat; j++) {
    string layer_name = _toString(layer_num, "layer%d");
    double layer_thickness = lay->thickness();
    DetElement layer(stave, layer_name, layer_num);
    DDRec::LayeredCalorimeterData::Layer caloLayer ;
     // Layer position in Z within the stave.
    layer_pos_z += layer_thickness / 2;
    // Layer box & volume
    Volume layer_vol(layer_name, Box(layer_dim_x, detZ / 2, layer_thickness / 2), air);

    // Create the slices (sublayers) within the layer.
    double slice_pos_z = -(layer_thickness / 2);
    int slice_number = 1;
    double totalAbsorberThickness=0.;

    for (xml_coll_t k(x_layer, _U(slice)); k; ++k) {
      xml_comp_t x_slice = k;
      string slice_name = _toString(slice_number, "slice%d");
      double slice_thickness = x_slice.thickness();
      Material slice_material = lcdd.material(x_slice.materialStr());
      DetElement slice(layer, slice_name, slice_number);

      slice_pos_z += slice_thickness / 2;
      // Slice volume & box
      Volume slice_vol(slice_name, Box(layer_dim_x, detZ / 2, slice_thickness / 2), slice_material);
       if (x_slice.isSensitive()) {
        sens.setType("calorimeter");
        slice_vol.setSensitiveDetector(sens);
      }
      // Set region, limitset, and vis.
      slice_vol.setAttributes(lcdd, x_slice.regionStr(), x_slice.limitsStr(), x_slice.visStr());
      // slice PlacedVolume
      PlacedVolume slice_phv = layer_vol.placeVolume(slice_vol, Position(0, 0, slice_pos_z));

      slice.setPlacement(slice_phv);
      // Increment Z position for next slice.
      slice_pos_z += slice_thickness / 2;
      // Increment slice number.
      ++slice_number;
    }
```

Example HCal Barrel Driver

- Always within a function called

```
static Ref_t create_detector(LCDD& lcdd, xml_h e, SensitiveDetector sens)
{

…

return sdet;

}
```

- Macro to declare detector constructor at the end:

```
DECLARE_DETELEMENT(HCalBarrel_o1_v01, create_detector)
```

LayeredCalorimeterStruct