DETECTOR SIMULATIONS
WITH DD4hep

CHEP 2016

Markus Frank, Frank Gaede, Shaojun Lu,
Nikiforos Nikiforou, Marko Petrič, André Sailer
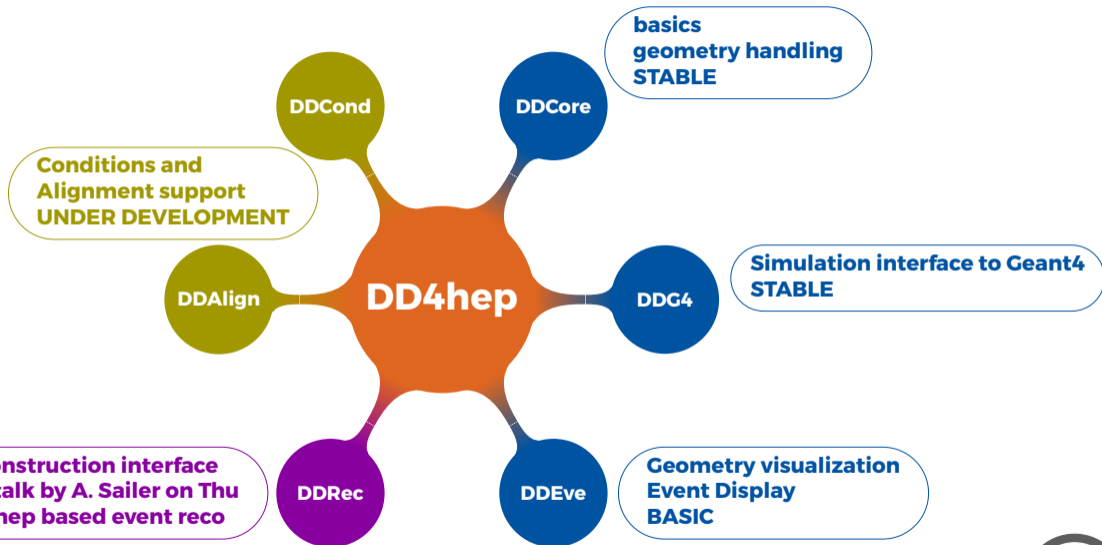
On behalf of the CLICdp and ILD collaborations

San Francisco, 11 October 2016

Overview
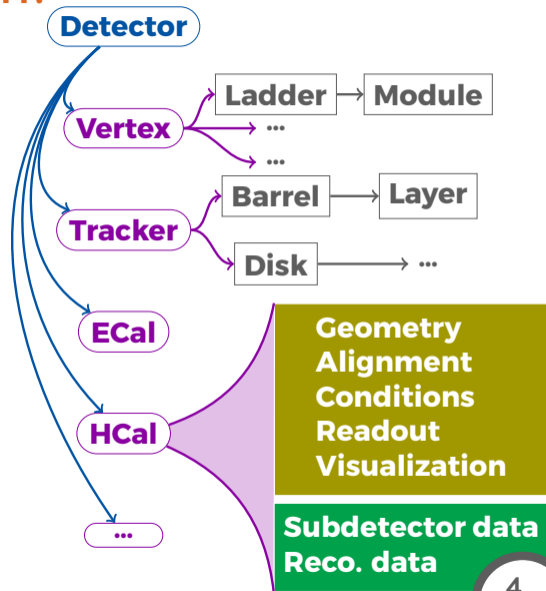
- **Complete Detector Description**
  - **Providing geometry, materials, visualization, readout, alignment, calibration...**
- **Supports full experiment life cycle**
  - **Detector concept development, detector optimization, construction, operation**
  - **Facile transition from one stage to the next**
- **Single source of information → consistent description**
  - **Use in simulation, reconstruction, analysis, etc.**
- **Ease of Use**
- **Few places for entering information**
- **Minimal dependencies**

Structure and packages

basics
geometry handling
STABLE

DDCore

Conditions and
Alignment support
UNDER DEVELOPMENT

DDCond

DDAlign

DD4hep

Simulation interface to Geant4
STABLE

DDG4

Reconstruction interface
See talk by A. Sailer on Thu
DD4hep based event reco

DDRec
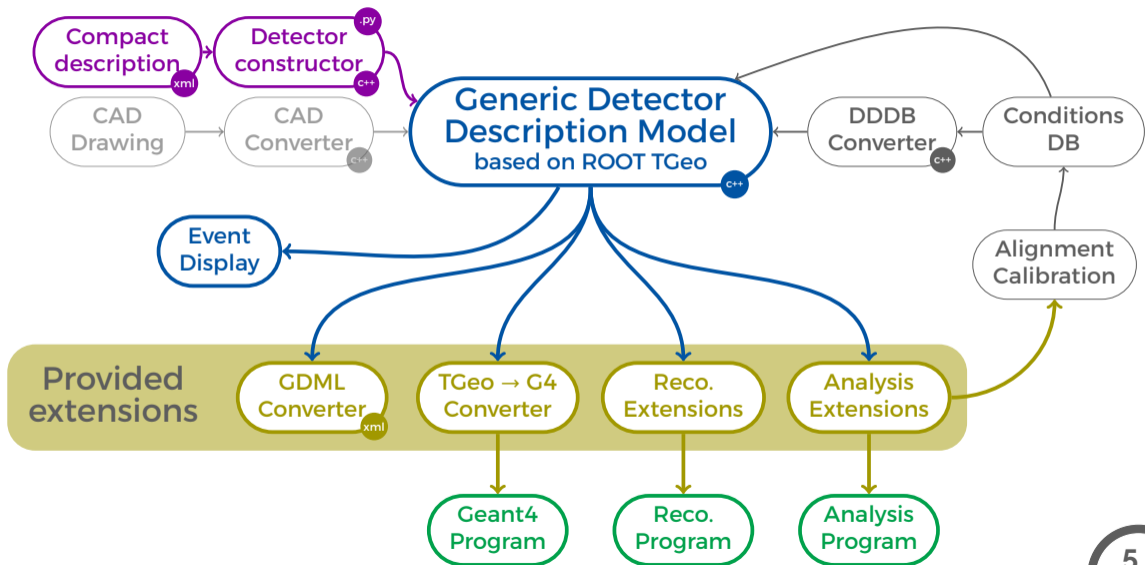
DDEve

Geometry visualization
Event Display
BASIC

What is Detector description?

- **Description of a tree-like hierarchy of 'detector elements'**
  - Sub-detectors or parts of subdetectors

- **Detector Element describes:**
  - Geometry
  - Environmental conditions
  - Properties required to process event data
  - Extenstions (optionally): experiment, sub-detector or activity specific data, measurement surfaces...

Detector

Vertex → Ladder → Module
→ ...
→ ...

Tracker → Barrel → Layer
→ Disk → ...

ECal

HCal

...

Geometry
Alignment
Conditions
Readout
Visualization

Subdetector data
Reco. data

The Big Picture

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**

Your Detector Palette

```cpp
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);
```

- **Users can easily write their own detector drivers, if needed**
- **Detector geometry extendable with additional info.**
- **C++ model of separation of 'data' and 'behavior'**
  - **Classes consist of a single 'reference' to the data object**
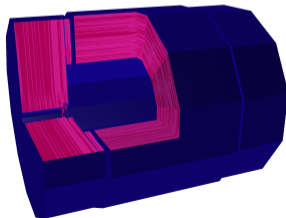
DDG4 – Gateway to Geant4

- **In-memory translation of geometry from TGeo to Geant4**
  - **Materials, Solids, Limit sets, Regions**
  - **Logical volumes, placed volumes and physical volumes**
- **External configuration:**
  - **Plugin mechanism**
  - **Property mechanism to configure plugin instances**
  - **Supports configuration via XML, Python or ROOT-AClick**
- **Use plugin 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 – Configuration example

- **DDG4 is highly modular**
- **Very easily configurable through python**

```python
#...
gen = DDG4.GeneratorAction( kernel , "LCIOInputAction/LCIO1" )
gen.Input = "LCIOFileReader|" + inputFile
#...
```

- **Or 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
#...
```

9

Plugin 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

e.g. several possibilities for tracker
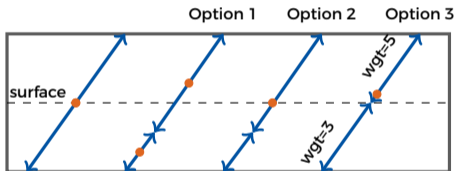
- Providing palette of most 'common' sensitive components for trackers and calorimeters
- **Physics lists, Physics/particle constructors etc.**
    - Wrapped factory plugins directly taken from Geant4
    - Users extend physics list (e.g. QGSP)
- **Several IO handlers (LCIO, ROOT, StdHep, HepEvt, HepMC)**

The Linear Collider Case

- **ddsim python executable part of lcgeo and the latest iLCsoft 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, keV
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} )
```

```
$ ddsim
--action.calo              --filter.tracker          --part.keepAllParticles
--action.mapActions        -G                        --part.minimalKineticEnergy
--action.tracker           --gun.direction           --part.printEndTracking
--compactFile              --gun.energy              --part.printStartTracking
--crossingAngleBoost       --gun.isotrop             --part.saveProcesses
--dump                     --gun.multiplicity        --physics.decays
--dumpParameter            --gun.particle            --physics.list
--dumpSteeringFile         --gun.position            --physics.pdgfile
--enableDetailedShowerMode -h                        --physics.rangecut
--enableGun                --help                    --printLevel
--field.delta_chord        -I                        --random.file
--field.delta_intersection --inputFiles              --random.luxury
--field.delta_one_step     -M                        --random.replace_gRandom
--field.eps_max            --macroFile               --random.seed
--field.eps_min            -N                        --random.type
--field.equation           --numberOfEvents          --runType
--field.largest_step       -O                        -S
--field.min_chord_step     --outputFile              --skipNEvents
--field.stepper            --output.inputStage       --steeringFile
--filter.calo              --output.kernel           -v
--filter.filters           --output.part             --vertexOffset
--filter.mapDetFilter      --output.random           --vertexSigma
```
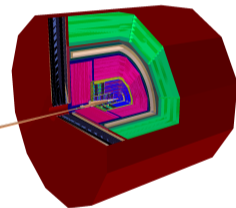
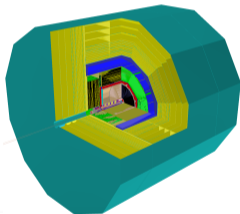# CLIC Example

- **The CLIC Detector Model**
- $e^+ e^- \rightarrow H(\rightarrow b\bar{b})\nu\nu$
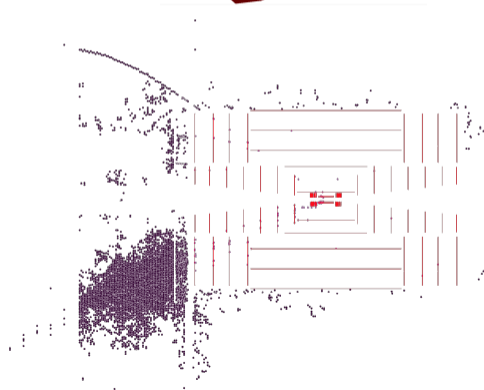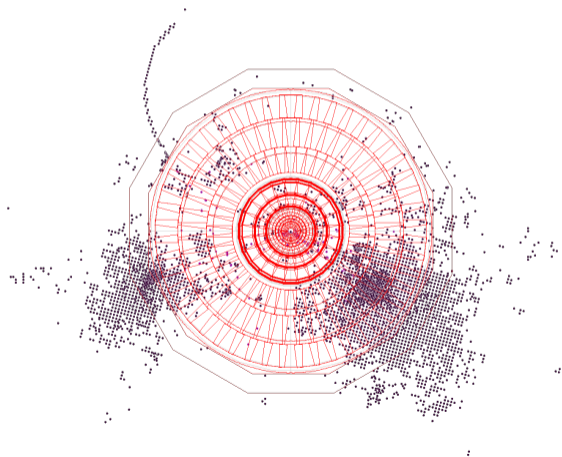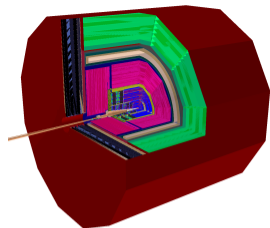
Users of DD4hep
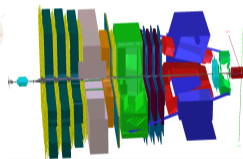

**DD4hep
DDG4**

**DD4hep
DDG4**
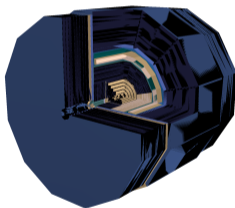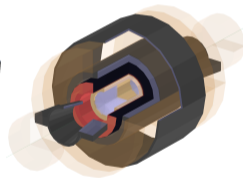
**DD4hep
DDG4**

DD4hep

**Under
investigation**

▶ **The user base for DD4hep is growing**

Summary and Conclusion

- **DD4hep provides a consistent single source of detector geometry for simulation, reconstruction, analysis**
- **Enables full simulation w/ Geant4 of particle collisions in detectors with minimal effort: simple, easy, flexible**
- **The DD4hep toolkit is getting accepted by wider HEP community**
  - **Used by CLIC, ILC, FCC detector communities**
- Development continues in parallel with validation
- DD4hep can host user plugins: extensible
- Continued plugin suite development to cover all simulation needs (I/O, MC truth, etc)
- Find us at https://github.com/AIDASoft/DD4hep