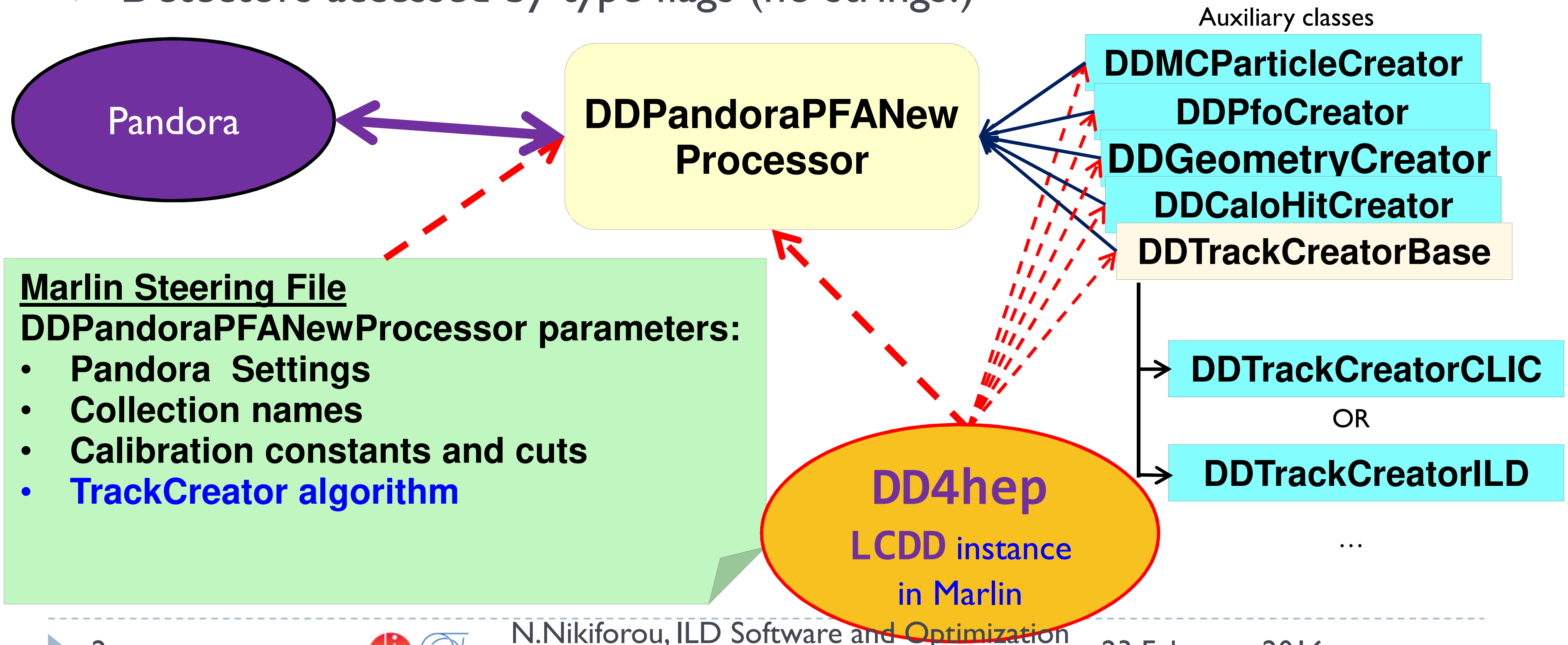
DDMarlinPandora: Status and Validation

N. Nikiforou, CERN/EP-LCD

DDMarlinPandora

- Direct copy of **MarlinPandora + ILDCaloDigi + SimpleMuonDigi**
- **DD4hep** (with **DDRec**) as single source of information
  - No material or other geometry info in processor parameters
- Also tried to uncouple from ILD-specific geometry
  - Detectors accessed by type flags (no strings!)



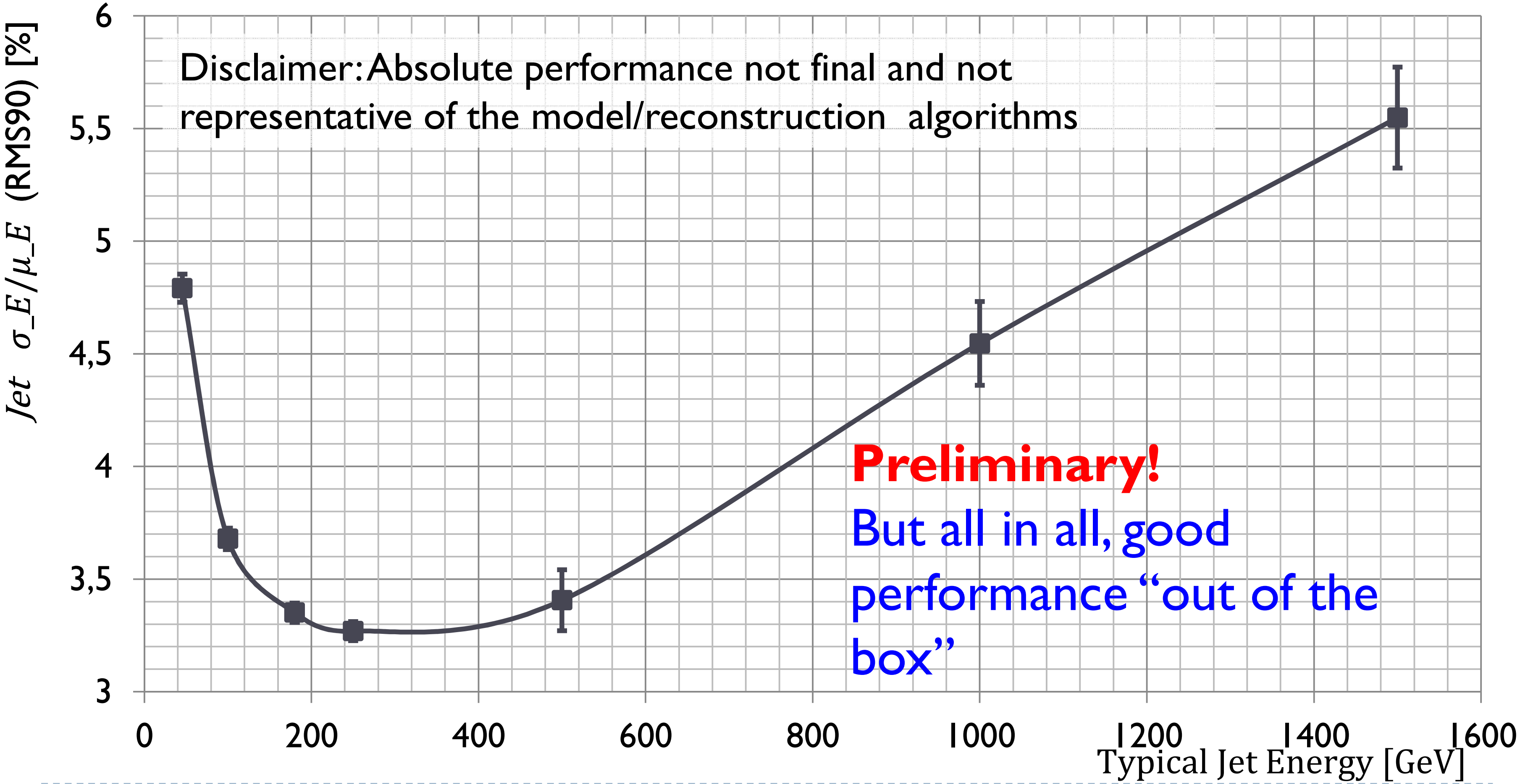
Validation

- 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

- **Ultimate test**: JER and Physics Performance

First look at JER

- `CLICdet_2015` with `DDMarlinPandora` and truth track cheater
- Usual $Z \rightarrow uds$ events, JER estimates from `PandoraAnalysis/AnalysePerformance`



Open Issues

- General cleanup needed, rearrangement of includes, …
- Access by **DetType** flags nice, but gets tiresome
  - Centrally access and serve **DetElements** to sub-classes?
- Auxiliary static functions (e.g. **getExtension(…), getTrackingRegion(), getField(),…**)? => Maybe move to **DD4hep** core?
- Access of "OTHER" subdetectors (e.g. **ECalPlug**)
  - Some algorithms in **LCContent** need ECal inner radius and would then use the "wrong" value, coming from **ECalEndcap**
- **Digitizers**? Ported **ILDCaloDigi** and **SimpleMuonDigi**, shipped with **DDMarlinPandora** for simplicity
  - Only access to ECal (Barrel + Endcap) geometry was needed (?)
    - What about e.g. "**ECalPlug**"? Currently, collections are handled together
- **Calibration**: Almost there (script and steering file)
  - Need to decouple **PandoraAnalysis** (calibration binaries) and **GEAR**
    - **Switch to DD4hep done in private clone, testing needed, possibly cleanup**

BACKUP SLIDES

Pandora Track Creation and Selection

- `DDTrackCreatorILD` is almost identical to old `TrackCreator` in MarlinPandora
  - Already fairly comprehensive but interface to DD4hep allows more flexibility, further optimization and refinement (for the future)
- `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

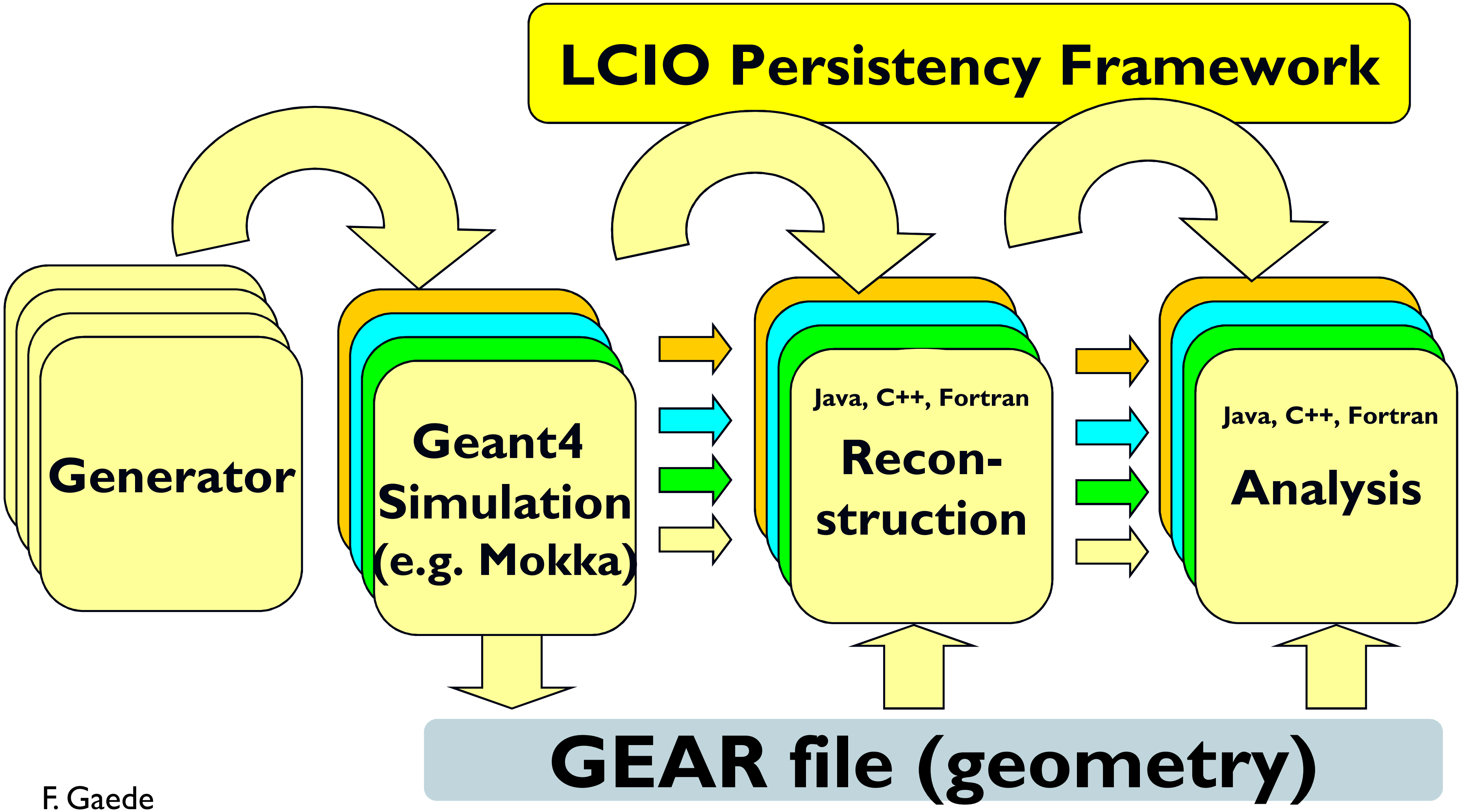
Calibration

- Porting **calibration procedure** from S. Green to use `DDMarlinPandora`
  - Necessary to set digitization and Pandora constants
    - No other way to obtain constants for new CLIC det. model!
  - Working in principle, but not yet ready for production

- Calibration script, Marlin configuration decoupled from `GEAR`, ILD geometry and Cambridge cluster

- Still need to modify binaries in `PandoraAnalysis` (the things that actually compute the constants) to remove use of `GEAR`

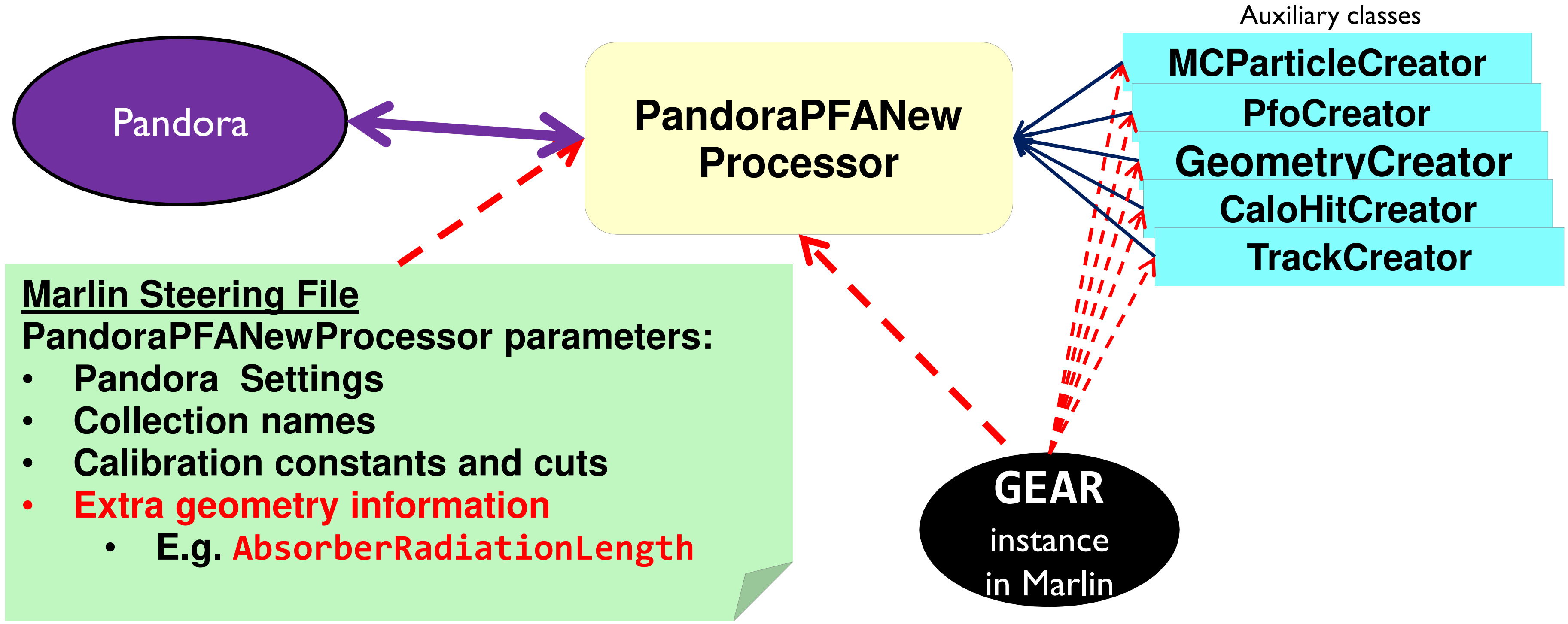
Introduction: chain currently in use



F. Gaede

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 Pandora
    - It's also significantly tied to the ILD detector concept



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 structures 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

```
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

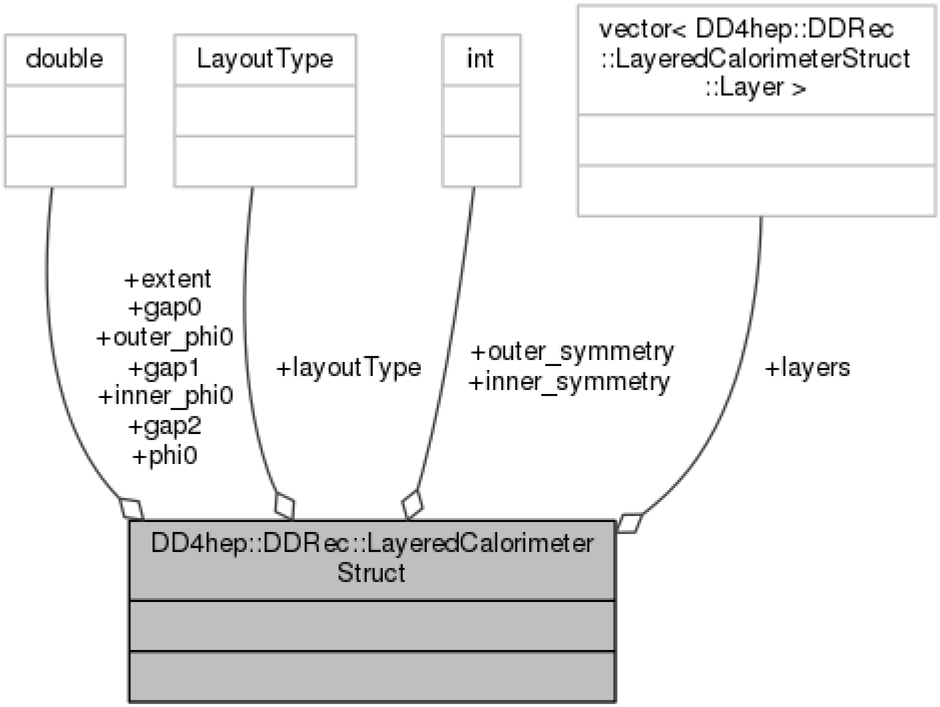
- Always within a function called

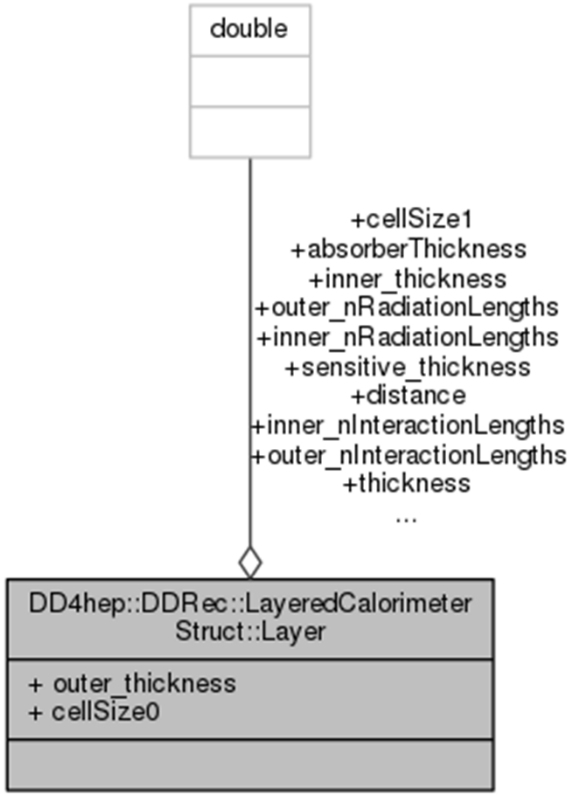
```
static Ref_t
create_detector(LCDD&
lcdd, xml_h e,
SensitiveDetector sens)
{

...

return sdet;

}
```

- Macro to declare detector constructor at the end:

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

LayeredCalorimeterStruct





DRec: Reconstruction Extensions

- The user can ***attach*** any object that could help in reconstruction to a **DetElement** (e.g HCal barrel, ECal endcap, VXD, …)
  - Uses the **DD4hep** extension mechanism
- We currently have two main options:
  - **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
      - Provides static as well as dynamic info
    - Could use "**auxiliary**" **surfaces** (not attached to sensitive volume) in **DDMarlinPandora** in the future
      - Say to determine if a track reaches the calorimeter

DDRec Data Structures

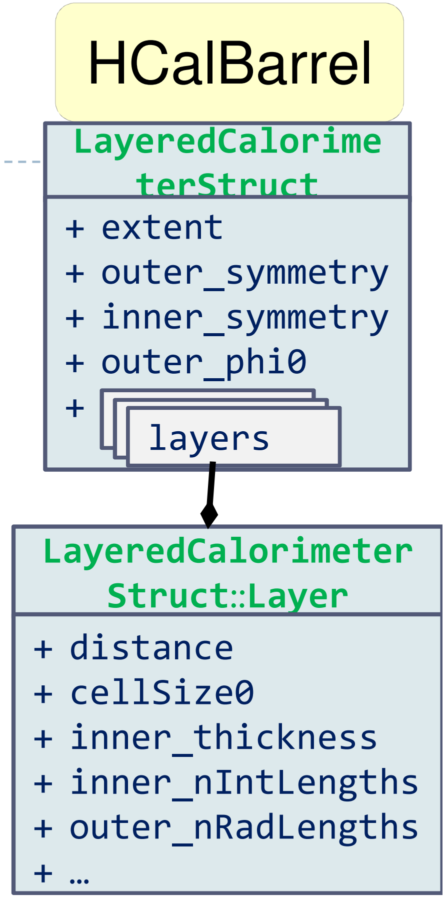
Extend subdetector driver with arbitrary user data

- 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**)

- Current use case: Fill during driver construction
  - Driver has access to all the information
  - Take advantage of material map

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`

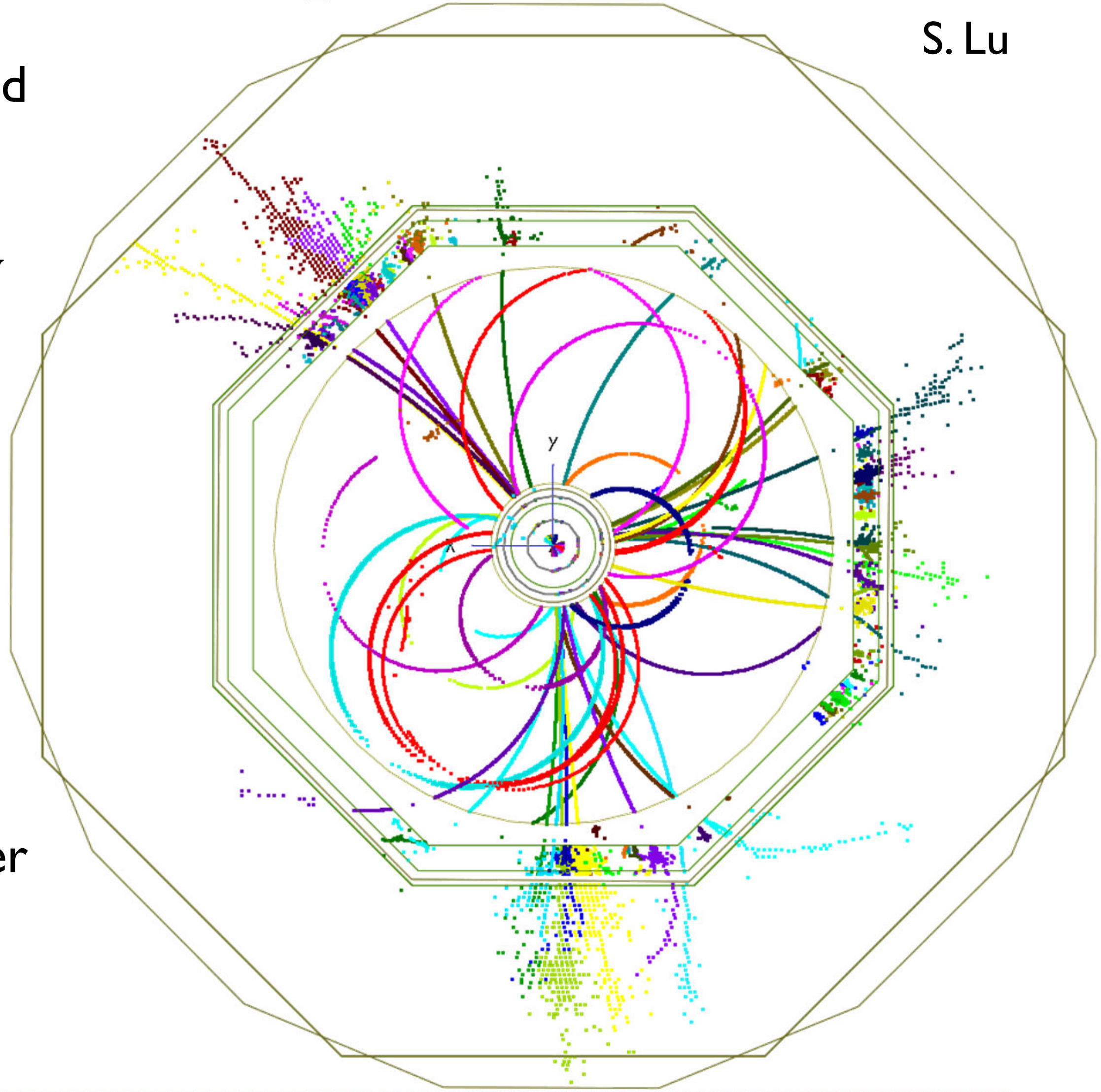
Reco with the available detector models

- **ILD_o1_v05** model implemented in DD4hep (F. Gaede, S. Lu)
- New CLIC detector model evolving
  - No complete geometry equivalent in older frameworks
  - Can't validate against old geometries
- Rely on ILD validation effort and detailed low-level checks

Event simulated, reconstructed and visualized fully with DD4hep

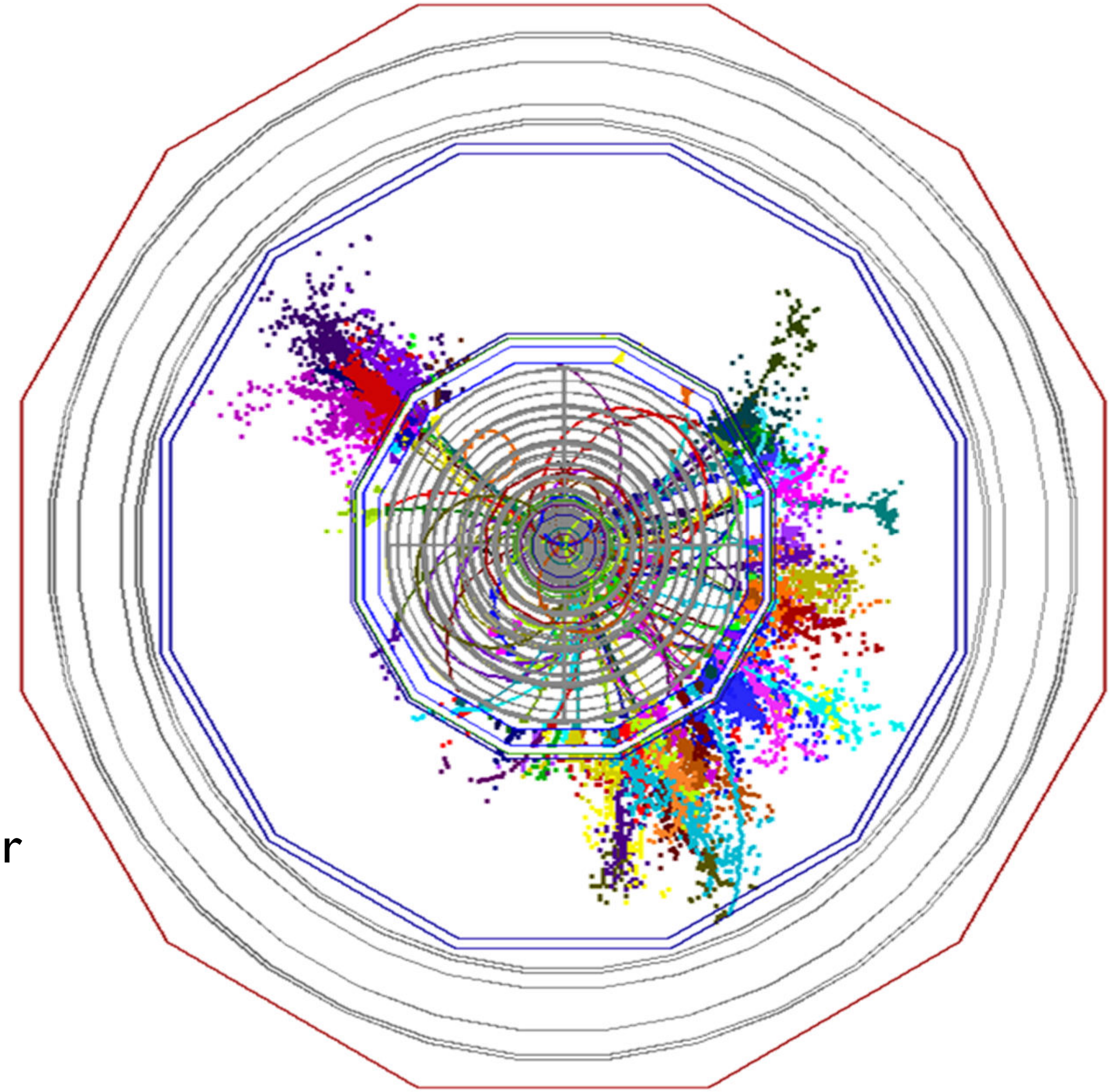
- **ILD_o1_v05** model implemented in **DD4hep**
- $Z \rightarrow uds$ event at $\sqrt{s} = 500$ GeV simulated in **DDSim**
- Tracks reconstructed using **DDSurfaces**
- PFOs from **DDMarlinPandora** using the **DDRec** data structures
- Event display from the **CED** viewer interfaced with **DD4hep**
  - Also uses **DDRec** and **DDSurfaces**

S. Lu



Event simulated, reconstructed and visualized fully with DD4hep

- **New CLIC detector** model implemented in **DD4hep**
- $Z \rightarrow uds$ event at $\sqrt{s} = 1$ 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**



Residuals

The linearity needs some work but this is known and it's not due to DDMarlinPandora itself: Did not apply Non Linearity Corrections