DD4Hep Based Event Reconstruction

A Sailer[1], M Frank[1], F Gaede[2], D Hynds[1], S Lu[2], N Nikiforou[1,3], M Petric[1], R Simoniello[1], G Voutsinas[2,4]

[1] CERN, 1211 Geneva 23, Switzerland
[2] DESY, Notkestraße 85, 22607 Hamburg, Germany
[3] Now at University of Texas at Austin
[4] Now at CERN
On behalf of the CLICdp and ILD collaborations

E-mail: andre.philippe.sailer@cern.ch

Abstract. The DD4Hep detector description toolkit offers a flexible and easy-to-use solution for the consistent and complete description of particle physics detectors in a single system. The sub-component DDRec provides a dedicated interface to the detector geometry as needed for event reconstruction. With DDRec there is no need to define an additional, separate reconstruction geometry as is often done in HEP, but one can transparently extend the existing detailed simulation model to be also used for the reconstruction. Based on the extension mechanism of DD4Hep, DDRec allows one to attach user defined data structures to detector elements at all levels of the geometry hierarchy. These data structures define a high level *view* onto the detectors describing their physical properties, such as measurement layers, point resolutions, and cell sizes. For the purpose of charged particle track reconstruction, dedicated surface objects can be attached to every volume in the detector geometry. These surfaces provide the measurement directions, local-to-global coordinate transformations, and material properties. The material properties, essential for the correct treatment of multiple scattering and energy loss effects in charged particle reconstruction, are automatically averaged from the detailed geometry model along the normal of the surface. Additionally, a generic interface allows the user to query material properties at any given point or between any two points in the detector's world volume. In this paper we will present DDRec and how it is used together with the linear collider tracking software and the particle-flow package PandoraPFA for full event reconstruction of the ILC detector concepts ILD and SiD, and of CLICdp. This flexible tool chain is also well suited for other future accelerator projects such as FCC and CEPC.

1. Introduction

Proper analysis of physics events can only be done with a geometry description that is consistent for data taking, simulation, and reconstruction. Different levels of information are needed for the simulation and the reconstruction of events. Parallel implementations of simulation and reconstruction geometries easily result in systematic discrepancies.

The DD4Hep detector description toolkit [1] aims to solve this problem by providing a single source of geometry. The most detailed geometry is used for the simulation of events. This detailed geometry can be extended by attaching additional information needed for the reconstruction, allowing different views of the geometry depending on the requirements.

The linear collider detector collaborations, CLICdp for the CLIC accelerator, and ILD and SiD for the ILC accelerator, have successfully shared a large part of their software stack for many

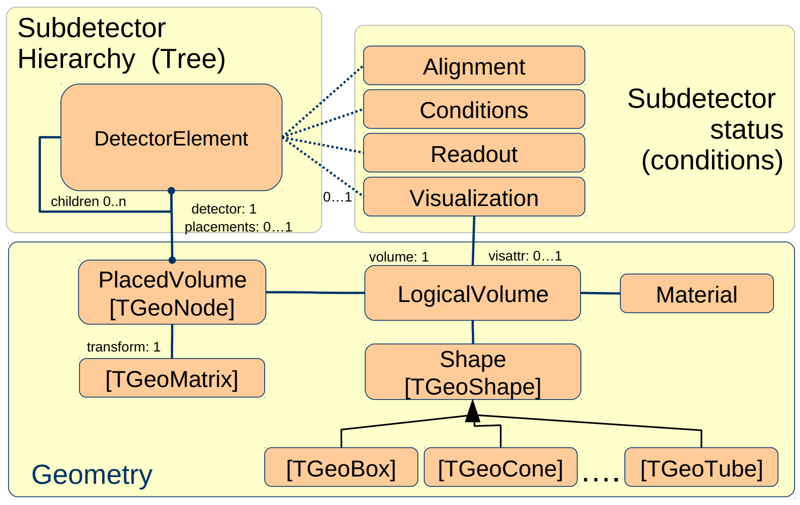

Figure 1. Schema of the detector hierarchy composed of `DetElements`

years. One of the most important parts of this success is the common event data model and persistency format called LCIO [2, 3]. To further increase the synergy from shared software, the DD4Hep geometry package was adopted to replace the legacy geometry descriptions. The reconstruction software is developed to be as generic as possible and thin wrapper *processors* are used to link the reconstruction framework, Marlin [4], with the reconstruction libraries. The following sections describe the DD4Hep toolkit, its extension mechanism, and the reconstruction software based on them.

2. DD4Hep and its Extension Mechanism

In DD4Hep the geometry is described as a *tree* of `DetElements`, where each `DetElement` is a sub-detector or a significant part of a sub-detector. Figure 1 shows the principle of the `DetElement` tree. A `DetElement` object can contain a `PlacedVolume`, which can also contain additional `PlacedVolumes`. The `DetElement` can also contain other `DetElements` as children. To add additional information, for example for alignment or visualisation, extensions are added to each `DetElement`.

The logical volumes that describe the detector geometry are created with the `ROOT::TGeo` framework based on a compact XML description. For simulation, the `TGeo` geometry is converted in memory to a Geant4 geometry [5]. More on the use of DD4Hep-based geometry in simulation is given in another part of these proceedings [6]. Figure 2 shows part of a `C++ detector constructor` for a calorimeter barrel and Figure 3 shows the corresponding compact XML description.

Figure 4 shows how an object is attached as an extension to or accessed from a `DetElement`. There is no limitation to the number of extensions that can be attached, but it is only possible to add one instance of each extension class. It is also possible to attach extensions to a `DetElement` after the `detector constructor` is finished.

2.1. DDRec: Reconstruction Extensions

The high-level view onto the detectors, which is needed by the reconstruction, is given through simple data structures in a DD4Hep sub-package called DDRec. The data structures contain information about the overall dimensions, positions, and parameters for the layers. The `detector constructors` are responsible for filling these data structures, while the reconstruction software accesses the information decoupled from the implementation of the detector. Table 1 gives a list of existing data structures and what types of sub-detectors they

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

  xml_det_t x_det = e;
  Layering layering(x_det);
  xml_comp_t staves = x_det.staves();
  xml_dim_t dim = x_det.dimensions();
  DetElement sdet(det_name, x_det.id());
  Volume motherVol = lcdd.pickMotherVolume(sdet);

  PolyhedraRegular polyhedra(numSides, rmin, rmax, detZ);
  Volume envelopeVol(det_name, polyhedra, air);

  for (xml_coll_t c(x_det, _U(layer)); c; ++c) {
    xml_comp_t x_layer = c;
    int n_repeat = x_layer.repeat();
    const Layer* lay = layering.layer(layer_num - 1);
    for (int j = 0; j < n_repeat; j++) {
      string layer_name = _toString(layer_num, "layer%d");
      double layer_thickness = lay->thickness();
      DetElement layer(stave, layer_name, layer_num);
```

Figure 2. Part of the detector constructor for a sub-detector

```
<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>
```

Figure 3. Compact XML description for a sub-detector

```
//Adding data structure to a DetElement
DetElement caloDet("myCalo", detID);
DDRec::LayeredCalorimeterData* caloData = new DDRec::LayeredCalorimeterData;
// create detector, fill the caloData
caloDet.addExtension< DDRec::LayeredCalorimeterData >( caloData );

//Accessing extension from a DetElement
LCDD& lcdd = LCDD::getInstance();
DetElement caloDet = lcdd.detector("myCalo");
const DDRec::LayeredCalorimeterData *layerData = caloDet.extension< DDRec::LayeredCalorimeterData >();
```

Figure 4. Code listing showing how a DDRec extension is added to or accessed from a DetElement

Table 1. DDRec: Data structures and detector types for which they are used

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

are typically used for. The five – or four if the detector does not contain a Time Projection Chamber (TPC) – data structures are sufficient to describe the general parameters necessary for the reconstruction geometry of the detectors currently envisaged for the linear collider projects. Additional structure can be added as needed in the future.

2.2. Surface Extensions

Track reconstruction requires knowledge of the measurement directions of hits, local-to-global coordinate transformations, and the material properties of the detector, such as radiation length. This information is provided by *surfaces* added to the appropriate parts of the detector. Figure 5 shows an example module that is equipped with a *surface* extension. The four components (`material_sensitive`, and `material_support_1`–3) are separate volumes, which are placed together into a containing volume. The surface is placed in the center of the sensitive volume and the properties of the surface are obtained from averaging the sensitive and the support materials. Such a module could be, for example, a ladder of a vertex detector. It is also necessary to add

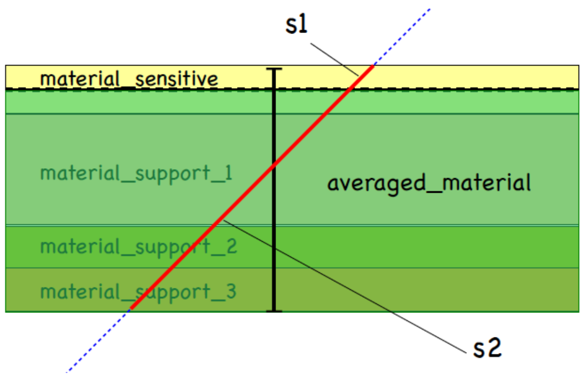

Figure 5. Schematic view of a module that is a part of a tracking detector and how a surface (black dashed line) is placed in the module. `s1` and `s2` are the track segments before and after the surface.

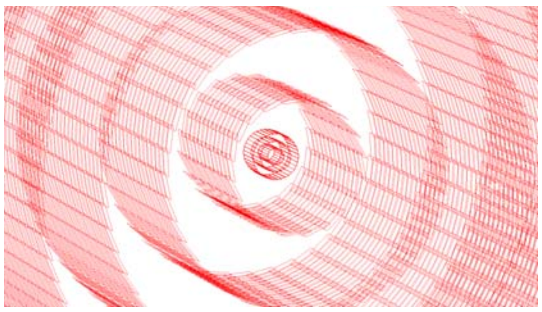

Figure 6. Surfaces for a silicon tracker barrel

surfaces to non-sensitive materials like the beam pipe, or to create helper surfaces independent of volumes, for example just in front of the calorimeter, as required for a Kalman filter and track extrapolation.

To be more flexible and allow surfaces outside of the `detector constructors` the `SurfaceInstallerPlugin` has been created. This plug-in traverses a sub-detector geometry tree and creates a surface object for each `DetElement` that contains a sensitive `DetElement`, similar to the example volumes from Figure 5. The plug-in allows one to specify the normal vector, along which the material is averaged, and the direction vectors `u` or `v` that define the coordinates in the plane of the surface. Figure 6 shows the surfaces automatically created for a silicon barrel tracking detector.

2.3. *Additional Ways to access Geometry Information*

If the information provided by the DDRec or surface extensions is not sufficient, additional possibilities exist to access information about the detector geometry. The function that is used in the calculation of the average material for the surface

```
const MaterialVec& materialsBetween ( const Vector3D& p0, const Vector3D& p1 )
```

can also be used to obtain the material between any arbitrary points `p0` and `p1`. One can also access the `DetElements` and query the position of the placed volumes or their transformation matrices.

3. Reconstruction Software Based on DD4Hep

3.1. *Track Reconstruction*

Pattern recognition and track finding algorithms can be optimised based on the features of specific tracking detectors. A gaseous TPC, with many hits left by charged particles, offers a different problem than the few hits left in the layers of a silicon-based detector. As a result, a range of pattern recognition algorithms exist in the linear collider software framework. All of these different approaches use the surfaces and extensions created in DDRec. Figure 7 shows the relationship between the pattern recognition and track-fitting algorithms in the linear collider software.

Clupatra is used for pattern recognition in a TPC where it is possible to use a nearest-neighbor-like approach of finding tracks. Figure 8 shows the clustering of hits into tracks.

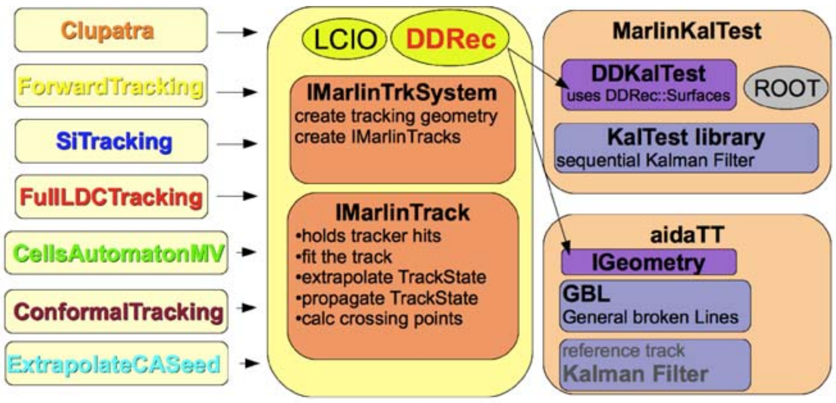

Figure 7. Track reconstruction packages in the linear collider software framework

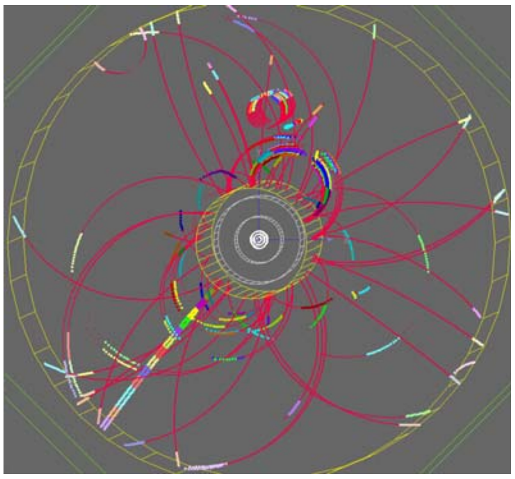

Figure 8. Pattern recognition in a TPC with the *Clupatra* package

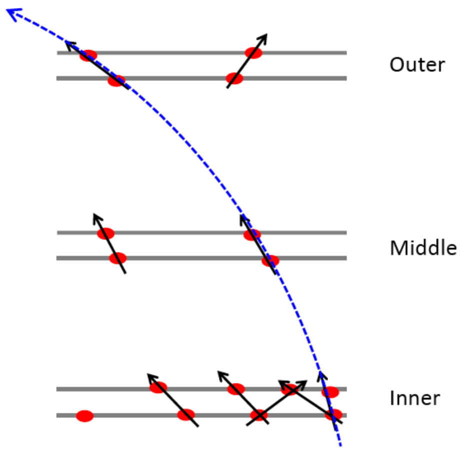

Figure 9. Track finding based on mini-vectors created in a *double-layered* vertex detector

Another pattern recognition algorithm is based on the idea of a *double-layer* layout of the vertex detector. Two hits in adjacent layers of a double-layer can be turned into a *mini-vector*, whose direction helps in the extrapolation towards the next layer, as shown in Figure 9. A special algorithm has also been developed for the reconstruction in forward silicon disks. This approach is based on a cellular automaton [7]. See Figure 10 for an example of the hits and cells considered during the execution of this algorithm.

Another approach is based on a conformal mapping of the hits within the silicon vertex and tracking detectors. Each hit position in the plane perpendicular to the magnetic solenoid field (x, y) is transformed into conformal space $(u = x/(x^2+y^2), v = y/(x^2+y^2))$, with the effect that helix projections (circles) are mapped on to straight lines (Figure 11). The pattern recognition is thus reduced to a straight line search, and is performed in a geometry-agnostic way, considering only the hit positions in global space.

The track fitting can be done with a Kalman filter, such as implemented in the DDKalTest package, or via the *General Broken Lines* approach in the aidaTT package, which offers a connection to the Millepede alignment tool [8].

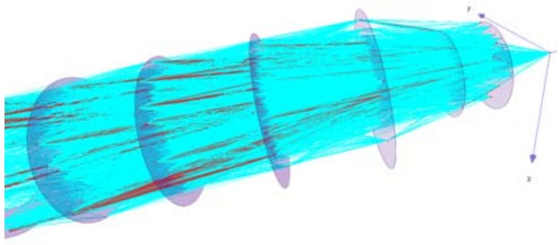
Figure 10. Potential track candidates in the cellular automaton used for the forward tracking algorithm

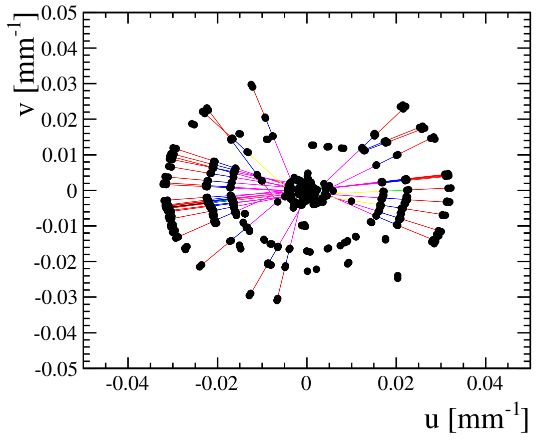

Figure 11. Hits transposed into conformal space and the straight lines connecting track candidates.

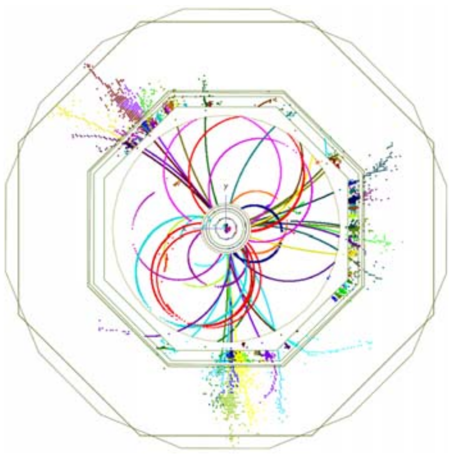
Figure 12. t $\bar{\text{t}}$ event reconstructed in the ILD detector using a TPC and silicon tracking

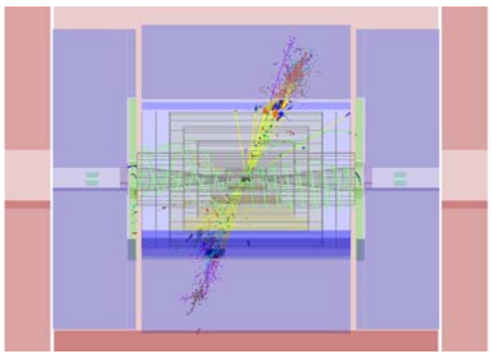
Figure 13. Z to two jets event reconstructed with the CLIC detector using only silicon tracking

3.2. Particle Flow Clustering
Particle flow clustering in the event reconstruction is delegated to PandoraPFA [9, 10]. PandoraPFA is a toolkit for pattern recognition algorithms in highly granular calorimeters, originally developed for the ILC and CLIC detectors. It has now been extended to solve pattern recognition problems in the LAr-TPC reconstruction for the DUNE neutrino experiment and to include functionality to support particle flow reconstruction in *coarse* calorimeters. The `DDMarlinPandora` processor glues the event data model LCIO, the geometry package DD4Hep, and PandoraPFA together. `DDMarlinPandora` passes the geometry information, tracks, and calorimeter hits into the PandoraPFA framework, and later converts the PandoraPFA objects back into the LCIO format.

3.3. Examples
Figures 12 and 13 show events simulated and reconstructed with the same software framework, but in different detector models.

4. Summary

The DD4Hep detector description toolkit offers a flexible and easy-to-use solution for the consistent and complete description of particle physics detectors. With the extension mechanism, which allows one to attach derived information to existing volumes, there is no need to define a separate geometry for reconstruction. This means no duplication of effort and fewer bugs.

The reconstruction software for tracks and particle flow reconstruction as well as the DD4Hep geometry framework are now used by the CLICdp, ILD, and SiD collaborations. Only thin wrappers are needed between the experiment framework and geometry on one side and the reconstruction packages on the other. This generic approach allows other users to describe their detectors via DD4Hep and use the existing reconstruction software.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168.

References

[1] Frank M, Gaede F, Grefe C and Mato P 2013 *J. Phys. Conf. Ser.* **513** 022010
[2] Gaede F, Behnke T, Graf N and Johnson T 2003 LCIO — A persistency framework for linear collider simulation studies *CHEP 2003* (La Jolla, California)
[3] Gaede F, Behnke T, Cassell R, Graf N, Johnson T and Vogt H 2004 LCIO persistency and data model for LC simulation and reconstruction *CHEP 2004* (Interlaken, Switzerland)
[4] Gaede F 2006 *Nucl. Instrum. Meth.* **A559** 177–80
[5] Frank M, Gaede F, Nikiforou N, Petric M and Sailer A 2015 *J. Phys. Conf. Ser.* **664** 072017
[6] Petric M, Frank M, Gaede F, Lu S, Nikiforou N and Sailer A 2016 *J. Phys. Conf. Ser.* **These Proceedings**
[7] Gaede F, Aplin S, Glattauer R, Rosemann C and Voutsinas G 2014 *J. Phys.: Conf. Ser.* **513** 022011
[8] Kleinwort C 2012 *Nucl. Instrum. Meth.* **A673** 107–10
[9] Marshall J S and Thomson M A 2012 *J. Phys. Conf. Ser.* **396** 022034
[10] Marshall J S and Thomson M A 2015 *Eur. Phys. J.* **C75** 439