The LHCb Detector Description Framework

G. Barrand[1]*, I. Belyaev*[2]*, P. Binko*[3]*, M. Cattaneo*[3]*, R. Chytracek*[3,4]*, G. Corti*[3]*, M. Frank*[3]*, G. Gracia*[3]*, J. Harvey*[3]*, E. van Herwijnen*[3]*, B. Jost*[3]*, I. Last*[3]*, P. Maley*[3]*, P. Mato*[3]*, S. Probst*[3]*, F. Ranjard*[3]*, A. Tsaregorodtsev*[3]

[1] Laboratoire de l'Accélérateur Linéaire (LAL), Orsay, France
[2] Institute for Theoretical and Experimental Physics (ITEP), Moskva, Russia
[3] European Laboratory for Particle Physics (CERN), Genéve, Switzerland
[4] Technical University Kosice, Slovakia[‡]

Abstract

A coherent O-O framework has been developed for describing the LHCb experimental apparatus to be used by the simulation, reconstruction and analysis applications. This framework is used for the realization of the detector data service, which is one of the main components of the GAUDI architecture in use in LHCb. Examples of detector description data are: the geometry and alignment parameters, the electronics calibration and their organization, the environmental parameters needed for the data processing, etc. The detector description data are made available to the physics algorithms through a number of transient objects. These transient objects are hierarchically organized and are a view of a global persistent representation, which is currently implemented using XML and takes into account versioning and a time validity interval. Other representations of the data are also provided to domain specific frameworks such as Geant4 and visualization components. We present the use-cases that have driven the development, the main design choices, and details of the first implementation of the framework. Finally we describe how it is customized to provide the specific needs of the various LHCb sub-detectors.

Keywords: LHCb, GAUDI, framework, geometry, XML

1 Introduction

One category of data objects identified in the GAUDI architecture [1] that is required by the *Algorithms* in all event data processing applications are the so called *detector data* objects. The detector data fully describe and qualify the detecting apparatus and are used to interpret the event data. They consist typically of the description of the detector in terms of its geometry, the calibration and alignment constants, the environmental parameters such as temperatures and pressures, etc. A framework has been developed to manage in a coherent way all the detector data and to ease the implementation of the specifics of the sub-detectors in LHCb. Use cases have driven the design of the framework. A very important one has been that users want to have a single detector description for all applications (simulation, reconstruction, event display...) Therefore, the information in the database should be the union of all the information needed by all applications; the level of detail should be selected by the client application. The developer using the framework should only concentrate with the specifics of his/her sub-detector and leave the commonality to be provided by the framework itself.

2 Detector description overview

The choice in the GAUDI architecture of separating the transient from the persistent representation of data objects applies also for detector data. The overall structure of the detector description is

[‡] This work was supported by the grants Nr. 1/5086/98 and 1/6278/99 of the Slovak Scientific Grant Agency

shown in Figure 1, where we can see how *Algorithms* access detector data through a transient representation in the transient detector store and how this store is populated from the detector description database. Conversion to other views (representations) is carried out by *Conversion Services*. Examples of other representations are graphical views, Geant4 geometry description, etc.

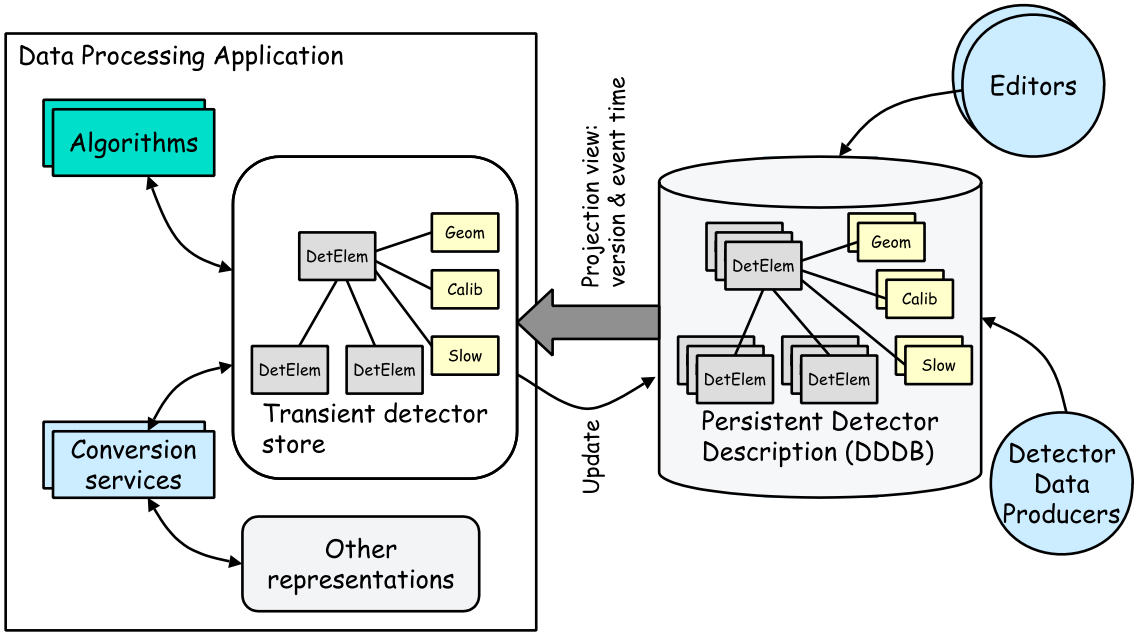

Figure 1: Detector description overview

The detector database includes a physical and a logical description of the structure of the detector. The physical description covers dimensions, shape and material of the different type of element from which the detector is constructed as well as information specific to each element which is actually manufactured and assembled into a detector. Both active and passive elements should be included. The description of active ones should allow for the specification of deficiencies (dead channels), mapping to electronic channels, alignment corrections, calibration of electronic responses, etc. The logical description provides two main functions.

- Simplified access to particular parts of the physical detector description via a hierarchical description which a given detector setup is composed of various sub-detectors, each made up of a number stations, modules or layers, etc. There is a simple way for a client to use this description to navigate to the information of interest.
- Provide a means of detector element identification. This allows for different sets of information which are correlated to specific detector elements to be correctly associated with each other.

In a detector description, the definition of the detector elements and the data associated to their physical description may vary over time, for instance due real or hypothetical changes to the detector. It is foreseen that such changes are recorded as a different version of the detector element. Additionally, it is also foreseen to capture, for an entire description, a version of each of detector elements and associate a name to that set. This is similar to the way CVS allows one to tag a set of files so that one does not need to know the independent version numbers for each file in the set.

3 Transient store structure

The transient representation of the detector description is implemented as a standard GAUDI data store. The data objects are organised in a hierarchical tree structure. The transient store is populated on the demand of *Algorithms* with data objects of a given version and whose validity include the time of the event currently being processed. It is the responsibility of the *Detector Data Service* to make sure the elements in the transient store are up to date.

At present, there are tree top level catalogs in the transient store: detector setups (logical structure), geometry elements, and materials. Other catalogs will be added when needed.

Links between objects in different catalogs, e.g. the association of a solid to a material, a detector element to its geometry information, etc. are resolved at runtime when needing to traverse them. For example, a material object corresponding to a detector element will only be loaded when needed to know some material properties of such detector element.

3.1 Detector Setup

A detector setup is described using a tree of detector elements. More than one setup can be loaded concurrently in the transient store. The complete identification of a detector element is by a name in the form ”/dd/Structure/LHCb/Muon/Stations/Station1”. The generic *DetElement* class may be specialized in order to provide answers to concrete questions from the *Algorithms*.

3.2 Geometry

The detector geometry is described by a hierarchy of *logical* and *physical* volumes, following very closely the ideas of Geant4 [2], ensuring an easy conversion to Geant4 objects for the simulation application besides other also needed conversions. A logical volume is an unplaced volume described in terms of a solid with a shape or a boolean operation of solids, a material and a number of placed sub-volumes (physical volumes). The simplified class diagram showing the relationships between logical and physical volumes, solids and materials is shown in Figure 2.

Transient geometry objects provide basic geometrical functionality to *Algorithms*. Examples are local to global coordinate transformations, finding out if a given 3D point is inside a detector volume, returning the complete hierarchy of volumes containing a given point, etc.

3.3 Materials

Materials are needed by logical volumes. They are defined as isotopes, elements or mixtures. Elements can optionally be composed of isotopes. Composition is always done by specifying the fraction of the mass. Mixtures can be composed of elements or other mixtures. For a mixture the user can specify composition either by number of atoms or by fraction of mass.

4 Persistent representation

The current implementation is based on text files whose structure is described by XML (eXtendible Markup Language) [3]. XML is an application independent format for describing data. It has recently acquired wide support and is an industry standard issued by the W3 Organization. XML files can be understood by humans as well as computer programs. XML allows to define a custom markup language to describe a specific application domain.

The structure of an XML document can be easily represented as a tree of XML elements, their attributes and content. This tree structure fits very well into the hierarchical navigation

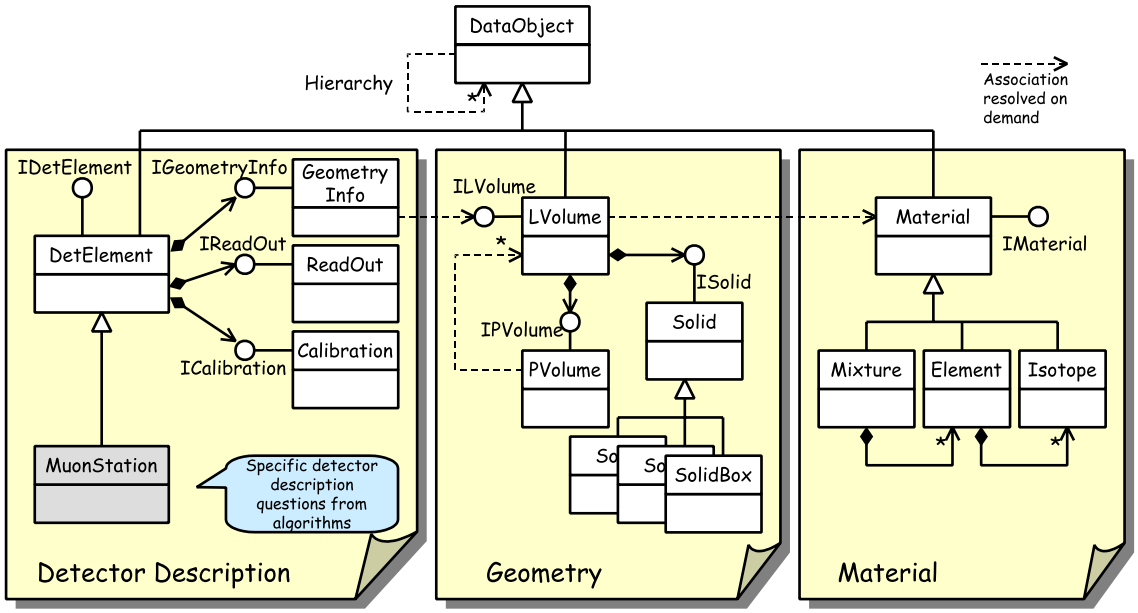

Figure 2: Simplified class diagram

schema used by OO-systems. The basic problem is the proper translation from the source OO-language (in our case transient C++ data structures) into the corresponding XML form (persistent data). The translation rules used in the current implementation are shown in Table 1.

Table I: Mapping C++ to XML

C++	XML
Class	XML Element
Class data members	Element attributes
Composition	Element embedded in Content Model
Reference	Element referenced with a hyperlink

Using these rules it was relatively easy to produce an XML representation of the transient detector description. The XML representation allowed us to create a very flexible XML based database with hyperlinks.

4.1 XML converters

Following the GAUDI philosophy, the XML converters have been built on top of the SAX [4] interface, one per data type. The current implementation provides read-only XML data but work has been started to allow bi-directional data flow. The reason is that SAX is optimal for parsing and analyzing XML data but the XML structure itself is not preserved.

4.2 Hyperlinks

XML allows to define hyperlinks in a way similar to HTML. This feature helps to split the XML based database into multiple files that can be maintained independently of each other, minimizing coupling between developments of different sub-detectors. On the other hand is possible to define in XML the persistent references between different pieces of data in a way similar to C++ or C.

One can thus refer to objects inside the same file or objects located somewhere on the network. It is possible to use standard internet protocols like http or ftp to retrieve them.

5 User Customization

The specific detector description can be made available to algorithms by customizing the generic detector element. Customization is done by inheritance of the *DetectorElement* base class. The sub-detector specialist can provide specific answers to algorithms based on a combination of common parameters and functionality (general geometry, material, etc.) and some specific parameters. The specialist can ”code” the answer by using the minimal number of parameters specific to the detector being described. For example, an *Algorithm* may need the local position of a calorimeter cell knowing its cell ID. In this case, probably, a simple parameterized equation in a *detector element* method can give the answer.

6 Current status

The first version of the detector description framework has been released to the LHCb collaboration at the end of November. Several sub-detector groups have started to implement their geometry description using the framework with success. As a result, we have received a sizeable amount of user feedback in terms of unwanted features, suggestions, and new functionality requests. We expect to complete the second iteration rather soon by correcting the unwanted features and enhancing the functionality.

References

1 G.Barrand et al., *GAUDI - The Software Architecture and Framework for building LHCb Data Processing Applications*, in this proceedings (see also `http://lhcb.cern.ch/computing/Components/html/GaudiMain.html`).

2 S. Giani et al., *GEANT4: An Object-Oriented Toolkit for Simulation in HEP*, CERN/LHCC/98-44 (see also `http://wwwinfo.cern.ch/asd/geant4/geant4.html`).

3 The World Wide Web Consortium, *Extensible Markup Language (XML)*, `http://www.w3.org/XML`

4 *SAX - The Simple API for XML*, `http://www.megginson.com/SAX/index.html`