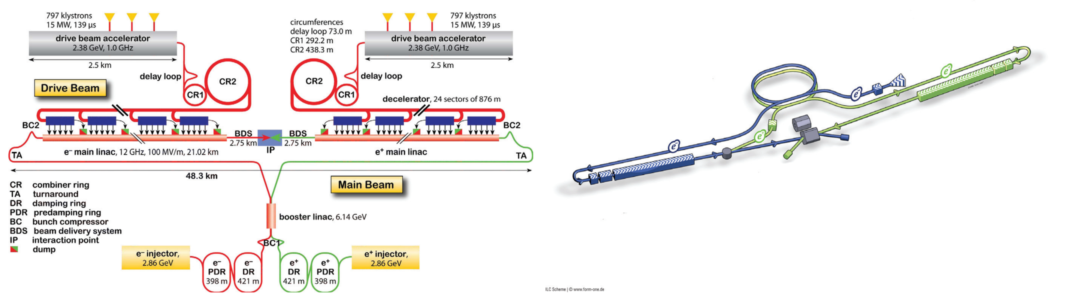
Integration of DD4hep in the Linear Collider Software Framework

Markus Frank[1] Frank Gaede[1,2] Shaojun Lu[2] Nikiforos Nikiforou[1] Marko Petric[1] André Sailer[1]

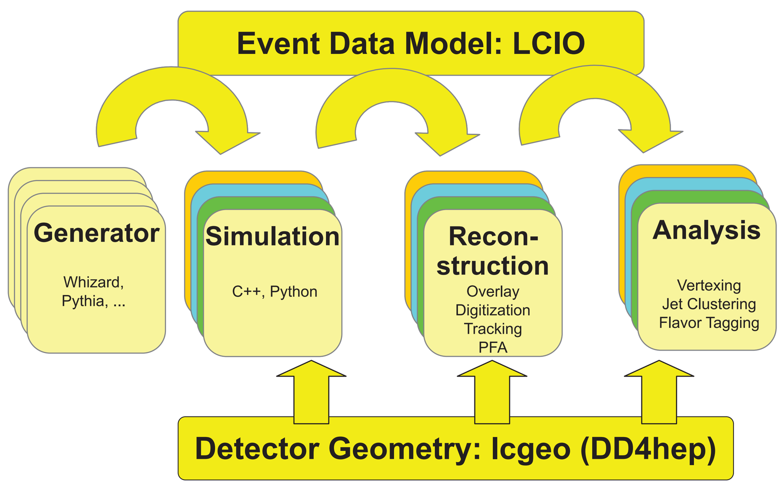
[1]CERN, [2]DESY

Overview

Two large, high-energy linear lepton colliders, CLIC and ILC, are currently proposed



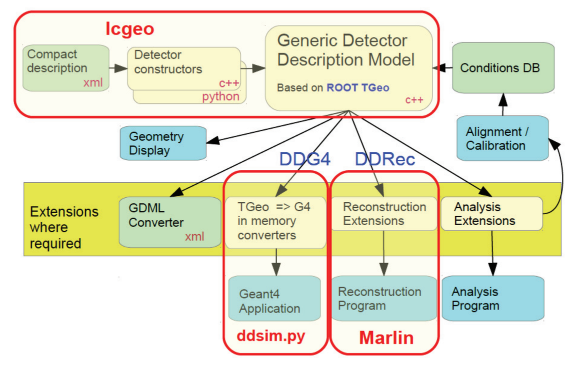
- Three detector concepts are under development for these machines
- For the detector optimisation and physics studies detailed and realistic Monte Carlo simulations are performed
  - Enabled through flexible geometry description and reconstruction software
- Software is **shared** by the detector concepts **across colliders** and hardware R&D groups
- Make use of a common *event data model*: **lcio**
- Second leg of common software, *geometry description*: **DD4hep**

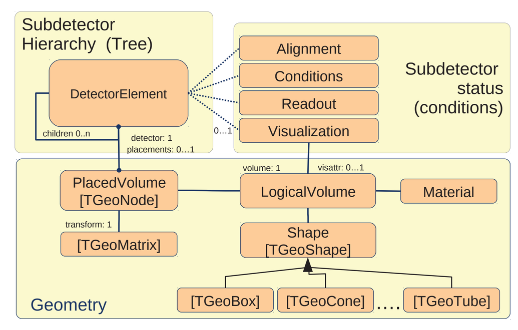


DD4hep and lcgeo

The **DD4hep** detector description toolkit offers a flexible and easy to use solution for the **consistent** and **complete description** of particle physics detectors in one single system. It provides software components addressing visualisation, simulation, reconstruction, and analysis of high energy physics data.

- Detailed geometry implemented with `Root::TGeo`
  - `Detector constructors`
- A detector model is a tree of `DetElements`
- Functionality is **extensible** through plug-ins
- Additional user defined information can be attached to `DetElements` (e.g.: alignment, conditions, readout)

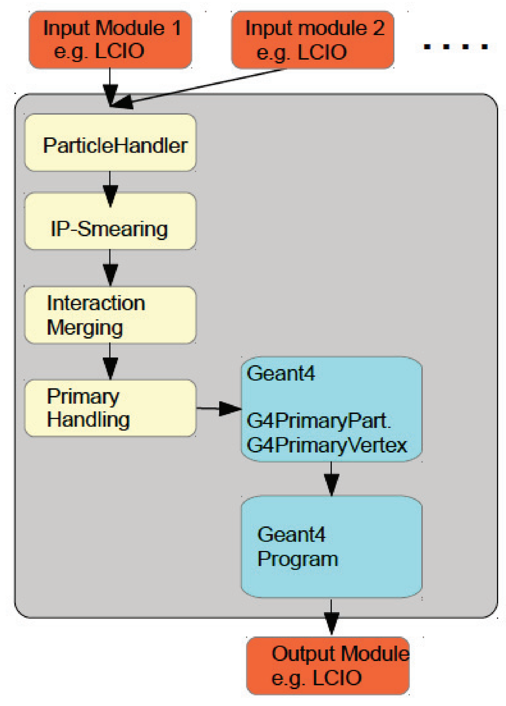


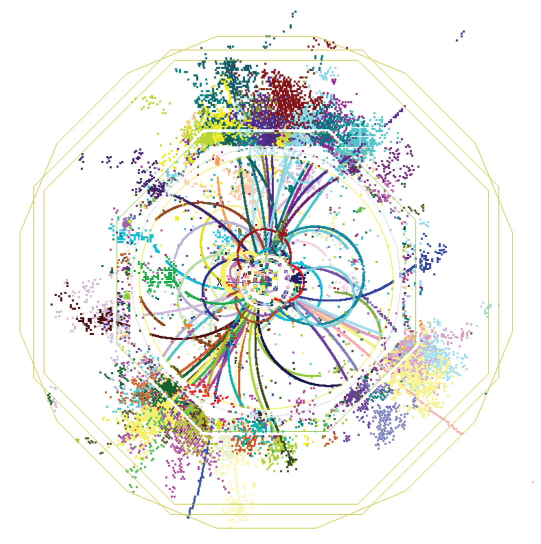


- The **lcgeo** (Linear Collider Geometry) package is the collection of **flexible** but **detailed** detector constructors for linear collider detector models and test beam activities
- The complete detector models follow a **hierarchical** structure
- Detailed detector description but flexible enough to allow for scaling of the detector dimensions and for sharing between concepts

Simulation&DDG4

- The simulation uses DDG4 the **DD4hep built-in** gateway to `Geant4` (Talk by M.Frank: Track 2, Apr. 16, 11:45)
- The **lcio** input and output converters read and write event data for the linear collider studies
- Overlay of events and boosting or smearing of primary vertex
- Customised sensitive detectors and readout segmentation can be assigned to the sub-detectors
- **Detailed Monte Carlo history** is provided to understand the contributions from primary particles to individual hits in the trackers and calorimeters
- Python based interface to control the simulation: `ddsim.py`





Simulated event $e^+e^- \to t\bar{t}$ at $\sqrt{s} = 500$ GeV: colouring based on MC history

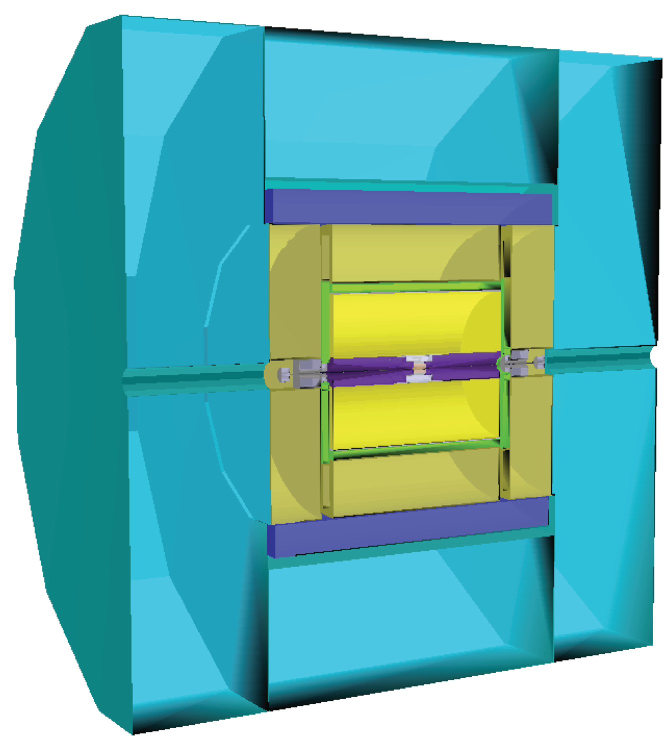
Detector Models

- Detector models are combinations of the sub-detector drivers in **lcgeo**
- Sub-detector constructors are shared between detector concepts
  - Stable and generic detector constructors can be integrated into the **DD4hep** detector library: `DDDetectors`
- Each sub-detector is contained in an **Envelope** volume defining its boundaries
  - Envelopes described with high-level parameters, e.g., inner and outer radius
  - Envelopes can be completely described in the XML file
  - Can build detector with envelopes only for a **simplified view**

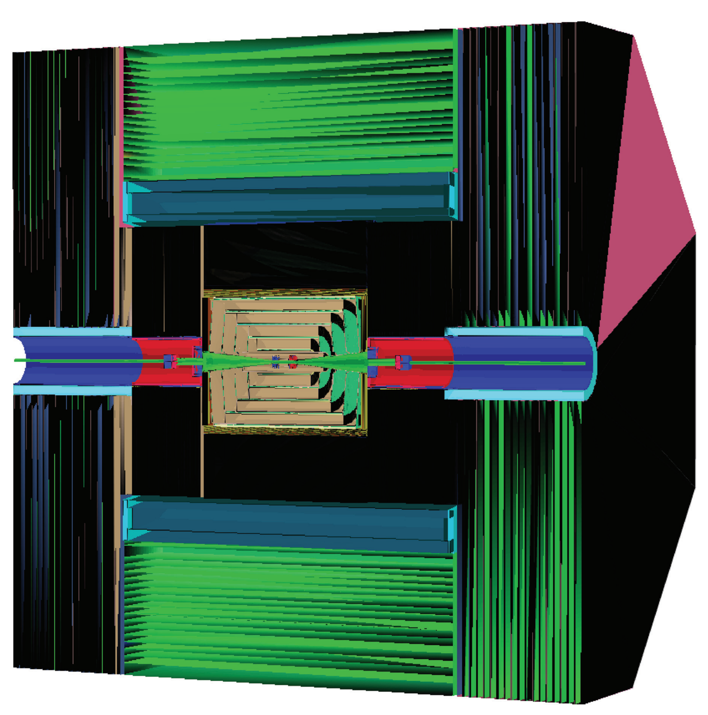
```
<envelope vis="ILD_ECALVis">
  <shape type="BooleanShape" operation="Subtraction" material="Air">
    <shape type="BooleanShape" operation="Subtraction" material="Air">
      <shape type="BooleanShape" operation="Intersection" material="Air">
        <shape type="Box" dx="Det_outer_radius" dy="Det_outer_radius" dz="Det_max_z"/>
        <shape type="PolyhedraRegular" numsides="Det_symmetry"
               rmin="0" rmax="Det_outer_radius" dz="2.0*Det_max_z"/>
        <rotation x="0*deg" y="0*deg" z="90*deg-180*deg/Det_symmetry"/>
      </shape>
      <shape type="Box" dx="Det_inner_radius" dy="Det_inner_radius" dz="Det_max_z"/>
    </shape>
    <shape type="Box" dx="Det_outer_radius" dy="Det_outer_radius" dz="Det_min_z"/>
  </shape>
</envelope>
```

Complex envelope described in XML and placed with single line in the c++ driver

```
Volume envelope = XML::createPlacedEnvelope(lcdd, element, sdet);
if(lcdd.buildType() == BUILD_ENVELOPE) return sdet;
```



ILD Detector envelopes



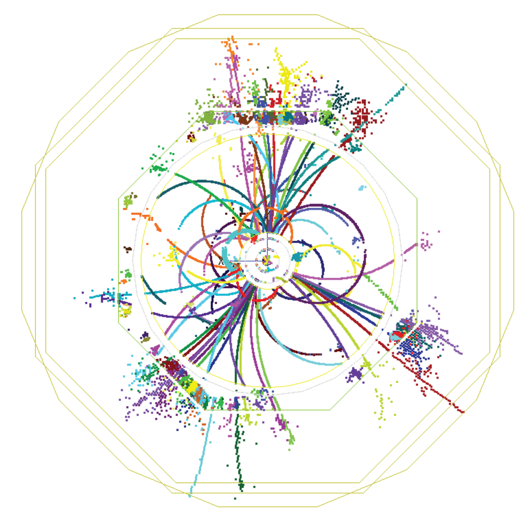
Detailed CLIC detector

Reconstruction

To facilitate the use of common reconstruction tools agnostic of the detailed geometry implementation, a **high level interface** to the geometry information is required

- The generic DDRec API **decouples** the reconstruction code the from the specific implementation of the **detailed sub-detector geometry**
- For the track reconstruction the interface is provided via **surfaces** that are attached to the `DetElement` volumes
  - Track reconstruction can be run on **any** detector equipped with these surfaces
- For calorimeter reconstruction, e.g., clustering using particle flow algorithms, the required **information is abstracted** from the detailed sub-detector geometry
  - *Simple data structures* to describe calorimeter information are attached to the `DetElements` using the **DD4hep** extension mechanism
- Use the segmentation classes to identify neighbouring cells based on cell IDs

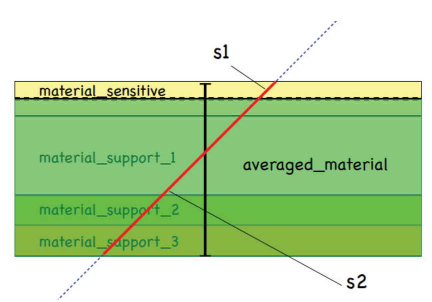
- The linear collider reconstruction software is programmed against the DDRec API
- Allows separate development of detector geometry and reconstruction software
- Software chain can be used by groups outside the linear collider community if geometry is described through **DD4hep**



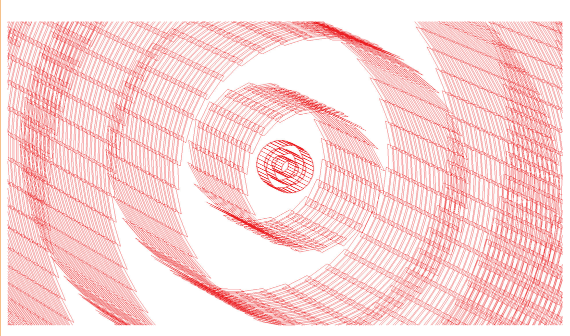
Same event as above after reconstruction: colours based on reconstructed particle flow objects

Tracking Surfaces

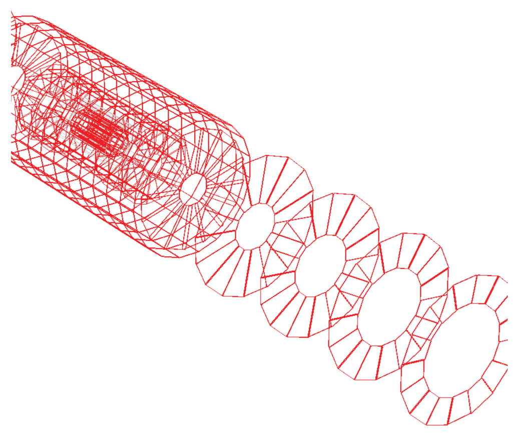
- Tracking code needs a special interface to geometry
- Measurement and dead material surfaces
- Surfaces are attached to volumes, which define the boundaries
- Surfaces provide
  - vectors describing the the surface (u, v), normal vector, origin
  - Inner and outer thicknesses
  - Material properties are **automatically averaged** from the detailed geometry
  - Global to Local, Local to Global transformations: $(u, v) \iff (x, y, z)$



- Surfaces can be added through plug-ins identifying volumes that need to be equipped with a surface; no changes to the detector constructor required
- Surfaces can also be used for **fast simulation**



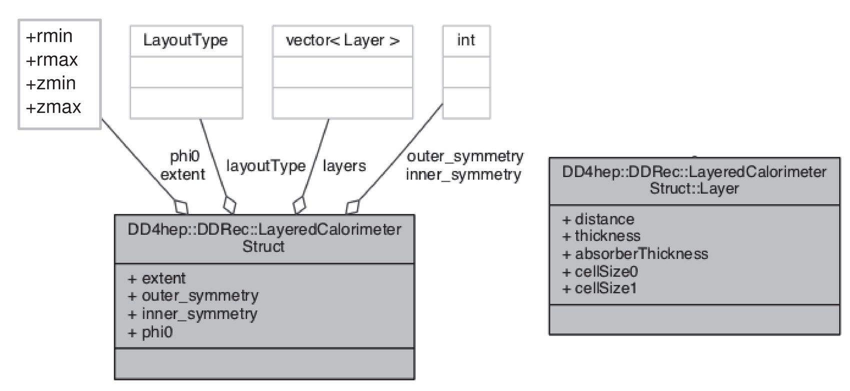
Surfaces of the CLIC Vertex and Silicon Barrel Tracker



Surfaces of the ILD inner tracking modules (VXD, SIT, FTD)

Subdetector Abstraction

- Define simple data structures with abstract view on sub-detectors needed for reconstruction
- Provides additional information on tracking geometry that can be used for pattern recognition
- Use extension mechanism to attach these to `DetElements`
- Example: `LayeredCalorimeterData`



- Similar data structures for other sub-detectors

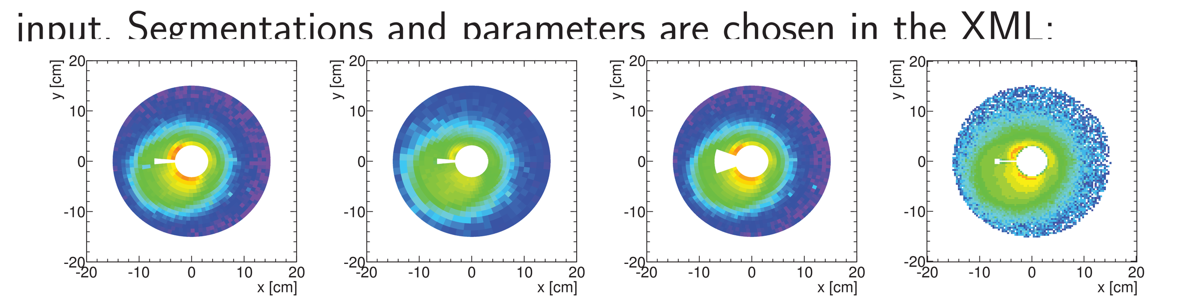
Data Structure	Detector Type	Example
`ConicalSupportData`	Cones and Tubes	BeamPipe
`FixedPadSizeTPCData`	Cylindrical TPC	TPC
`LayeredCalorimeterData`	Sandwich Calorimeters	ECal, HCal, BeamCal, LumiCal, LHCal
`ZPlanarData`	Planar Silicon Trackers	VXD, SIT, SET
`ZDiskPetalsData`	Forward Silicon Trackers	FTD

Segmentation

Energy deposits in the calorimeter sensitive detectors are combined into cells uniquely identified by a cell ID. `DDSegmentation` provides virtual segmentation of the sensitive volumes

- Different segmentation types already exist: Cartesian grids, polar R-Phi grids, projective cylinders
- Segmentations must fulfil only simple interface
  - Transform position local or global to integer cell ID
  - Transform cell ID to local or global position

Different segmentations for the same sub-detector and simulation input. Segmentations and parameters are chosen in the XML:




```
<segmentation type="PolarGridRPhi2"
              grid_r_values="3*cm 4*cm 5*cm ... 15*cm"
              grid_phi_values="SpanningPhi/(4*8)*deg SpanningPhi/(5*8)*deg
                               ... SpanningPhi/(15*8)*deg"
              offset_phi="-180*deg+(360*deg-SpanningPhi)*0.5" />
```

```
<segmentation type="CartesianGridXY" grid_size_x="0.35*cm"
                                     grid_size_y="0.35*cm" />
```

Summary&Outlook

- The implementation of **DD4hep** into the linear collider software framework is almost complete
  - Testing and validation are ongoing
- CLICdp and ILD will use the new software chain for future physics and detector optimisation studies