AIDA-2020-SLIDE-2020-027

AIDA-2020

Advanced European Infrastructures for Detectors at Accelerators

Presentation

DD4hep a community driven detector description tool for HEP

Gaede, F. (DESY) *et al*

05 November 2019

The AIDA-2020 Advanced European Infrastructures for Detectors at Accelerators project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168.

This work is part of AIDA-2020 Work Package **3: Advanced software**.

The electronic version of this AIDA-2020 Publication is available via the AIDA-2020 web site <http://aida2020.web.cern.ch> or on the CERN Document Server at the following URL: <http://cds.cern.ch/search?p=AIDA-2020-SLIDE-2020-027>

DD4hep

a community driven detector description tool for HEP

F.Gaede DESY
M.Frank, M.Petric, A.Sailer CERN

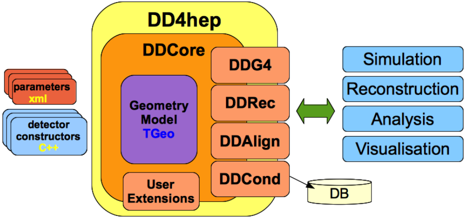
CHEP 2019, Adelaide, Nov 5,2019

Outline

- Introduction
- DD4hep Features
- Recent Developments
- Future Plans
- Summary and Outlook

- see also related talks:
  - A.Sailer: Towards a Turnkey Software Stack for HEP Experiments
  - C.Vuasolo: CMS Experience with Adoption of the Community-supported DD4hep Toolkit
  - D.Muller: Gaussino - a Gaudi-based core simulation framework

- **Goal:** develop a generic detector (geometry) description for HEP
- support the **full life cycle** of the experiment
  - detector concept development
  - detector optimization
  - construction and operation
  - extendible for future use cases
- consistent description with one single data source
  - for simulation, reconstruction, analysis
- complete description:
  - geometry readout, alignment, calibration (conditions), . . .

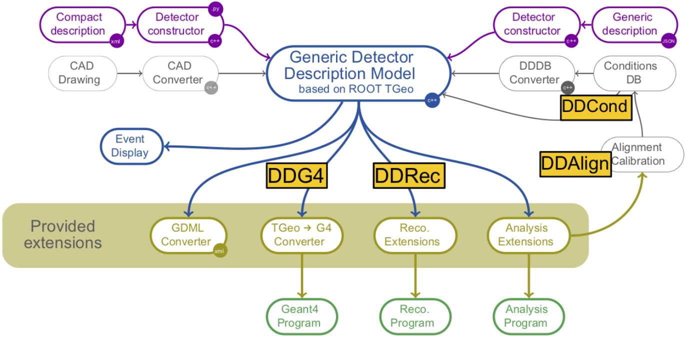


component based design

- developed in AIDA/AIDA2020
- HSF incubator project

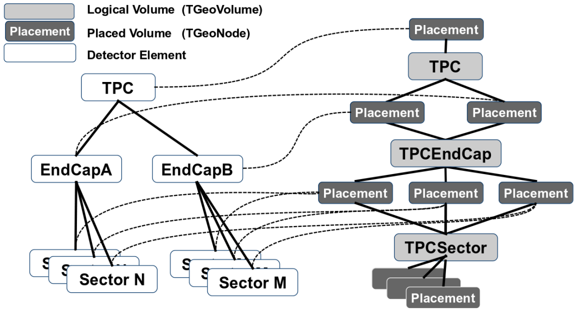
DDCore - core implementation of DD4hep

- DD4hep uses **Root TGeo** as geometry implementation
- geometry description in
  - compact *xml-files* and *C++ drivers*
  - other (generic) input sources
- output formats/interfaces
  - Geant4, GDML, *easily extendible*
- various interfaces (views) on geometry:
  - DDRec, DDEve, DDAlign, . . .



- additional hierarchy of *DetElements* provides access to
  - Alignment, Conditions, Readout (sensitive detectors), Visualization
  - arbitrary *user defined objects*
- define for every *touchable* that needs extra data

- the tree of *DetElements* provides the *high level view* into the detector geometry with subdetectors, measurement layers, etc. . .



DDG4: gateway to full simulation with Geant4

- convert geometry *on the fly* from TGeo to Geant4
- hook into the user entry points provided by Geant4:
  - Stepping, Stacking, Tracking,. . .
- choose detector response and physics list from pre-defined plugins
- start the simulation

provided by DDG4 and DDDetectors

- pre-defined palette of standard **sensitive detectors** (trackers, calorimeters)
- large set of **standard HEP detectors** to get quickly started for conceptual studies

- DDG4 provides all modules needed to run full simulations with Geant4 on any detector that is described in DD4hep
  - either from *DDDetectors* or *user-defined*

- pre-defined, extendible palette of I/O handlers
  - Input: **stdhep, LCIO, HepEvt(ASCII), HepMC(ASCII)**
  - Output: **ROOT, LCIO**
  - easily extendible, e.g. with **EDM4hep**

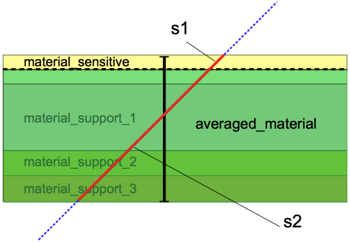
- detailed **MC-Truth** handling w/ and w/o record reduction

DDRec: *high level* interface for reconstruction

- `DetectorData` classes
  - describe high level view of generic detectors (motivated by ILC/CLIC detectors): *dimensions, #layers, thicknesses,...*
- `CellIDPositionConverter`
  - convert cellID to position and vice versa
- `MaterialManager`
  - access material properties at any point or along any straight line

- **tracking** needs special interface to geometry
- measurement and dead material *surfaces* (planar, cylindrical, conical)
- **surfaces** *attached to volumes* in detailed geometry model

- u,v, origin and normal
- inner and outer thicknesses and material properties
- local to global and global to local coordinate transformations:
  - $(x, y, z) \leftrightarrow (u, v)$



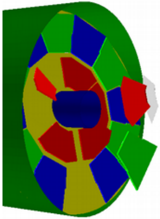
- materials are automatically averaged over surface thickness
  - roughly equivalent for E-loss (Bethe-Bloch)
  - identical for multiple scattering

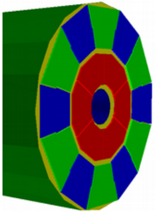
DDAlign: interface for alignment of detector components

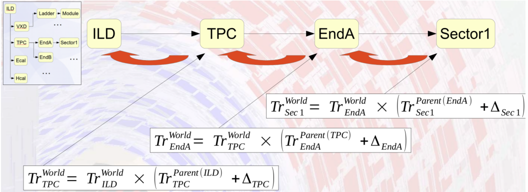
- global alignment corrections
  - physically modify geometry in memory (by TGeo)
  - **not thread safe**
  - possibility to simulate mis-aligned geometries

local alignment corrections

- geometry is static (ideal or mis-aligned)
- **thread-safe**
- local alignments are conditions
- provide *delta-transformations* for hit positions

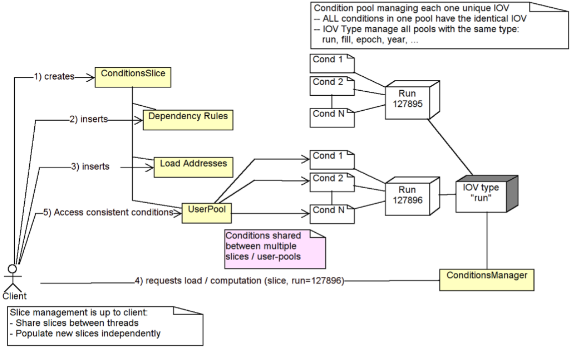






DDCond - Interface to conditions data

- DDCond provides an interface to access conditions data
  - *'slowly'* varying data
  - access through *DetElement* and *Key*
  - organized in IOVs (*Interval of Validity*)
- allow for *derived conditions data*
- **thread-safe** access
- use for example for (delta)-misalignments



- presented in detail at CHEP 2018: M.Frank: Conditions and Alignment extensions to the DD4hep Detector Description Toolkit

Recent developments - since last CHEP

- DD4hep is fully functional for many years and was heavily used in large scale Monte Carlo productions for the **ILD** and **CLICdp** detector concepts, as well as for many **FCC-ee** and **FCC-hh** studies
- interest from **LHCb** and **CMS** has triggered many developments and bug fixes
  - adaptation of geometry input to *legacy* format used by these experiments
  - implementation of more *exotic* shapes not used in the linear collider detectors: *TGeoCtub, TGeoScaledShape, TGeoArb8* and *G4GenericTrap*
  - **surfaces and optical photons**
  - **reflection implementation for left handed coordinates**
- continuous evolution of DD4hep
  - introduce **Python 3** compatibility
  - update cmake builds
  - make compatible with C++17
  - prepared switch to Geant4 system of units *(mm, nsec, MeV)*
  - improved the *plugin manager*

Surfaces and optical photons

- import of surface optical objects in compact input files
  - surface types and optical properties (*refraction*, *absorption*, ...)
  - create TGeo surface objects and tabulated properties
- translation from TGeo to Geant4
- physics components in DDG4, handling:
  - scintillation, Cerenkov and transition radiation
  - reflection, refraction, absorption, wavelength shifting

```
<opticalsurface name="/world/BubbleDevice#WaterSurface" finish="ground" model="unified"
                type="dielectric_dielectric">
  <property name="RINDEX"                   coldim="2" values="2.034*eV  1.35  4.136*eV  1.40"/>
  <property name="SPECULARLOBECONSTANT"     coldim="2" values="2.034*eV  0.3   4.136*eV  0.3 "/>
  <property name="SPECULARSPIKECONSTANT"    coldim="2" values="2.034*eV  0.2   4.136*eV  0.2 "/>
  <property name="BACKSCATTERCONSTANT"      coldim="2" values="2.034*eV  0.2   4.136*eV  0.2 "/>
</opticalsurface>
```

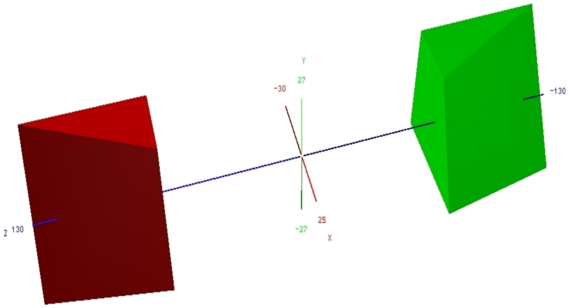
```
#if ROOT_VERSION_CODE >= ROOT_VERSION(6,17,0)
  // Now attach the surface
  OpticalSurfaceManager surfMgr = description.surfaceManager();
  OpticalSurface waterSurf  = surfMgr.opticalSurface("/world/"+det_name+"#WaterSurface");
  OpticalSurface airSurf    = surfMgr.opticalSurface("/world/"+det_name+"#AirSurface");
  BorderSurface  tankSurf   = BorderSurface(description, sdet, "HallTank",   waterSurf,
                                            tankPlace,    enclPlace);
  BorderSurface  bubbleSurf = BorderSurface(description, sdet, "TankBubble", airSurf,
                                            bubblePlace, tankPlace);
  bubbleSurf.isValid();
  tankSurf.isValid();
```

- implemented complete treatment of Optical Photons and surfaces as defined in Geant4

Reflection implementation for left handed coordinates

- reflection with left-handed coordinates in the reflected volumes
- used for *endcap-like* detectors in CMS

```
Rotation3D rot3D(RotationZYX(...));
Position pos3D(...);
if ( reflect )
 rot3D = Rotation3D(1., 0., 0.,
                    0., 1., 0.,
                    0., 0., -1.) * rot3D;
Transform3D tr(rot3D, pos3D);
pv = assembly.placeVolume(volume,tr);
```



Original Reflected

Made DD4hep compatibility with Python 3

- make codebase **python 2 and 3 compatible** at the same time
  - don't maintain two codebases
- replace `print` statements with the python logging module
- used `caniusepython3.pylint_checker` module to identify critical code segment groups
  - address as many as possible of these groups with `python-modernize`
  - run test suite after application of every fix
  - use the `six` module for easier python 2 and 3 compatibility (shipped w/ DD4hep)
- at the end mostly two groups remain: division and unicode
  - use new style division and check all division cases by hand
  - use `unicode_literals` in all files and fix calls to externals API where string is required
  - special care needed to be taken when de-unicoding dicts

users decide whether to use python 2 or python 3 in their software ecosystem

Next evolution step: DDDigi

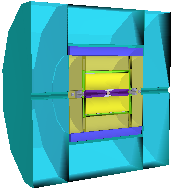
- digitization is the logical next step after simulating the detector response in **DDG4**
- keep separate from simulation step - if possible
  - important for overall CPU usage and re-use of simulated data
- study detector segmentation effects
- detector response to energy depositions
  - Hit/Cluster creation
  - charge sharing
- Incorporate electronics effects
  - noise, cross-talk, etc

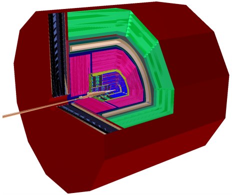
work plan

- develop a suitable data model,
  - consistent with experiments' models – needs to match input and output data
- investigate digitization types
  - detailed models for specialized studies
  - simplified model for bulk productions
- develop a palette of digitization plugins for typical subdetector types
  - parameterized and flexible
  - valid for *most readouts*

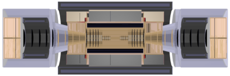
Examples of DD4hep users

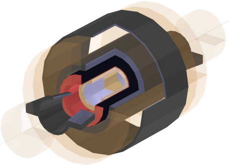
- ILC/CLICdp



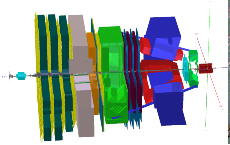


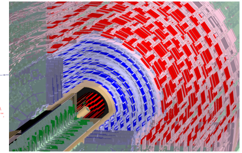
- FCC





- LHCb and CMS





- DD4hep user community is continuously growing:
  - Calice, Super-Charm-Tau factories in Novosibirsk and China,. . .
- we are happy to support more use cases
  - expected with *Turnkey Software Stack*

Documentation and Websites

- Main Web page
  - http://dd4hep.cern.ch
  - main entry point for DD4hep
- Github code repository:
  - https://github.com/AidaSoft/DD4hep
- Manuals
  - available from Github (`./doc`)
  - or main Web page
- Doxygen documentation
  - https://dd4hep.web.cern.ch/dd4hep/reference
    - or build from source

Summary and Outlook

- **DD4hep** is fully functional for many years
- **DDCore, DDG4 and DDRec** have long reached **production quality**
  - used for large scale Monte Carlo productions of ILD and CLICdp
  - DDCore and DDG4 (partly) used for FCC CDRs
- **DDAlign** and **DDCond** are more recent developments
  - full integration in iLCsoft still pending
- recent adoption of **DD4hep** by LHCb and CMS has triggered lots of activity
  - bug fixes, missing shapes, new features

Outlook

- will continue to improve and evolve DD4hep as a true community tool
- next step: implement **DDDigi** and integration in *Turnkey Software Stack*
- **new users are welcome**