Software for Detector Development Studies and Production

André Sailer, Marko Petrič

CERN-EP-LCD

Joint Workshop of Future Tau-Charm Factory
December 4, 2018
Outline

1 Introduction

2 Reconstruction

3 Distributed Computing with DIRAC

4 Software Infrastructure

5 Conclusions
Introduction: CLICdp

- Detector simulation studies for high-energy (380 GeV to 3 TeV) electron–positron collider CLIC
- Physics studies with full Geant4 simulation and reconstruction, including beam-induced backgrounds
- Detector optimisation studies to understand hardware requirements, e.g., Pixel sizes, timing requirements
- Hardware R&D on Si Tracking and Calorimeters
Linear Collider Software

- Linear collider community has used and developed **common software** for many years
  - Event data model (EDM) and persistency: LCIO
    - podIO is being investigated in AIDA2020
  - Particle flow reconstruction: **PANDORAPFA**

- Adopted DD4HEP geometry description to develop more common software this geometry information

- Interface generic reconstruction packages via thin wrappers to linear collider framework
DD4hep – Overview

- **Complete Detector Description**
  - Providing geometry, materials, visualization, readout, alignment, calibration...

- **Supports full experiment life cycle**
  - Detector concept development, detector optimization, construction, operation
  - Facile transition from one stage to the next

- **Single source of information → consistent description**
  - Use in simulation, reconstruction, analysis, etc.

- **Ease of Use**
  - Few places for entering information
  - Minimal dependencies

- **AIDA-2020 and HSF member project**

---

**DD4hep**

- **DDCond**
  - Basics
  - Geometry handling
  - STABLE

- **DDCore**
  - Simulation interface
  - to Geant4
  - STABLE

- **DDAlign**
  - Conditions and
  - Alignment support
  - FIRST RELEASE

- **DDG4**
  - Geometry visualization
  - Event Display
  - BASIC

- **DDRec**
  - Reconstruction interface
  - STABLE

- **DDEve**
  - Event Display
  - BASIC
What is DD4hep Detector description?

- Description of a tree-like hierarchy of ‘detector elements’
  - Sub-detectors or parts of subdetectors

- Detector Element describes:
  - Geometry
    - points to placed logical volumes
  - Environmental conditions
  - Properties required to process event data
  - Extensions (optionally): experiment, sub-detector or activity specific data, measurement surfaces...
The Big Picture

Generic Detector Description Model based on ROOT TGeo

Provided extensions
- GDML Converter
- TGeo → G4 Converter
- Reco. Extensions
- Analysis Extensions

Event Display

Compact description
- CAD Drawing
- CAD Converter

Detector constructor
- py
- C++

Detector constructor
- Conditions DB

DDDB Converter
- C++

Alignment Calibration

Analysis Program
- Analysis Extensions

Reco. Program
- Reco. Extensions

Geant4 Program
- GDML Converter

Marko Petrič (CERN)
marko.petric@cern.ch
The Generic Detector Palette

- Generic driver available → scalable and flexible
- Parameters are provided in compact XML files, e.g.

```xml
<detector id="15" name="HCal" type="GenericCalBarrel_o1_v01" readout="HCalCollection">
  <envelope vis="HCALVis">
    <shape type="PolyhedraRegular" numsides="HCal_sym" rmin="HCal_rmin" rmax="HCal_rmax" dz="HCal_dz" material="Air"/>
    <rotation x="0*deg" y="0*deg" z="90*deg-180*deg/HCal_symmetry"/>
  </envelope>
  <dimensions numsides="HCal_sym" rmin="HCal_rmin" z="HCal_dz*2"/>
  <layer repeat="(int) HCal_layers" vis="HCallayerVis">
    <slice material="Steel235" thickness="0.5*mm" vis="HCalAbsorberVis" radiator="yes"/>
    <slice material="Steel235" thickness="19*mm" vis="HCalAbsorberVis" radiator="yes"/>
    <slice material="Polystyrene" thickness="3*mm" sensitive="yes" limits="cal_limits"/>
    <slice material="Copper" thickness="0.1*mm" vis="HCalCopperVis"/>
    <slice material="PCB" thickness="0.7*mm" vis="HCalPCBVis"/>
    <slice material="Steel235" thickness="0.5*mm" vis="HCalAbsorberVis" radiator="yes"/>
    <slice material="Air" thickness="2.7*mm" vis="InvisibleNoDaughters"/>
  </layer>
</detector>
```

- You can scale, change layers, radii and compositions...
- Propagate visualization attributes to Display
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);
DDG4 – Gateway to Geant4

- In-memory translation of geometry from TGeo to Geant4
  - Materials, Solids, Limit sets, Regions
  - Logical volumes, placed volumes and physical volumes

- External configuration:
  - Plug-in mechanism
  - Property mechanism to configure plug-in instances
  - Supports configuration via XML, Python or ROOT-AClick

- Use plug-in mechanism to configure: Generation, Event Action, Tracking Action, Stepping Action, SensitiveDetector, PhysicsList...

- Provides out of the box MC truth handling w/o record reduction

- DDG4 is highly modular
  - Very easily configurable through python (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
```
Plug-in Palettes

- Providing input handlers, sensitive detectors for most cases...
- Hard to provide Geant4 Sensitive Detectors for all cases
  - Couples detector ‘construction’ to reconstruction, MC truth and Hit production
  - Too dependent on technology and user needs

- Providing palette of most ‘common’ sensitive components for trackers and calorimeters

- Physics lists, Physics/particle constructors etc.
  - Wrapped factory plugins directly taken from Geant4
  - Users extend physics list (e.g. QGSP)

- Several IO handlers (LCIO, ROOT, StdHep, HepEvt, HepMC)
DDSim

- ddsim python executable is part of the DD4hep release
- Get steering file ddsim --dumpSteeringFile > mySteer.py
  - Steering file includes documentation for parameters and examples
  - The python file contains a DD4hepSimulation object at global scope
  - Configure simulation directly from commandline

```python
from DDSim.DD4hepSimulation import DD4hepSimulation
from SystemOfUnits import mm, GeV, MeV
SIM = DD4hepSimulation()
SIM.compactFile = "CLIC_o3_v06.xml"
SIM.runType = "batch"
SIM.numberOfEvents = 2
SIM.inputFile = "electrons.HEPEvt"
SIM.part.minimalKineticEnergy = 1*MeV
SIM.filter.filters ['edep3kev'] =
dict (name="EnergyDepositMinimumCut/3keV", parameter={"Cut" : 3.0*keV})
```

Future Tau–Charm, Dec. 4, 2018
A. Sailer: Software for Detector Development Studies and Production
**High level view** onto the detectors through DDRec DataStructures extensions for DetElements

- Constructors fill DDRec DataStructures
- DataStructures allow to decouple detector implementation from reconstruction algorithms

DataStructures contain sufficient information to provide geometry information to particle flow clustering via PandoraPFA

<table>
<thead>
<tr>
<th>Data Structure</th>
<th>Detector Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConicalSupportData</td>
<td>Cones and Tubes</td>
</tr>
<tr>
<td>FixedPadSizeTPCData</td>
<td>Cylindrical TPC</td>
</tr>
<tr>
<td>LayeredCalorimeterData</td>
<td>Sandwich Calorimeters</td>
</tr>
<tr>
<td>ZPlanarData</td>
<td>Planar Silicon Trackers</td>
</tr>
<tr>
<td>ZDiskPetalsData</td>
<td>Forward Silicon Trackers</td>
</tr>
</tbody>
</table>
Track Reconstruction

Track reconstruction using DD4HEP surfaces

- Pattern recognition/track finding algorithms
  - From detector specific: Clupatra for TPC; mini-vector for vertex detector double layers
  - to geometry agnostic: pattern recognition in conformal space

- Track fitting, fairly generic: DDKalTest, AIDA2TT
  - ACTS might be long term replacement (AIDA2020)

- Geometry: Interfaced via DDREC and Surfaces

DDKalTest using DD4HEP surfaces for track fitting
Particle Flow Reconstruction

**PANDORA PFA:** generic toolkit for pattern recognition algorithms in highly granular calorimeters

- Originally developed for ILC/CLIC detectors
- Extended to work in LAr-TPC reconstruction for the DUNE neutrino experiment

**ClientApplication:** DDMARLINPANDORA glues linear collider framework (Marlin), DD4HEP, and PANDORA PFA

- Passes DDREC DataStructures information, tracks, and calorimeter hits to PANDORA PFA
- Converts PANDORA PFA objects into LC EDM objects
Distributed Computing with DIRAC

DIRAC (Distributed Infrastructure with Remote Agent Control): High level interface between users and distributed resources

- **WorkloadManagement**
- **Transformation** system for automated and centralized tasks
- **DataManagement** system including *Replica* and *Metadata Catalog*, asynchronous operations (file transfers (FTS), removal)
- *iLCDirac* extension for the linear collider community, implementing interfaces for LC applications and transformation workflows
Resources

Resources behind abstract interfaces, new types can be added easily

- **Computing**
  - Grid Computing Elements: CREAM, ARC, HTCondor-CE
  - Clouds
  - Batch farms

- **Storage**
  - xroot, SRM2, HTTPS, GSIFTP, ECHO

ResourceStatusSystem for automatic state transitions: Active/Banned/Probing
DIRAC users can submit jobs in different ways (JDL, Ganga, python script)

- In iLCDirac python API for existing workflow modules offer a straightforward interface
- Including modules for Gaudi (FCCSW)

```python
from DIRAC.Core.Base import Script
Script.parseCommandLine()
import UserJob, Marlin, DiracILC
d = DiracILC()
j = UserJob()
j.setOutputData("recEvents.slcio")
m = Marlin()
m.setVersion("ILCSoft-01-17-09")
m.setSteeringFile("Steering.xml")
m.setInputFile("SimEvents.slcio")
j.append(m)
j.submit(d)
```
Transformation System

- Central system to run standard reconstruction, create large Monte Carlo, file transfers, etc.
- Define tasks and the system takes care of job (re)-submission, consistency checks
- Interfaced with Dirac Metadata Catalog: use files with given metadata; automatically pick up new files
- Requires very little interaction after definition of tasks
Software Infrastructure

- Source code hosted on github and CERN GitLab instance
  - github.com/ilcsoft
  - github.com/AIDAsoft/DD4hep
  - github.com/DIRACgrid/DIRAC
  - gitlab.cern.ch/CLICdp/ilcdirac/

- Continuous integration (CI) attached to all repositories

Software releases provided via CVMFS

- Available on grid site and for interactive use
- Automatic deployment of nightlies and releases to CVMFS via CI in GitLab

<table>
<thead>
<tr>
<th>Nightly</th>
<th>Build GCC</th>
<th>Build LLVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>iLCSoft</td>
<td>build passing</td>
<td>build passing</td>
</tr>
<tr>
<td>AIDASoft</td>
<td>Build Status</td>
<td>Issues 13</td>
</tr>
<tr>
<td>DD4hep</td>
<td>build passing</td>
<td>Issues 0</td>
</tr>
<tr>
<td>aidATT</td>
<td>build passing</td>
<td>Issues 0</td>
</tr>
<tr>
<td>iLCSoft</td>
<td>Build Status</td>
<td>Issues 13</td>
</tr>
<tr>
<td>CED</td>
<td>build passing</td>
<td>Issues 0</td>
</tr>
<tr>
<td>CEDViewer</td>
<td>build passing</td>
<td>Issues 13</td>
</tr>
<tr>
<td>CLICPerformance</td>
<td>build passing</td>
<td>Issues 0</td>
</tr>
</tbody>
</table>

TAG:

- installSL6
- compiledoc
- docDev
- makeILCConfig
- deploydoc
- deploytag
- deployToCVMFS
Conclusions

- Validated software for detector optimisation and physics studies from Geant4 simulation to user analysis
  - Based on DD4hep
  - Reconstruction aimed at Si tracking, Particle flow, and high energy

- Access to distributed resources via (ilc)Dirac
  - Installation can be extended to accommodate additional communities

- Plans at CERN to create more synergy between software used for CLIC and
  - Replace underlying framework
  - Common components: DD4hep; later maybe podIO and ACTS
  - Need to keep our software running to continue the full simulation studies