EPJ Web of Conferences **245**, 02004 (2020) https://doi.org/10.1051/epjconf/202024502004
CHEP 2019

DD4hep a community driven detector description for HEP

Frank Gaede[1,*], *Markus* Frank[2,**], *Marko* Petric[2,***], and *Andre* Sailer[2,****]

[1]DESY, 22607 Hamburg, Germany
[2]CERN, 1211 Geneva 23, Switzerland

Abstract. Detector description is an essential component in simulation, reconstruction and analysis of data resulting from particle collisions in high energy physics experiments and for the detector development studies for future experiments. Current detector description implementations of running experiments are mostly specific implementations. DD4hep [1] is an open source toolkit created in 2012 to serve as a generic detector description solution. The main motivation behind DD4hep is to provide the community with an integrated solution for all these stages and address detector description in a broad sense, including the geometry and the materials used in the device, and additional parameters describing e.g. the detection techniques, constants required for alignment and calibration, description of the readout structures and conditions data. In these proceedings, we will give an overview of the project and discuss recent developments in DD4hep as well as showcase adaptions of the framework by LHC and upcoming accelerator projects together with the road map of future developments.

1 Introduction

A detailed and realistic description of the detector geometry and its material properties is an essential component for the development of almost all data processing applications in High Energy Physics experiments. This is particularly evident for the case of Monte Carlo simulations, where the exact knowledge of the position, shape and material contents of every detector component is crucial for the accuracy of the simulated detector response and underlying physics. For the subsequent processing steps of digitization and reconstruction, typically a different, higher level view onto the detector geometry is needed which includes information like subdetector components or measurement layers. This should ideally be created from the same source in order to avoid inconsistencies. The DD4hep [1] (Detector Description for HEP) software package is a generic geometry toolkit that builds on top of the two most widely used software packages in HEP: ROOT [2] and Geant4 [3]. Even though DD4hep was developed in the context of the linear colliders with other future accelerator projects in mind, it has from the start been designed to generically support the full experiment life cycle, such that it can continuously be used also beyond the project-planning phase or be adopted by running experiments.

*e-mail: frank.gaede@desy.de
**e-mail: markus.frank@cern.ch
***e-mail: marko.petric@cern.ch
****e-mail: andre.sailer@cern.ch

We will provide an overview of DD4hep and its main components in section 2, followed by a description of recent developments in section 3 and an outlook into future plans in section 4.

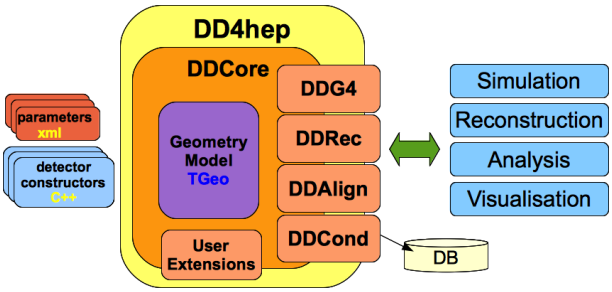

Figure 1. Schematic view of the component structure of DD4hep with the core component DDCore and optional extensions such as DDG4, DDRec, DDCond and DDAlign.

2 Overview

The DD4hep toolkit follows a component-based design (see Figure 1) that provides a high level of flexibility for the users. The core functionality is implemented in DDCore, where the geometry of the detector is represented in memory as a hierarchy of geometrical shape objects holding material properties implemented using ROOT's geometry system TGeo [4]. A parallel geometry tree with touchable *DetElements* and an extension mechanism allows storing of additional data structures in order to provide relevant physical properties of individual detector components such as alignment constants, measurement surfaces or visualization. The standard input format consists of compact xml files with parameters and *C++* detector constructor modules. Support for other input formats has also been implemented (see section 3.1). The DDG4 [5] component provides access to full simulations with Geant4. A large number of useful plugins for such applications exists, dealing with various input and output formats or handling of detailed associations of simulated hits to Monte Carlo truth particles. DDRec [6] adds the functionality for event reconstruction, such as measurement surfaces for tracking, dedicated high level sub detector descriptions or conversions between *cellID* and *position* of hits. Support for accessing conditions data and applying mis-alignment to detector elements is provided with DDAlign and DDCond [7].

3 Recent Developments

In this section we focus on new developments since DD4hep was last presented at CHEP 2018 [7].

3.1 Alternative input data sources

DD4hep was originally developed in the context of the linear collider studies and had adopted an input format with compact compact xml files with parameters and *C++* detector constructor modules used before in the community. At the same time, it has been from the start developed in a way that would allow the adaptation of other input sources for the detector geometry. The LHCb and CMS experiments, that had already implemented their own, different

ways of defining the detector geometry, have shown an increasing interest in using DD4hep. To facilitate their evaluation of DD4hep, dedicated plugins have been developed for both experiments that allow one to instantiate the DD4hep detector geometry from their pre-existing sources. Figure 2 shows examples of the CMS and LHCb detector models implemented in

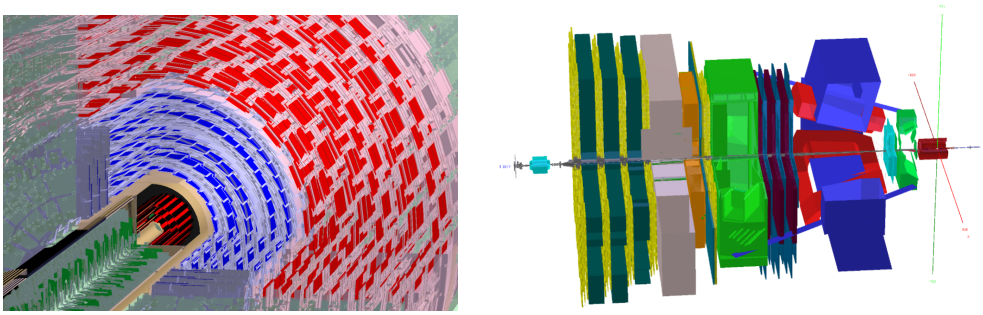

Figure 2. Left:Model of the CMS tracking system in DD4hep. Right: a model of the LHCb detector in DD4hep.

DD4hep. The plugin for LHCb reads the geometry parameters from a dedicated database, the one used for CMS reads the parameters from dedicated xml files before converting the geometry to the underlying TGeo representation. Both experiments have now decided to use DD4hep in the future [8, 9].

3.2 Reflections with left handed coordinates

Pairs of endcap detectors in HEP can be created either by simply rotating a copy of the corresponding *DetElement* by 180°, as is done for the linear collider detectors, or, as is the case in the CMS experiment, by reflecting the *DetElement* into a mirrored element with a left handed coordinate system. In the later case, every point is transformed as:

$$\vec{x}_r = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \vec{x}$$

In practice this implies creating new copies of volumes for all volumes in the hierarchy of the *DetElement* with left handed coordinates, placed at $-z$, as shown for an example volume in Figure 3. Such reflections are now implemented in DD4hep through a new detector type: *DD4hep_ReflectedDetector* that makes it very easy to define reflections in the compact xml format, as shown in Listing 1.

```
<detector id="EcalEndcap_ID+100" name="EcalEndcapB"
          type="DD4hep_ReflectedDetector"
          sensitive="true" sdref="EcalEndcapA" readout="EcalEndcapBHits">
</detector>
```

Listing 1. Example XML code for creating a reflected detector *EcalEndcapB* with left handed coordinates.

3.3 Additional shapes

The missing shapes provided by the ROOT TGeo that until recently have not been used in DD4hep have now been implemented, as they are needed in the CMS detector model:

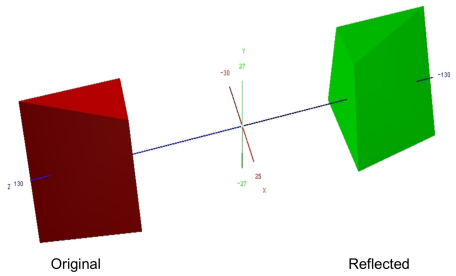

Figure 3. Example for a shape that is reflected (mirrored) into a left handed copy.

- **TGeoCtub**: A tube segment cut with 2 planes.
- **TGeoScaledShape**: A shape scaled by a scale transformation.
- **TGeoArb8**: An arbitrary trapezoid with less than 8 vertices standing on two parallel planes perpendicular to Z axis.

with their corresponding Geant4 shapes, used in DDG4.

3.4 Surfaces and optical photons

The correct simulation of optical photons in Geant4 requires additional information on the surfaces and optical properties of the volumes in the relevant detector elements. This information is not normally present in the geometry model, where typically only material properties are stored for every volume. Both Geant4 and ROOT-TGeo [1] provide the functionality to attach additional surface objects with optical properties to the volumes. This information can now also be provided as input in the compact xml format in DD4hep (see Listing 2) and suitable surface objects can be attached to the volumes in the detector constructor C++ code.

```
<opticalsurface name="/world/BubbleDevice#WaterSurface" finish="ground"
                model="unified" type="dielectric_dielectric">
  <property name="RINDEX"                coldim="2" values="2.03*eV 1.35 4.136*eV 1.4"/>
  <property name="SPECULARLOBECONSTANT"  coldim="2" values="2.03*eV 0.30 4.136*eV 0.3"/>
  <property name="SPECULARSPIKECONSTANT" coldim="2" values="2.03*eV 0.20 4.136*eV 0.2"/>
  <property name="BACKSCATTERCONSTANT"   coldim="2" values="2.03*eV 0.20 4.136*eV 0.2"/>
<opticalsurface>
```

Listing 2. Example XML code for defining an optical surface with properties.

If present in the TGeo geometry, the surfaces are then automatically translated to the Geant4 geometry in DDG4. All relevant optical physics processes are also available in DDG4: scintillation, Cerenkov and transition radiation, reflection, refraction, absorption and wavelength shifting. The complete treatment of optical photons and surfaces as defined in Geant4 is thereby now available in DDG4.

3.5 Compatibility with Python 3

DD4hep provides Python bindings for most of its core components via PyROOT [10]. Additional glue code facilitates the use within Python programs. This code has originally been

[1]The implementation of optical properties in ROOT had actually been triggered by the request to have this functionality in DD4hep.

implemented with Python 2.7. In order to facilitate the transition for users to Python 3 all Python code in DD4hep has been made compatible with both Python versions using the *python-modernize* [12] tool together with the *six* [11] compatibility library that is shipped with DD4hep. This is an important usability feature as it now allows users to decide whether to use Python 2 or Python 3 in their software ecosystem, a decision that often depends on many other packages and requirements.

4 Future Plans

With DDG4 and DDRec, DD4hep provides gateways to the simulation and reconstruction of HEP events. Adding a component that deals with the digitization of the simulated detector response before it can be reconstructed would be the logical next step in extending the functionality of DD4hep. This digitization component (DDDigi) should ideally be kept separate from the simulation step as much as possible, as this is important for the overall CPU and memory usage and a potential re-use of simulated data with different digitization criteria. This will naturally allow one to study the effects of detector segmentation, detector response to energy depositions, charge sharing and eventually the formation of hits and clusters. At the same time it needs to allow one to incorporate electronics effects, such as noise or cross-talk. The work plan for development of the DDDigi component would be to:

- develop a suitable data model that is consistent with experiments' models, i.e. that matches existing input and output data formats
- investigate the implementation of different digitization types:
  - detailed models for specialized studies, taking into account all known physics and electronics effects
  - simplified/parameterized models for bulk productions
- develop a palette of digitization plugins for typical subdetector types in a parameterized and flexible way that is valid for most readouts

5 Conclusion

DD4hep is fully functional for many years. DDCore, DDG4 and DDRec have long since reached production quality and have been used for large scale Monte Carlo productions of ILD and CLICdp. DDCore and partly DDG4 have also been used for the FCC CDRs. Other experiments have started to use or evaluate DD4hep. More recently developed components, such as DDAlign and DDCond, are yet pending integration in the experiments' frameworks in order to be fully exploited.

The recent adoption of DD4hep by LHCb and CMS has triggered a fair amount of activity, including the addition of missing shapes, new features such as the reflection with left-handed coordinates and, last but not least, bug fixes.

We will continue to improve and evolve DD4hep as a true community tool. The next planned step is the implementation of the DDDigi component and the integration into the Turnkey Software Stack [13] that is currently under development by the community.

6 Acknowledgements

This project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 654168.

References

[1] M. Frank, F. Gaede, C. Grefe and P. Mato, "DD4hep: A Detector Description Toolkit for High Energy Physics Experiments," J. Phys. Conf. Ser. **513** (2014) 022010.

[2] R. Brun and F. Rademakers, "ROOT: An object oriented data analysis framework," Nucl. Instrum. Meth. A **389** (1997) 81.

[3] S. Agostinelli *et al.* [GEANT4 Collaboration], "GEANT4: A Simulation toolkit," Nucl. Instrum. Meth. A **506** (2003) 250.

[4] R. Brun, A. Gheata and M. Gheata, "The ROOT geometry package," Nucl. Instrum. Meth. A **502** (2003) 676.

[5] M. Frank, F. Gaede, N. Nikiforou, M. Petric and A. Sailer, "DDG4 A Simulation Framework based on the DD4hep Detector Description Toolkit," J. Phys. Conf. Ser. **664** (2015) no.7, 072017.

[6] A. Sailer *et al.* [CLICdp and ILD Collaborations], "DD4Hep based event reconstruction," J. Phys. Conf. Ser. **898** (2017) no.4, 042017.

[7] M. Frank, F. Gaede, M. Petric and A. Sailer, "Conditions and alignment extensions of the DD4hep detector description toolkit," EPJ Web Conf. **214** (2019) 02042.

[8] C. Vuosalo, I. Osborne, "CMS Experience with Adoption of the Community-supported DD4hep Toolkit," These proceedings.

[9] D. Muller, "Gaussino - a Gaudi-based core simulation framework," These proceedings.

[10] W. Lavrijsen, "Python in the Cling World," J. Phys. Conf. Ser. **664** (2015) no.6, 062029.

[11] https://pypi.org/project/six

[12] https://github.com/python-modernize/python-modernize

[13] G. Ganis, A. Sailer, S.A. Graeme, P. Mato Vila, "Towards a Turnkey Software Stack for HEP Experiments," These proceedings