Overview of detector descriptions

C. Cheshkov[a]
for the ALICE, ATLAS, CMS and LHCb Collaborations

[a] *Gesellschaft fur Schwerionenforschung mbH (GSI)
Darmstadt, Germany*

Abstract
The present overview describes the detector description frameworks implemented by the four LHC experiments. The main reviewed items include the detector geometry implementation, the usage of alignment constants in the simulation and the reconstruction, and the storage and retrieval of the alignment constants. The review concludes with a brief comparison of the experiment frameworks.

9.1 Introduction

Given the complexity of the LHC experiments and the required tracking precision, the alignment of the detectors is a challenging goal. During the commissioning and operation of the detectors several tasks are of primary importance, namely reconstruction and simulation with a misaligned detector geometry, developing and training of track-based alignment algorithms, and estimation of the impact of the misalignment on the physics. These tasks necessitate reliable and user-transparent software frameworks in order to deal with the detector geometries including misalignments. For this purpose, all the LHC experiments have developed detector description frameworks. These frameworks are rather sophisticated packages with a wide range of functionality. Therefore it is very important to keep track of their current status as well as to exchange information for further developments in this field. This overview aims to present briefly the frameworks of the four LHC experiments by highlighting the common and experiment-specific items.

The content of the overview is the following. A common terminology is defined in Section 9.2. Sections 9.3 to 9.6 describe the frameworks employed by each experiment. A discussion of the similarities and the differences between the experiments is given in Section 9.7.

9.2 Terminology

Given the variety of the terms used in the LHC experiments and their sometimes controversial meaning, the following terminology has been adapted for the needs of the present overview.

– *Logical volume (node)*: Representation of a detector element à la GEANT [1], i.e., one logical volume represents one or many real detector elements. A set of logical volumes forms the detector geometry hierarchy. As the logical volume can refer to several real detector elements, it can not be misaligned.
– *Physical volume (node)*: Representation of a single (unique) real detector element. It is coupled to corresponding logical volume. The physical volume is used to handle the misalignment and any other detector-element-specific information.
– *Alignment constants*: The objects which are actually stored in the Condition Database and used to misalign the detector geometry.
– *Default transform*: The detector-element position and orientation in case of the ideal (no misalignments) geometry. By definition it coincides with logical and physical volumes.
– *Delta transform*: A correction to the default transform in case of a misaligned geometry. The delta transform applies only to the physical volumes.
– *Geometry overlaps*: The overlaps between physical volumes at the level of the detector simulation in the case of geometry misalignments. These overlaps should be distinguished from the overlaps caused by improper geometry implementations in the case of an ideal geometry.

9.3 The detector description framework in the ALICE experiment

The detector description in the ALICE experiment is based entirely on the ROOT Geometrical Modeler

(TGeo) [2]. TGeo is a framework for building, browsing, tracking, and visualizing detector geometry. Currently it is being used in many high-energy physics experiments. It relies on a hierarchical model of logical volumes (like in GEANT3 [1] and GEANT4 [3]) and is independent of the transport Monte Carlo. The use of TGeo has the following basic advantages.

1. It decouples the detector geometry description from the transport Monte Carlo. Thus it enables the use of the so-called Virtual Monte Carlo [2], i.e., the detector simulation with different transport Monte Carlo (GEANT3, GEANT4 or Fluka [4]).
2. The same geometry is used for tracking, reconstruction, and visualization.
3. An advantage of using ROOT features related to bookkeeping, I/O, histograming, browsing, and GUI.

In TGeo the misalignments are introduced at the level of the so-called TGeo Physical Node (PN). As defined in Section 9.2, PN describes unique detector element. It is fully identified by a path which consists of the names of logical nodes. Each logical node is positioned in its mother logical node (container) by means of a local matrix (Fig. 9.1).

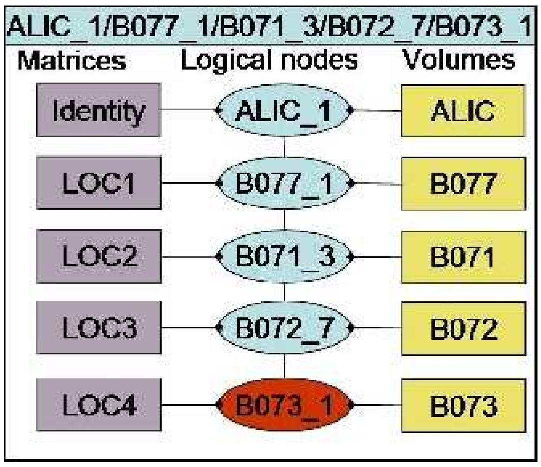

Fig. 9.1: Example of TGeo geometry hierarchy. The PN path is shown on the top

The PN has a pointer to the last logical node in the path. Thus the global matrix which transforms from the local reference system of the PN to the global reference system is given by:

$$GLOBAL = LOC_1 * LOC_2 * ... * LOC_n \qquad (9.1)$$

where LOC_i are the local matrices used to position the Logical Nodes inside their containers. PNs can be created for any Logical Node at any level in the geometry hierarchy. Since the entire ALICE detector geometry contains $\approx 2.5 \times 10^6$ detector elements, the PN are created on demand for selected alignable detector elements. PNs are misaligned by changing the local matrix of the last Logical Node in the path (LOC_n). The default matrix corresponding to the ideal position and orientation of the PN is backup. The misalignment is automatically active for the transport Monte Carlo or any other access to the TGeo geometry. All PNs can be addressed by unique symbolic names created during the geometry initialization. This allows for a proper identification of the alignable detector elements disregarding a given geometry implementation or version.

The ALICE alignment constants are simple ROOT objects which contain the misalignment information in a form of Delta Transforms [5]. The transform is defined in the global reference system, i.e., it is applied on the left side in Eq (9.1). The reference system is chosen to make the misalignment independent of a given geometry implementation (matrices LOC_i in Eq. (9.1)) and to facilitate the use of alignment constants during the alignment procedures based on tracks. Each alignment constant contains also a symbolic name that identifies the TGeo PN to which it is applied. In addition, the alignment constants related to sensitive detector elements have a unique (ALICE-wide) integer identifier, which allows for a fast navigation during alignment procedures. The ALICE alignment constants are applied on several levels in the geometry hierarchy (e.g., sensitive detector elements, superstructures and entire detectors) via a TGeo interface.

The alignment constants described above are stored in the ALICE Offline Condition Database (OCDB) [6]. The offline condition files are ROOT files available to applications running on the Grid. For that purpose, the condition files are registered into the AliEn file catalogue with their corresponding logical names and are stored in a Grid storage element. In ALICE the condition data is uniquely identified by three parameters:

– Logical path (for example, 'TPC/Align/Data', 'TRD/Align/Data')
– Run range validity
– Version.

It is worth noting that the ALICE offline condition data are accessed only once per simulation or reconstruction job. Within the ALICE OCDB framework, the alignment constants are stored as one array per detector. The detectors store their arrays in separate files contained in the corresponding detector folders (as shown above). The ideal detector geometry is stored as a single object (TGeo manager object) in the same OCDB.

During the start-up of a simulation or reconstruction job, the alignment constants are read from the

OCDB and applied to the TGeo geometry. This is done in a user-transparent way using the same code for both simulation and reconstruction. Any further access to the detector geometry (ideal or misaligned) is done only via a TGeo interface. In some cases misalignments of detector elements situated on two or more geometry hierarchy levels have to be introduced. Then the alignment constants are applied to the TGeo geometry in order of their geometry level, namely from the highest level to the lowest.

Frequently the application of detector misalignments leads to volume overlaps. The latter can be inspected and visualized by means of the so called TGeo Overlap Checker. In general, two basic methods to diminish the effect of the overlaps are used:

- Replacing a set of overlapping TGeo volumes by TGeo Assemblies. The TGeo Assembly is a logical union of two or more volumes and does not have its own shape.
- Application of the misalignments on high-level structures. Big movements are applied on high-level structures, while the residual small movements are left at the level of primitive detector elements.

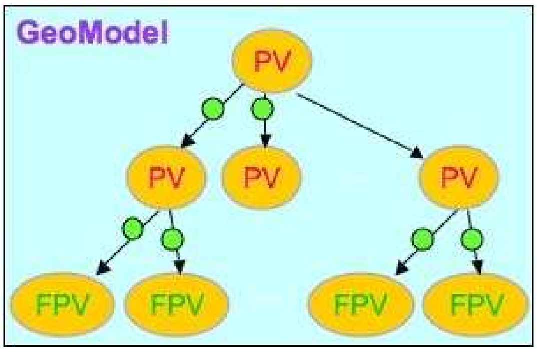

Fig. 9.2: The GeoModel geometry hierarchy. PV - Physical Volume, FPV - Full Physical Volume. The Alignable Nodes are shown as small circles on the connection arrows.

9.4 The detector description framework in the ATLAS experiment

The ATLAS geometry implementation is based on the GeoModel [7]. The GeoModel is a general-purpose package used for the description of various geometry structures. Its kernel contains a toolkit of geometry primitives (shapes, materials, etc.). The geometry description within the GeoModel relies on a hierarchical structure of Physical Volumes (Fig. 9.2).

The Physical Volumes are attached to each other by Nodes. The Nodes encapsulates the transformation used to place a Physical Volume inside its parent. The support for geometry misalignments is based on special Nodes, called Alignable Nodes. An Alignable Node contains not only the Default Transform of the ordinary Node, but also the Delta Transform which represents the misalignment. Alignable Nodes can be created at any level in the geometry hierarchy. In order to handle the access to the position and orientation of the geometry elements, the GeoModel uses so-called Full Physical Volume objects. These objects calculate and cache transformations from the local to the global reference frame. In case an Alignable Node from a higher level in the hierarchy is modified, the Full Physical Volume cache is invalidated and the transformation is recalculated upon the next access to that particular volume.

In ATLAS the alignment constant is defined as the Delta Transform of an ‘Alignable Node’ [8]. For primitive detector elements, the constants are defined in the local reference frame, and for the high-level structures in the global reference frame. Two types of alignment constants are used. The first one represents a rigid body transformation (rotation and translation). It is implemented via the CLHEP HepTransform3D class [9]. The alignment constant is a container of a HepTransform3D object and an identifier. The latter is a common ATLAS-wide unique identifier of detector elements and high-level structures. The alignment constants are applied directly to the detector geometry at several levels in the hierarchy. The second type of alignment constants is used to describe some fine corrections (e.g., module distortion, wire sag). The constant consists of a vector of float numbers and therefore its interpretation is detector-specific. The fine correction constants are not applied at the detector geometry level, but in the reconstruction at the time the tracks are known.

For the Offline Condition Database (COOL) the ATLAS experiment makes use of the same persistent technology as for the event data [10]. The alignment constants are written using the POOL [11]. The COOL database records the interval of validity (IOV) and the reference to the POOL file. The clients register callbacks on the conditions data object. Then IOV services take care of loading new data if IOV changes and trigger callbacks. The ideal detector geometry description is stored in a separate database. The GeoModel detector geometry is constructed starting from primary objects stored in the database. In this way, the ideal detector element positions and orientations (Default Transform) are initialized. The conditions data which handle the alignment constants contain a tag pointing to the database entry with the corresponding geometry description version.

The GeoModel detector geometry is a common source for both the GEANT4 simulation and the reconstruction. Moreover, the misalignment infrastructure is identical in both cases.

Figure 9.3 shows two example distributions in the case of a simulation with a misaligned detector geometry followed by a reconstruction with the ideal geometry and vice versa. During the simulation and the reconstruction, the detector geometry is handled by a Detector Manager object. This object transfers the alignment constants from the Condition Database to the GeoModel Alignable Nodes. The misalignments are updated via callbacks or explicit calls during the geometry initialization. In order to facilitate and speed up the retrieval of the detector geometry during the reconstruction, the Detector Manager provides an access to Detector Element objects. These objects point to the corresponding GeoModel Full Physical Volumes and cache all the derived quantities. The reconstruction job accesses the geometry information via Detector Elements only.

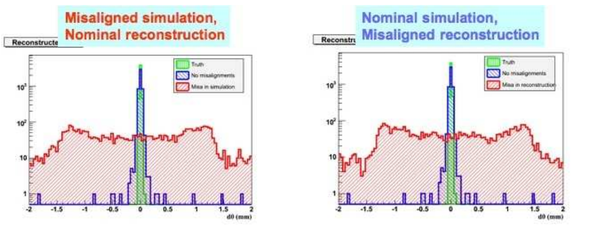

Fig. 9.3: Example distributions in case of a simulation with a misaligned detector geometry followed by a reconstruction with the ideal geometry (left) and a simulation with the ideal detector geometry followed by a reconstruction with a misaligned detector geometry (right).

The simulation with misaligned geometry throws up some extra challenges in order to deal with volume overlaps. The potentially dangerous overlaps are avoided by allowing for enough clearance between various detector elements. This is done mostly by resizing and reshaping of detector-element envelopes. Some services which are not of primary importance for a correct detector simulation are artificially thinned or moved. The detector misalignments are facilitated also by using alignable nodes at several levels in the geometry hierarchy. The approach allows for larger movements of big structures (detectors, subsystems, etc.) and smaller movements of primitive structures (e.g., modules).

9.5 The detector description framework in the CMS experiment

The misalignment framework in the CMS experiment is based on a dedicated set of objects (called AlignableTracker and AlignableMuon). These objects map the tracker and the muon system geometry hierarchies from the high-level structures down to the sensitive volumes used by the reconstruction (Fig. 9.4).

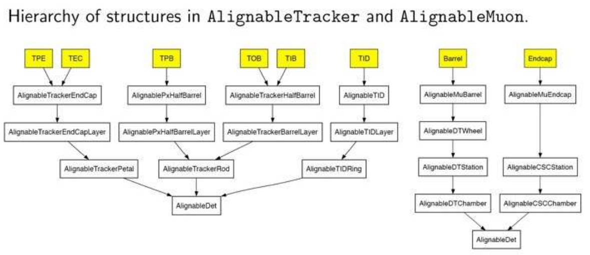

Fig. 9.4: The CMS detector geometry description based on AlignableTracker and AlignableMuon objects

The movements of the high-level structures are propagated through the corresponding detector hierarchy. The sensitive volumes are identified also by a unique CMS-wide identifier. The framework provides an access to the Alignment object, where the latter represents the global position and orientation and the identifier of a sensitive volume. The geometry misalignment procedure consists in movements applied at various geometry hierarchy levels and propagated down to the sensitive volumes. The users can apply custom misalignments by means of a set of pre-defined parameters stored in special configuration files. In order to study the impact of the detector misalignment of the reconstruction quality and train alignment algorithms, several configuration files ('misalignment scenario') are prepared for general usage. The configurations are supposed to describe the expected detector misalignments as known from the engineering design or from the detector geometry survey.

The alignment constants are stored as Alignment objects. As mentioned above, the objects contain global position and orientation of sensitive volumes (Default Transform + Delta Transform using the common notations defined in Section 9.2) and the unique identifier. The same objects also contain all the alignment (or misalignment) related information, like, for example:

- the applied misalignment scenario,
- the alignment algorithm used to produce the alignment constants (e.g., survey data, hardware alignment, or track-based alignment).

The CMS Condition Database is based on POOL. Alignment objects provided by AlignableTracker and AlignableMuon are stored in the database. Different instances in the database have different tags (string defining content) and different intervals of validity. The ideal geometry description is stored in a XML format.

The detector misalignment is performed in different ways for the simulation and the reconstruction. During the simulation, the Alignment objects are read from the condition database and applied to the detector geometry via an interface to GEANT4. At the reconstruction level, the Alignment objects are retrieved by the same tools and then used to construct a reconstruction geometry. Thus the detector misalignments are applied automatically just before the start of the global reconstruction, while the reconstructed space-point positions are converted from the local to the global reference frame. A custom misalignment can thus be performed on any simulated sample of events. Despite some discrepancies at the detector-element edges, the approach presented above turns out to be quite satisfactory for detector misalignments (or alignments) at the global level.

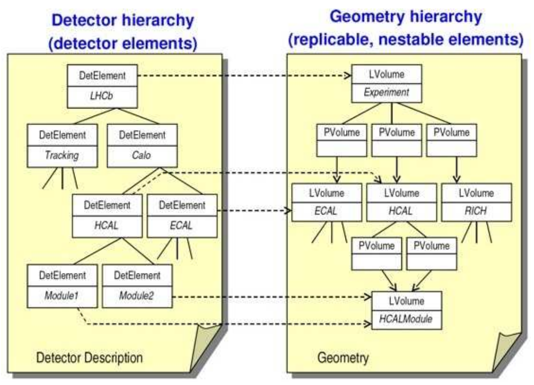

Fig. 9.5: The Geometry and Detector structures. The solid lines represent the geometry hierarchy within the structure. The dotted lines show the connection between the Detector Elements and their corresponding Logical Volumes.

9.6 The detector description framework in the LHCb experiment

In the LHCb experiment the detector geometry implementation resides in the Detector Description DataBase [12]. It is accessed via a transient detector data store based on the Gaudi package [13]. The LHCb detector geometry consists of two hierarchical structures - Geometry and Detector structure. The Geometry structure is a classical example of a GEANT-like geometry implementation. It consists of Logical Volumes with certain shape and material. The Logical Volumes can be replicated and nested within higher-level volumes. Thus the detector geometry hierarchy is formed by positioning of volumes within volumes. The Detector structure is coupled to the physical structure of the LHCb layout. Its hierarchy consists only of the 'interesting' Detector Elements (called Physical Volumes in the common notation). Each Detector Element corresponds to a real detector component and has knowledge of

- its position and orientation in the global and parent reference frames,
- its exact place in the geometry hierarchy,
- the list of its daughter volumes which are not necessarily detector elements.

A schematic view of the Geometry and Detector structures is given in Fig. 9.5.

The detector misalignments are applied through the Detector structure. They are handled by the corresponding Detector Elements in the following way:

- Combine the local misalignment with the local transformation matrix in order to obtain new local position and orientation of the Detector Element.
- Use the link to the parent volumes to calculate the global position and orientation of the Detector Element after the misalignment was applied.
- Use the links to the daughter volumes to propagate the misalignment down the Detector structure.

It is worth noting that the misaligned detector description is automatically available to the user.

The LHCb alignment constants are defined as Delta Transforms of Detector Elements in their parents' reference frame. Physically the constants are represented as rotations about pivot points followed by translations. The alignment constants contain a unique Detector Element identifier, which in general is subdetector-specific. The current implementation of detector misalignments allows the alignment constants to be updated at run-time. In this case, a special update manager service propagates the misalignment changes to all the necessary parts of the detector description. The run-time update of the alignment constants offers the possibility to easily develop iterative alignment algorithms. Owing to the flexibility of the detector description framework, the alignment constants can be generalized independently of their internal representation (e.g., Euler angles, matrix).

Like ATLAS and CMS, the LHCb Condition Database relies on POOL. The alignment constants are stored as COOL database records in the form of XML strings. The records also contain an interval of validity and a version number. Loading of the alignment constants in a given job is handled via the Condition Database Manager. The Gaudi transient store is created on demand.

Both the simulation and the reconstruction make use of the detector description and the misalignment framework in the same way. However, the misalignment in the simulation has several restrictions. First, the misalignments can lead to volume overlaps. Second, the construction of the GEANT4 geometry from the position and orientation of the volumes given by Detector Element objects is done at the initialization of a simulation job. Therefore there is no possibility to apply run-time-dependent misalignments and the simulation always runs with one set of constants.

9.7 Comparison of the experiment frameworks

Despite the quite different implementations and experiment-specific features, all four LHC experiments share common ideas and procedures used to build their detector description frameworks. In this section, we compare and discuss the basic building blocks of the experiment frameworks.

- Detector geometry implementation. All the experiments except CMS use external general-purpose packages to describe their detector geometries. Each experiment developed a set of tools in order to provide a robust and user-transparent way to access and modify the geometry information. In this sense, ALICE is a bit more specific, since the entire ALICE offline software is based on ROOT and therefore the geometry information is handled by direct calls to the ROOT geometrical package (TGeo).
- Geometry misalignment. The misalignments in the detector geometries are introduced via hierarchies of Physical Volumes. In the case of ALICE and LHCb, these hierarchies are closely connected to corresponding hierarchies of Logical Volumes.
- Alignment constants. There are two main points related to the alignment constants representation: the type of the transformation (Delta Transform or Delta + Default Transform) and the reference system in which the transformation is defined. ALICE, ATLAS and LHCb use alignment constants defined as Delta Transforms, while CMS stores the misaligned position and orientation of the detector elements (e.g., Delta + Default Transforms). The choice of the reference systems varies from experiment to experiment depending on the geometry implementation and the way the misalignments are applied in the simulation, reconstruction, or track-based alignment procedures.
- Condition databases. All the experiments except ALICE rely on the commonly developed POOL package. The alignment constants are stored as COOL database records in XML format. ALICE has developed its own condition database infrastructure based on ROOT files stored on the Grid.
- Misalignment in the simulation and the reconstruction. All the experiments use the same framework in both the simulation and the reconstruction. In the case of ATLAS, CMS and LHCb, the detector geometry is interfaced to GEANT4, while ALICE profits from the direct TGeo navigation which is independent of the transport Monte Carlo. It is worth noting, that the LHCb detector description framework allows for run-time misalignment changes and therefore can be used directly in case of iterative track-based alignment approaches.
- Geometry overlaps. Several different approaches to dealing with volume overlaps have been employed by the experiments — adding enough clearance between volumes, resizing of service volumes, application of misalignments on several levels in the geometry hierarchy. ALICE also makes use of the special TGeo geometry entities called Assemblies.

Acknowledgements

I am grateful to J. Palacios, A. Gheata, M. Gheata, F. Ronga, G. Gorfine, R. Grosso, M. Clemencic and M. Case for providing me with essential information and for the useful discussions during the preparation of the present overview.

References

[1] GEANT - Detector Description and Simulation Tool, *CERN Program Library Long Writeups Q123*.

[2] R. Brun, A. Gheata and M. Gheata, *Nucl. Instrum. Methods* **A502** (2003) 676–680.

[3] GEANT4 Collaboration, *Nucl. Instrum. Methods* **A506** (2003) 250–303.

[4] A. Fassò, A. Ferrari, J. Ranft and P.R. Sala, CERN-2005-10, INFN-TC-05-11, SLAC-R-773 (2005).

[5] C. Cheshkov, A. Gheata and R. Grosso, ALICE Internal Note, in preparation.

[6] `http://aliceinfo.cern.ch/Offline/Activities/ConditionDB.html`.

[7] `https://twiki.cern.ch/twiki/bin/view/Atlas/GeoModel`.

[8] `https://twiki.cern.ch/twiki/bin/view/Atlas/InDetAlignHowTo`.

[9] `http://proj-clhep.web.cern.ch/proj-clhep/`.

[10] `http://lcgapp.cern.ch/project/CondDB/`.

[11] `http://lcgapp.cern.ch/project/pool/`.

[12] `https://twiki.cern.ch/twiki/bin/view/LHCb/GeometryFramework`.

[13] `http://lhcb-comp.web.cern.ch/lhcb-comp/Frameworks/Gaudi/Gaudi_v9/GUG/Output/GUG_DetDescription.html`.