Using DD4hep through Gaudi for new experiments and LHCb

This content has been downloaded from IOPscience. Please scroll down to see the full text.
(http://iopscience.iop.org/1742-6596/664/7/072012)

View the table of contents for this issue, or go to the journal homepage for more

Download details:

IP Address: 137.138.93.202
This content was downloaded on 09/03/2016 at 08:02

Please note that terms and conditions apply.
Using DD4hep through Gaudi for new experiments and LHCb

M Clemencic and A Karachaliou
CERN, CH-1211 Genève 23, Switzerland
E-mail: marco.clemencic@cern.ch

Abstract. The LHCb Software Framework Gaudi is a C++ software framework for HEP applications used by several experiments. Although Gaudi is extremely flexible and extensible, its adoption is limited by the lack of certain components that are fundamental for the software framework of an experiment, in particular a detector description framework, whose implementation is delegated to the adopters.

To enable future experiments to quickly adopt Gaudi, we integrated the DD4hep toolkit in the existing software framework, and, as a proof of concept, we used it with the LHCb software applications, from simulation to reconstruction and analysis.

We will describe how the DD4hep toolkit can be used by a new experiment, as well as how we can migrate an existing detector description framework to the new toolkit.

1. Introduction
The Gaudi architecture and framework[1,2] are designed to provide a common infrastructure and environment for simulation, filtering, reconstruction and analysis applications. Initially developed for the LHCb experiment, Gaudi has been adopted and extended by several other experiments.

DD4hep[3,4] is a generic detector description toolkit, developed in the context of the AIDA Project[5], where the geometrical representation provided by ROOT[6,7] is the main source of information. In addition, DD4hep provides the automatic conversions to other geometrical representations, such as Geant4[8,9], and the convenient usage of these components without reinventing existing functionalities.

Gaudi did not provide an integrated detector description toolkit, and DD4hep looks like a natural candidate to fill the gap, so we developed the necessary components to provide integration between the two software packages.

We first addressed the use case of a new experiment wishing to adopt Gaudi as its software framework, then we considered the use case of an existing experiment already using Gaudi and wishing to migrate its existing detector description to DD4hep.

2. Integration between Gaudi and DD4hep
To use DD4hep from Gaudi applications we decided to avoid the introduction of an abstraction layer hiding the DD4hep API, because it would have introduced unneeded complexity in the early testing phase.
The first constraint we had to address was that Gaudi needs to control the lifetime of the main DD4hep component, so we developed a simple Gaudi service that instantiates the DD4hep main object and initializes it with configuration options passed to it via the Gaudi configuration system. At this stage we used the DD4hep built-in XML detector description format.

The other constraint we had to address was the message reporting channel. Gaudi provides a Message Service that is used by every component in Gaudi for message reporting, so that we can control the verbosity of the application in one single place, and use specialized message services to, for example, transmit the messages over network connections. DD4hep has a similar machinery, so we used the provided methods to replaced the default messaging functions with new functions that use the standard Gaudi Message Service.

Figure 1. Diagrams of the Gaudi architecture and DD4hep, with, starred, the new components developed to integrate the software packages.

3. Writing a new experiment framework with Gaudi and DD4hep
As an extension to the examples already available in Gaudi we developed a mini-framework meant to mimic, on a very small scale, the work a new experiment would need to do to adopt
Gaudi and DD4hep.

The developers of an experiment framework would need to provide the following elements:

- **an event model** a set of classes used to describe the physics events the experiment will collect and study
- **a detector description** classes used to instantiate the hierarchy of detectors and volumes, and the configuration files used to tune it.

With the event model and the detector description defined, they can work on the various applications of the analysis workflow, for example:

- simulation
- digitization
- reconstruction
- analysis

While digitization, reconstruction and analysis should be implemented as Gaudi applications, the simulation can be steered either via Gaudi or, if there are no special requirements, directly via Geant4. For our mini-framework example we decided to steer the simulation directly from Geant4.

Thanks to the methods provided by DD4hep to run Geant4 and to map the simulated ‘hits’ to detector elements in the detector description, we managed to quickly produce the code for the simulation, digitization and reconstruction phases of the analysis workflow. The analysis phase was not implemented because it was not very much connected to the main goal.

4. **Migrating an existing experiment detector description to DD4hep**

For an experiment already using Gaudi as underlying framework, but a different detector description toolkit, it is possible to migrate to DD4hep in different ways.

The data flow from configuration files to memory representation of a detector description can be summarized with a graph like the one in Figure 2.

![Diagram](image)

**Figure 2.** Data flow of a generic detector description toolkit from configuration files to memory representation.

When addressing the migration we are faced with two parallel data flow paths (see Figure 3) and we need to divert the data flow from one path to the other. It can be done in several ways,
with more or less work depending on the strategy and the implementation details of the old data flow. Depending on how much entangled are the old parser and factories, the cheapest option can be to keep configuration and parsing of the old toolkit and replace the factories so that they populate the DD4hep memory representation of the geometry. Alternatives strategies could be

**Figure 3.** Example of a migration strategy. We can redirect the flow of information from the old detector description to DD4hep in several ways. For example, as showed in the graph, we can replace the old factories with new ones that fill directly the DD4hep memory representation.

to replace the parser of DD4hep to use one that understands the old configuration files, but it will still require the writing of the factories for DD4hep.

5. The LHCb use case
The LHCb experiment developed its own detector description toolkit because there was no viable alternative. With the release of DD4hep we wanted to evaluate the possibility of a migration.

The analysis of the implementation details of the data flow of the LHCb detector description toolkit showed that there is a fundamental difference in the way the two main ingredients of a detector description (geometry and detectors hierarchy) are managed in LHCb and in DD4hep. LHCb uses two independent hierarchies with weak bindings between them, that are established only at runtime, after the memory representation has been populated. On the other hand, DD4hep uses tight bindings between the detectors hierarchy and geometry, which are established at construction time.

The difference between the two models, although not impossible, requires too much work at different levels (framework, detector and analysis developers) to be feasible in the short term, but it could still be taken into account as a possibility for a foreseen evolution of the LHCb software for the detector upgrade.

6. Conclusions
With the implementation of the components required for the integration of Gaudi with DD4hep and with the implementation of a mini-framework example and prototype, we allow new experiments to adopt more easily Gaudi as their underlying software framework. The mini-framework example will be soon made publicly available from the Gaudi web page.
We also studied possible migration paths from existing detector description toolkits to DD4hep for experiments based on Gaudi.

We concluded that a migration of the LHCb detector description to DD4hep could not be done without major disruptive changes to the existing software.

References