Linear Collider Software and Computing

N. Nikiforou, CERN/EP-LCD and University of Texas at Austin
On behalf of the CLICdp and ILD collaborations

This project has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement no. 654158.
Introduction

- The LC community has a tradition of common software development and tool sharing
- Software is shared by the detector concepts of both ILC and CLIC and the hardware R&D groups, and even projects like FCC, CEPC, ...
  - Detector design and optimization
  - Technology studies
  - Physics performance studies
- The tools should be generic, flexible, and robust to be used with different detector concepts and their variations
- Collaborative SW development, pooling of manpower and resources towards common goals
Linear Collider Software: iLCSOft

- Event Data Model **LCIO** common to all detector concepts
- Applications typically run via “processors” within a modular C++ application framework called **Marlin**
- New common Detector Geometry Description and Simulation Framework: **Detector Description 4 HEP (DD4hep)**

---

Event Data Model: **LCIO**

- Generator: Whizard, Pythia, …
- Simulation: DDG4, …
- Reconstruction: Overlay, Digitization, Tracking, PFA
- Analysis: Vertexing, Jet Clustering, Flavor Tagging

Detector Geometry from **DD4hep**

---

This talk: focus on new developments from ILD and CLICdp
DD4hep – A single source of information

**Compact description**
- xml

**Detector drivers**
- C++
- python

**Generic Detector Description Model**
- Based on ROOT TGeo C++

**Geometry Display**
- (Via ROOT OpenGL Viewer)
- geoDisplay
teveDisplay

**Extensions where required**
- TGeo ⇒ G4 converters
- DDG4

**Reconstruction Extensions**
- DDRec

**Analysis Extensions**

**Conditions DB**
- DDCond

**Extensions**
- Geant4 Program (ddsim)
- Event Display (less geometry detail) (CED)
- Reconstruction Program (Marlin)
- Analysis Program

**Alignment / Calibration**
- DDAAlign

**DD4hep aims to support the entire experimental lifecycle:**
- Detector design and Optimization
- Construction
- Operation, data taking, and analysis

**DD4hep** aims to support the entire experimental lifecycle:
- Detector design and Optimization
- Construction
- Operation, data taking, and analysis

**Extensions**
- Geant4 Program (ddsim)
- Event Display (less geometry detail) (CED)
- Reconstruction Program (Marlin)
- Analysis Program

**Alignment / Calibration**
- DDAAlign

**Conditions DB**
- DDCond

**DD4hep** aims to support the entire experimental lifecycle:
- Detector design and Optimization
- Construction
- Operation, data taking, and analysis

**Extensions**
- Geant4 Program (ddsim)
- Event Display (less geometry detail) (CED)
- Reconstruction Program (Marlin)
- Analysis Program

**Alignment / Calibration**
- DDAAlign

**Conditions DB**
- DDCond

**DD4hep** aims to support the entire experimental lifecycle:
- Detector design and Optimization
- Construction
- Operation, data taking, and analysis

**Extensions**
- Geant4 Program (ddsim)
- Event Display (less geometry detail) (CED)
- Reconstruction Program (Marlin)
- Analysis Program

**Alignment / Calibration**
- DDAAlign

**Conditions DB**
- DDCond
Detector drivers in DD4hep

- Generic driver palette available: **Scalable and flexible detector drivers**
- You can adapt/write your own
  - User decides balance between detail and flexibility
- Visualization, Radii, Layer/module composition in compact xml
  - Example above
- Volume building in C++ driver
  - See backup
- **Once you have the detector geometry, you can extend it, i.e. add more information using the Reconstruction Extensions** (next slide)
DDRec: Interface to Reconstruction

- The user can attach any object that could help during reconstruction to a DetElement
  - e.g. HCal barrel, ECal endcap, Vertex det., ...
- The subpackage DDRec currently offers two main options:
  - **Simple data structures** that get filled by the detector constructor at creation time
    - Detector layout, symmetry, extent, ...
    - # layers, technology, material properties, ...
  - **Surfaces**: special type of extension foreseen mainly for tracking
    - Measurement directions
    - Material effects automatically averaged from the detailed model
DDG4: Gateway to Geant4

- **DD4hep** performs **in-memory translation of geometry** from **TGeo** to **Geant4**
- Plugin Mechanism
  - Sensitive detectors, segmentations and configurable actions, …
- Configuration mechanism (via **Python**, **XML**, **ROOT**)
  - Physics lists, regions, limits, filters, fields, sensitive detectors, …
- MC Truth history and linking
- Support for different input/output file formats

- Users can write own simulation applications with **DDG4**

- We already provide a fully working and flexible simulation program called **ddsim** (next slide)
ddsim application

- Python executable with many configuration options
  - Configure most useful and common user options in the command line
  - Even supports tab-completion of arguments and their options!
  - Can also configure with a “steering file”

```
ddsim -h
usage: Running DD4hep Simulations: [-h] [--steeringFile STEERINGFILE]
      [[--compactFile COMPACTFILE] [-runType {batch,vis,run,shell}]]
      [--inputFiles INPUTFILES [INPUTFILES ...]] [--outputFile OUTPUTFILE] [-v PRINTLEVEL]
      [--numberOfEvents NUMBEROFEVENTS] [--skipNEvents SKIPNEVENTS]
      [--physicsList PHYSICSLIST] [--crossingAngleBoost CROSSINGANGLEBOOST]
      [--vertexSigma VERTEXSIGMA VERTEXSIGMA VERTEXSIGMA VERTEXSIGMA VERTEXSIGMA]
      [--vertexOffset VERTEXOFFSET VERTEXOFFSET VERTEXOFFSET VERTEXOFFSET VERTEXOFFSET]
      [--macroFile MACROFILE] [--enableGun]
      [--enableDetailedShowerMode] …
```

And much more…
Continuously implementing more options!

- Mature, validation in parallel to DDG4 in advanced stage
- To be used by ILD and CLICdp in the next large scale simulation productions
A collection of pattern recognition tools and algorithms is available in iLCSoft, shared by the various detector concepts

Even with different technologies: Si+TPC (ILD) Vs Full-Si (CLICdp and SiD)

MarlinTrk provides a common interface to pattern recognition

Interfaced with DD4hep

Could mix-and-match different pattern recognition algorithms with different track fitters

A generic Tracking Package is available to everybody out of the box
Particle Flow Reconstruction

- Reconstruction of the individual visible final state particles
- **PandoraPFA**: Not tied to particular geometry or framework
  - Customers: CLICdp, ILD, SiD, Calice, LAr-TPC, …
- Run through **DDMarlinPandora** with **DD4hep** as single source of geometry information
  - No material or other geometry info in processor parameters
  - Preserve independence from particular detector geometry

Marlin Steering File
DDPandoraPFANewProcessor parameters:
- Pandora Settings
- Collection names
- Calibration constants and cuts
- TrackCreator algorithm

DD4hep LCDD instance in Marlin

Auxiliary classes
- DDMCParticleCreator
- DDPfoCreator
- DDGeometryCreator
- DDCaloHitCreator
- DDTrackCreatorBase

- DDTrackCreatorCLIC
  - OR

- DDTrackCreatorILD
  - ...
Event simulated, reconstructed and visualized fully with DD4hep

- **ILD_o1_v05** model implemented in DD4hep
- \( Z' \rightarrow jj \) event at \( \sqrt{s} = 500 \text{ GeV} \) 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
iLCDirac is based on the DIRAC interware originally developed for LHCb

- **Dirac (Distributed Infrastructure with Remote Agent Control):** High level interface between users and distributed resources
- **iLCDirac:** Additional functionality to provide simple interface for the users to the LC Software
- Central system for large scale productions

Full Generation, Simulation, Reconstruction and Analysis chain from iLCSoft available on iLCDirac

```python
from ILCDIRAC ... Applications
import DDSim
dd = DDSim()
dd.setVersion("ILCSoft-01-18-00")
dd.setDetectorModel("CLIC_o2_v03")
dd.setInputFile("LFN:/ilc/prod/clic/500gev/Z_uds/gen/0/00.stdhep")
dd.setNumberOfEvents(30)
```
DIRAC Web interface

- Nice clean interface for configuration and management of both large scale production and user jobs
Using some dedicated and many opportunistic resources from LCG & OSG

- 4 Million jobs, 2500 CPU years
- Peak of about 15 to 20 thousand jobs
- Created and transferred almost 2 PB
- Mainly simulation and reconstruction for detector optimization and physics benchmark studies
Summary

- Effort to have as much as possible common software in the Linear Collider community

- **iLCSoft** offers a collection of flexible tools for **not just the LC community**. It can be used out of the box to:
  - Develop and optimize detector designs and simulation models
  - Develop and test reconstruction algorithms and analysis tools
  - Run the full generation, simulation, and reconstruction chain

- Generic software tools made possible due to common EDM (**lcio**) and geometry support (**DD4hep/DDRec/DDG4**)

- **iLCDirac** provides a unified interface to the grid resources used by the LC community
BACKUP SLIDES
LCIO – Common Event Data Model

- Common event data model (EDM) and persistency for linear collider community
  - Joint DESY and SLAC (and LLR) project - first presented @ CHEP 2003
- Used by ILD, SiD, CLICdp and test beams for more than 10 years
- Common EDM proven to be crucial for collaborative SW development across detector concepts
Marlin

- **Modular Analysis & Reconstruction for the Linear collider**
  - modular C++ application framework for the analysis and reconstruction of LC data
  - **LCIO** as transient data model
    - event data bus/white board model
  - **xml** steering files:
    - fully configure application
    - order of modules/processors
    - parameters global + processor
  - self documenting
    - parameters registered in user code
  - consistency check of input/output collection types
  - **Plug & Play** of modules
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

<table>
<thead>
<tr>
<th>Project</th>
<th>Notes</th>
<th>DD4hep</th>
<th>DDG4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILD</td>
<td>F. Gaede et al., ported complete model ILD_o1_v05 from previous simulation framework (Mokka)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CLICdp</td>
<td>New detector model being implemented after CDR, geometry under optimization</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SiD</td>
<td>Recently decided to move to DD4hep</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CALICE</td>
<td>Started</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FCC-eh</td>
<td>P. Kostka et al.</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FCC-hh</td>
<td>A. Salzburger et al.</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>FCC-ee</td>
<td>Interest expressed, already used in studies</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>CEPC</td>
<td>Investigations ongoing</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>LHCb</td>
<td>Investigations started for LHCb upgrade</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Feedback from users is invaluable and helps shaping DD4hep!
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
Geometry Implementation

Subdetector Hierarchy (Tree)

- Detectors
  - DetectorElement
    - PlacedVolume [TGeoNode]
      - [TGeoMatrix]
      - [TGeoBox]
      - [TGeoCone]
      - [TGeoTube]
      - LogicalVolume
        - Envelope [TGeoShape]
        - Material

Geometry

Subdetector status (conditions)

- Alignment
- Conditions
- Readout
- Visualization
- Segmentation

M. Frank
August 4th, 2016
ICHEP 2016
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:

\[
\begin{align*}
\text{nRadLengths} & \;+= \; \text{slice\_thickness}/(2*\text{slice\_material.radLength}()); \\
\text{nIntLengths} & \;+= \; \text{slice\_thickness}/(2*\text{slice\_material.intLength}()); \\
\text{thickness\_sum} & \;+= \; \text{slice\_thickness}/2;
\end{align*}
\]

- 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
Example HCal Barrel Driver

- Always within a function called

```cpp
static Ref_t create_detector(LCDD\& lcdd, xml_h e, SensitiveDetector sens)
{
    ...
    return sdet;
}
```

- Macro to declare detector constructor at the end:

```cpp
DECLARE_DETELEMENT(HCalBarrel_o1_v01, create_detector)
```
LayeredCalorimeterStruct

```
LayeredCalorimeterStruct
  +outer_thicknss
  +cellSize0

vector< DD4hep::DRec::LayeredCalorimeterStruct::Layer >

+extact
+gap0
+outer_phi0
+gap1
+inner_phi0
+gap2
+phi0

+layoutType
+outer_symmetry
+inner_symmetry

+layers

+cellSize1
+absorberThicknss
+inner_thicknss
+outer_nRadiationLengths
+inner_nRadiationLengths
+sensitive_thicknss
+distance
+inner_nInteractionLengths
+outer_nInteractionLengths
+thicknss
...
```
Measurement surfaces

- Special type of extension, used primarily in **tracking**
- Attached to **DetElements** and **Volumes** (defining their boundaries)
  - Can be added to drivers via **plugins** without modifying detector constructor
- They hold *u*, *v*, *normal* and origin vectors and **inner/outer thicknesses**
- Material properties **averaged automatically**
- Could also be used for **fast simulation**

- Outlines of surfaces drawn in teveDisplay for CLICdp Vertex Barrel and Spiral Endcaps
DDG4: Gateway to Geant4

- DD4hep facilitates **in-memory translation of geometry** from TGeo to Geant4
- Plugin Mechanism:
  - Sensitive detectors, segmentations and configurable actions, …
- **All shared with Reconstruction!**

- Configuration mechanism (via python, XML, CINT)
  - Physics lists, regions, limits, fields, …

- For example, configure and launch the simulation using python (next slide)
DDG4 configuration

- DDG4 is highly modular
- Easy to configure, especially if one uses the python dictionaries
- Configure actions, filters, sequences, cuts, ...

```python
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
...
Where can I find all this?

- **DD4hep** comes complete with example drivers and compact files in iLCSoft releases
  - Under **DD4hep/<version>/DDDetectors**
  - More examples and use cases under **DD4hepExamples**
- For the Linear Collider Community we have another package: **LCGeo**
  - We collect here the concrete implementations of Detector Models (currently for CLICdp and ILD)
    - All their versions, additional specialized subdetector drivers if needed
  - We also have use case examples, configuration files and tools including **ddsim**, a tool to run **DDG4 simulation**