Status of Geometry: DD4Hep

AIDA 1st Annual Meeting, 28-30 March 2012, DESY
Pere Mato / CERN

28/3/2012
Outline

- Geometry Task Goals
- Requirements
- Design Elements
- Prototype
- Deliverable
AIDA-WP2 Geometry Tasks

- Geometry Toolkit
  - Set of software tools which can describe the geometry of the detector, the material it is made from and different ways of detecting particles
  - High/Low level descriptions, primitives library, interchange formats, API for reconstruction, simulation, alignment support, etc.
  - Repository of generic detectors types

- Started the development of a prototype (DD4Hep)
  - Useful to clarify required functionality
  - Evaluation of design choices
Toolkit Main Requirements

* Full Detector Description
  * It includes geometry, materials, visualization, readout, alignment, calibration, etc.

* Full Experiment life cycle
  * Supporting all phases of the life cycle: detector concept development, detector optimization, construction, operation
  * Easy transition from one phase to the next

* Consistent Description
  * Single source of detector information for simulation, reconstruction, analysis

* Ease of Use
  * Few places to enter information. Minimal dependencies.
Current Ideas: The Big Picture

Compact description
Detector constructors
Detector Models Library
User Actions (plugins)

Generic Detector Description Model

Conditions DB (Alignment)

Geometry Display
GDML

G4 Converters
RECO Extensions
G4 Application
RECO Application
Alignment Application

Wednesday, March 28, 12
Compact Description

- Reusing the idea of “compact detector description” from SiD software
- Human readable and compact geometry description in XML format
- Used as the main input to the detector description system
- Extendable with new generic detector types together with very specific ones
Detector Constructors

- A set of code fragments that are able to convert the XML elements into detector description (DD) objects
  - Objects: Material, Element, VisAttributes, Limits, etc.
  - Generic Detectors: SiTrackerBarrel, CylindricalEndcapCalo, etc.
  - Specific Detectors: ILDTPC, etc.
- Prototyped two possible implementations
  - C++ functions (XercesC)
  - Python functions (PyROOT)
def detector_ILDExVXD(lcdd, det):
    vdx = DetElement(lcdd, det.name, det.type, det.id)
    mother = lcdd.worldVolume()

    for layer in det.findall('layer'):
        support = layer.find('support')
        ladder = layer.find('ladder')
        layername = det.name + '_layer%d' % layer.id
        nLadders = ladder.getI('number')
        dphi = 2.*pi/nLadders
        sens_thick = ladder.thickness
        supp_thick = support.thickness
        supp_radius = ladder.radius + sens_thick/2. + supp_thick/2.
        width = 2.*tan(dphi/2.)*(ladder.radius-sens_thick/2.)

        ladderbox = Box(lcdd, layername+'_ladder_box', (sens_thick+supp_thick)/2., width/2., ladder.zhalf)
        laddervol = Volume(lcdd, layername+'_ladder', ladderbox, lcdd.material('Air'))

        sensbox = Box(lcdd, layername+'_sens_box', sens_thick/2., width/2., ladder.zhalf)
        sensvol = Volume(lcdd, layername+'_sens', sensbox, lcdd.material(ladder.material))
        sensvol.setVisAttributes(lcdd.visAttributes(layer.vis))
        laddervol.placeVolume(sensvol, Position(-(sens_thick+supp_thick)/2.+sens_thick/2.,0,0))

        suppbox = Box(lcdd, layername+'_supp_box', supp_thick/2., width/2., ladder.zhalf)
        suppvol = Volume(lcdd,layername+'_supp', suppbox, lcdd.material(support.material))
        suppvol.setVisAttributes(lcdd.visAttributes(support.vis))
        laddervol.placeVolume(suppvol, Position(-(sens_thick+supp_thick)/2.+sens_thick/2.+supp_thick/2.,0,0))

    for j in range(nLadders):
        laddername = layername + '_ladder%d' % j
        radius = ladder.radius + ((sens_thick+supp_thick)/2. - sens_thick/2.)
        rot = Rotation(0,0,j*dphi)
        pos = Position(radius*cos(j*dphi) - ladder.offset*sin(j*dphi),
                        radius*sin(j*dphi) - ladder.offset*cos(j*dphi),0.)
        mother.placeVolume(laddervol, pos, rot)

    return vdx
Detector Description Prototype

- Developed by Markus Frank
- C++ model separation of ‘data’ and ‘behavior’
  - Classes consist of a only single ‘reference’ to the data object
  - Practical advantages concerning compile/link dependencies
  - Same ‘data’ can be associated to different ‘behaviors’
- Implementation based on TGeo (ROOT)
  - TGeom classes directly accessible (no hiding)
  - Support for alignment
Reconstruction Extensions

• The idea is to ‘extend’ the DetectorElement class with specific reconstruction code

• Be able to answer detector questions asked by the reconstruction algorithms. E.g.:
  • transform ECAL ‘cell id’ to local [global] coordinates
  • amount of material to next layer

• These extensions can be added as ‘plug-ins’

```cpp
struct GearTPC : public Geometry::Subdetector {
  typedef TPCData Object;
  GearTPC(const Geometry::RefHandle<TNamed>& e);
  GlobalPadIndex getNearestPad (double c0, double c1) const;
  double getDriftVelocity () const;
  double getReadoutFrequency () const;
  double getInnerRadius() const;
  double getOuterRadius() const;
};

double GearTPC::innerRadius() const {
    Subdetector gas   = data<Object>()->gas;
    Tube        tube  = gas.volume().solid();
    return tube->GetRmin();
}

double GearTPC::outerRadius() const {
    Subdetector gas   = data<Object>()->gas;
    Tube        tube  = gas.volume().solid();
    return tube->GetRmax();
}
```
Generic Geant4 Converters

- The Geant4 detector geometry can be created from the DD model
  - Conversion of TGeom to G4Geometry (currently using VGM)
  - Similarly the way it is done with SLIC (without having to generate an intermediate GDML file since we convert C++ objects to C++ objects)
- This will be facilitated by the USolids library to obtain the exact same behavior
- Plug-ins for User Actions and Sensitive Detectors are foreseen
The entry point to the Generic DD model is an ‘singleton’ (e.g. LCDD)

- Access the detector by its name (e.g. TPC)
- Associate it to a given ‘behavior’ (e.g. GearTPC)
- Start using it
- Draw it!

```cpp
#include "LCDD.h"
#include "GearTPC.h"

int main(int argc, char** argv) {
  LCDD& lcdd = LCDD::getInstance();
  lcdd.fromCompact(argv[1]);
  GearTPC tpc = lcdd.detector("TPC");
  cout << "Gear: Inner:" <<
       tpc.getInnerRadius() << endl;
  cout << "      Outer:" <<
       tpc.outerRadius() << endl;
  return 0;
}
```
Detector Display

- Detector Display applications can be developed using native ROOT functionality (OpenGL, Eve, etc.)

Display of the DD model produced from the SiD compact description (M. Frank)
DD4Hep Prototype

* Started to implement a simple prototype consisting of a simplified structure for the ILD central tracking detectors: VXD, SIT, TPC (Steven Aplin)

  * Simulation program based on Geant4 example N03

* Implemented initial implementations for described components

  * Examples for simulation, reconstruction programs, display

* [http://aidasoft.web.cern.ch/DD4Hep](http://aidasoft.web.cern.ch/DD4Hep)

Delivery Status

- D2.3 - Software design for geometry toolkit including the interfaces for the reconstruction toolkits

- Delayed 2 months:
  - We have made quite a lot of progress in the last two months producing a prototype to exercise some of the key design ideas (see https://aidasoft.web.cern.ch/DD4Hep)
  - We wanted to perform some cross-checks in particular in the area of the interfaces (or gateways) to the simulation and reconstruction applications before finalizing the design and producing the design document.
  - This has basically been done and now we are ready to produce the deliverable
Summary

- Started to develop a prototype to test some of the design ideas
  - Code exists in the repository
  - People are encourage to give feedback
- Usability of the prototype is currently being tested by A. Münich by trying to add new detector types (TPC end-caps, modules and pad planes)
- Started to write the design document for D2.3 deliverable
  - Will not be very different than this few slides
  - Including the USolids part