GAUDI Detector Data Model

Requirements & Analysis

R. Chytracek *EP/ALC, CERN, Geneva, Switzerland*

Abstract

This document describes the requirements and analysis for the design and development of the detector geometry description database of the GAUDI framework. The main domains we are interested in are the transient and persistent detector data models consisting of the detector description, detector geometry, conditions and other related topics. As a result the framework components of the GAUDI will be defined in terms of UML. To ensure that the design will be state-of-the-art an analysis of existing approaches to detector description databases and geometry models has been done and is included in the last chapters.

Keywords: detector description database (DDDB), detector geometry, alignment, calibration, ODBMS, transient detector data model, persistent detector data model.

Contents

1 What is detector data model (DDM)?

In this chapter the scope of the requirements & analysis will be defined in terms of the domains relevant to DDM. The DDM overview is shown on Figure 1.

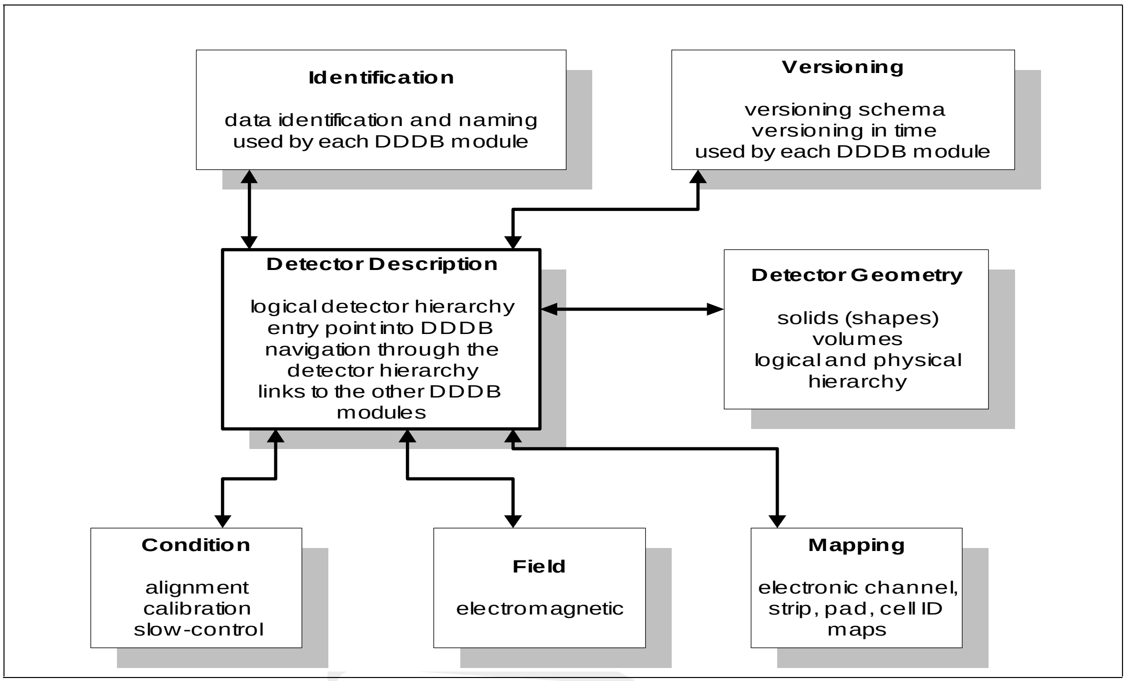

Figure 1: Detector data model schema

The DDM follows the GAUDI philosophy to provide separated transient and persistent stores to make it independent on the particular persistency technology. The schematic view of the stores is shown on Figure 2.

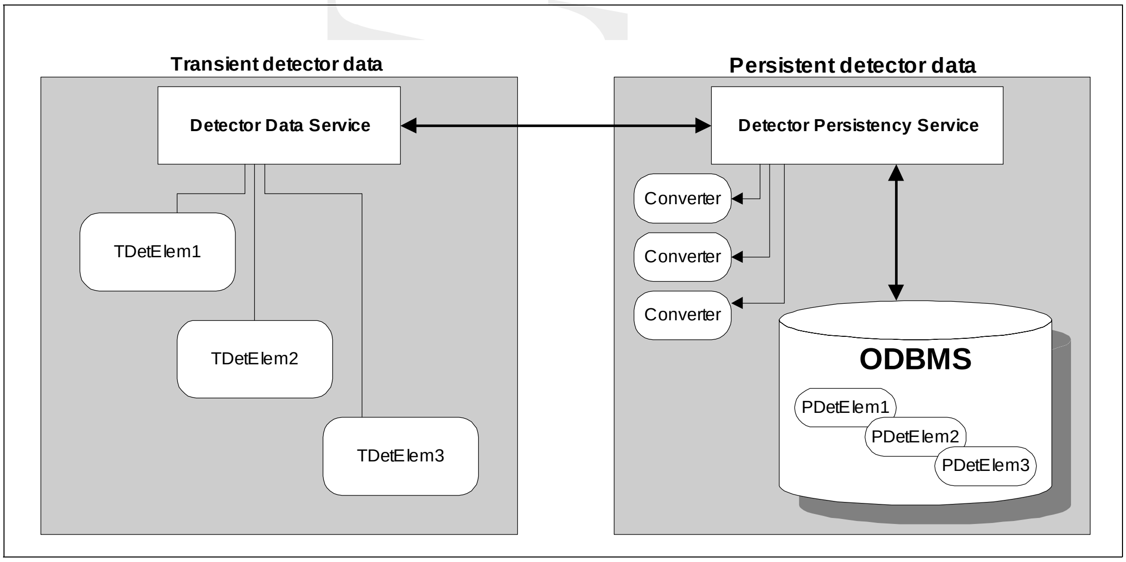

Figure 2: Transient and persistent data stores schema

1.1 Detector description

Detector description contains identifiable entities and should provide an information about detector logical hierarchy, its subsystems and the links to the relevant information. The information as geometry parameters, versions, conditions (alignment, calibration, slow control), mapping and field is managed by the other detector data model parts. The navigation should be provided here allowing to browse and query required data in detector data model.

1.1.1 Transient detector data model (TDDM) proposal

An idea is to reuse existing Gaudi transient event model philosophy to provide a way for storing detector description data in the memory and along with it to provide navigability through tree structure. How it might be structured in the memory is shown on the Figure 3. Each detector data directory contains always directory with conditions data tree and geometry data tree and variable number of subdetector directory.

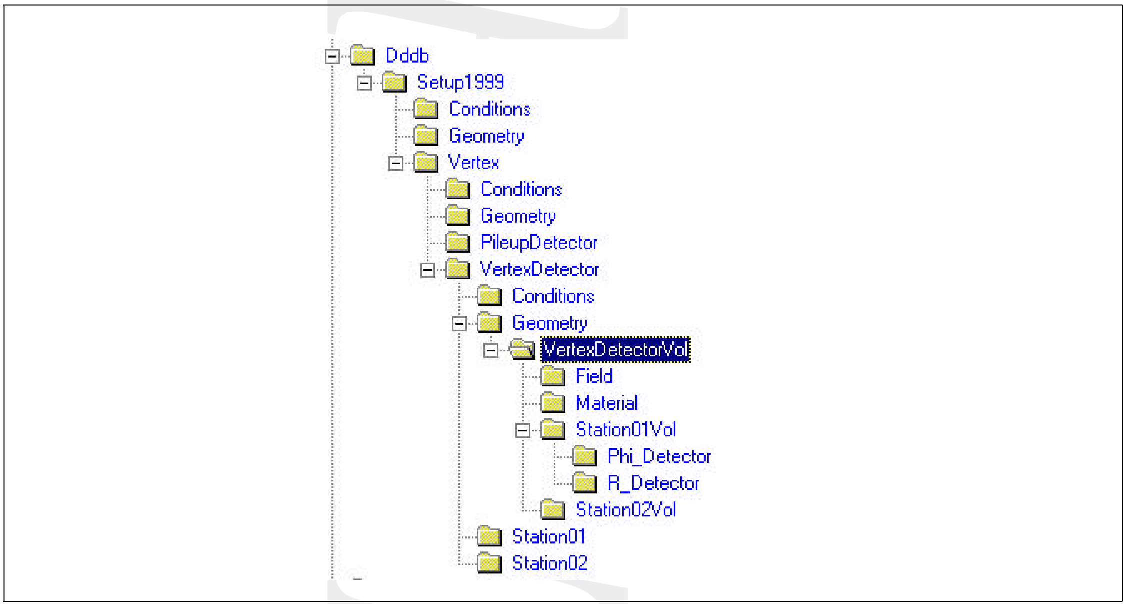

Figure 3: Detector description hierarchy in the transient detector data store

Due to huge amount of expected detector data and its rich variability is assumed to perform detector data selection over metadata[1]. This should allow data browsability and navigability in real-time without actual need to load real data from the persistent detector database into transient store. Of course metadata are expected to be stored in the persistent data store as well.

[1] A data about data, data relationships, data storage structures

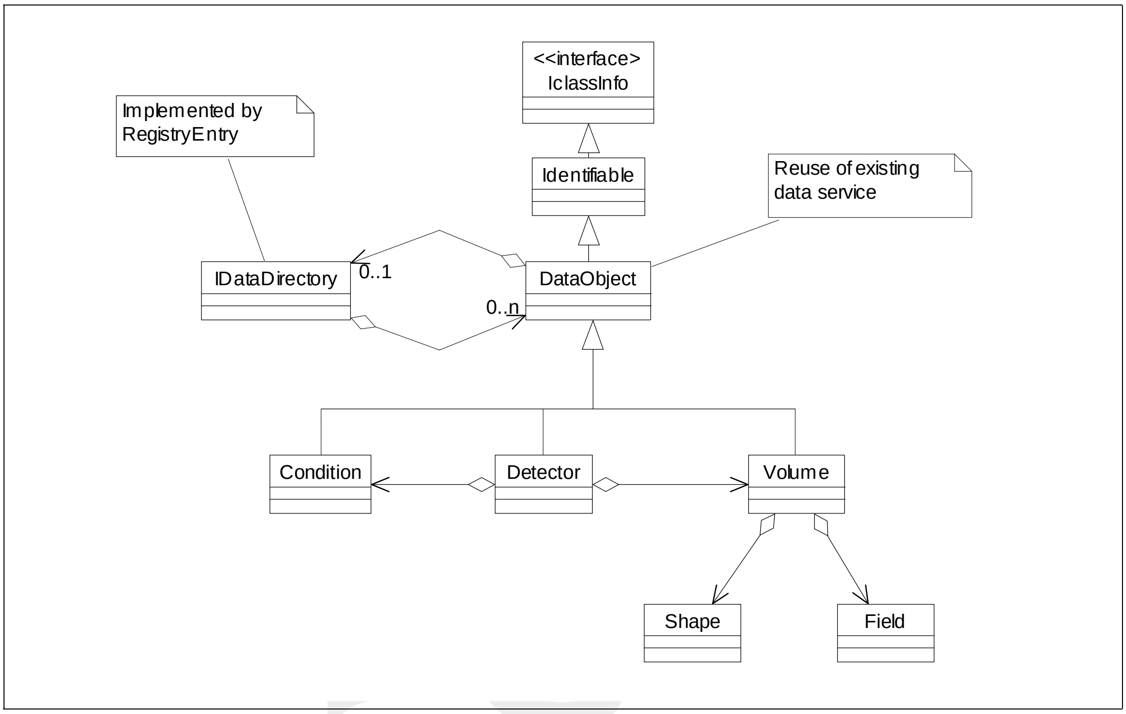

Figure 4: Transient detector data store model

1.1.2 Questions

1. What should be in detector description store?
    a. Projection of selected detector data from the persistent store.
    b. Hierarchical structure (requires identification, naming, versioning)
    c. Response, resolution, efficiency
    d. Hit information, digit information

1.2 Detector geometry

The geometry database should provide information about geometry hierarchy of the detector and its subsystems as the volume definition, volume's shape description, translation and rotation. information whether it is sensitive volume or passive one. Each volume must have its material and field description. The geometry hierarchy may not correspond one to one to detector description hierarchy as the some of the detector subsystems can consist of a few modules from the logical point of view, but may have different volume definition and position in the geometry hierarchy.

1.2.1 Questions

1. How to define the naming convention for the transient geometry model (volumes, assembly, setup, component, solid, shape etc)?
   a. see glossary for the initial proposal
2. What is the best way to keep coordinates information in the persistent and transient store?
3. What are the data needed for complete material definition?
   a. physics properties
   – interaction length, radiation length, density,A of material, Z of material,...
   b. tracking properties
   – ?magnetic field?, step size, energy cut-offs,...
4. How many geometries we need to store?
   – real, ideal,....?

1.3 Identification

Very important is to give a name to the detector and its parts and/or provide a mechanism to identify them. It's very important for querying for data and navigation in the DDDB. The possibility to identify a detector data object should work for multiple versions of the object, see the next section.

1.4 Versioning

It is foreseen to have more than one version of the various detector parts including the detector itself. This is the same for every domain of the detector description database. For example there may exist more than one version of set of the alignment and calibration constants. The versioning is supposed to take into account the differences in time. So one can have more than one version of the alignment constants valid during some time period.

There are two basic classes of objects from versioning system point of view. The first class are the objects which don't evolve in time and are bound to the time when they were created or registered. It means they are all the time the objects of the same class with the same set of their attributes but containing different data. In the other words their logical and physical structure remains the same through time flow but they contain a snapshot of information bound to the particular time or valid in the certain time range, e.g. temperature, high-voltage, pressure, thresholds, electronic channel maps[1]. These objects can be assigned a timestamp information.

The second class objects can evolve in time and usualy the objects reflect higher level of abstraction. When the time flows the objects can change their logical (e.g. geometry volume division process, removal or change of some part of a detector in detector description) or even the physical structure (upgrade or modification of class object's attributtes or methods according to a new requirement or need). This leads to a few possible scenarios for this class of objects.

1.4.1 Time indexing

The objects are indexed on time basis. It means for instance the calibration data are indexed on time of event. This does not assume that calibration data improve.

1.4.2 Version indexing

The objects are modified or reprocessed and so a new quality must be somehow reflected, so an increased version index is assigned to the object. In the given example above the calibration data were improved somehow and new version is stored and/or registered in a database. Another example can be having multiple versions of alignment or geometry data.

1.4.3 Event or process type indexing

There are objects which must not change during a running process or event, e.g. alignment, geometry, magnetic field direction. These can be assigned a run or event number as an index. The objects are part of run configuration and so cannot be versioned in this context.

1.4.4 Data input vs. versioning

The best way to version or index data depends on the way the data are introduced into a database or how the data are taken. Some of the data are measured on periodical time basis, others are taken when new run starts or ends. On the other hand there are data which are designed and constructed by people long time before run starts and already can consist of multiple versions with respect to quality, application use or time.

All of this leads to a conclusion that the data must be classified and assigned to groups which can overlap, see Figure 5.

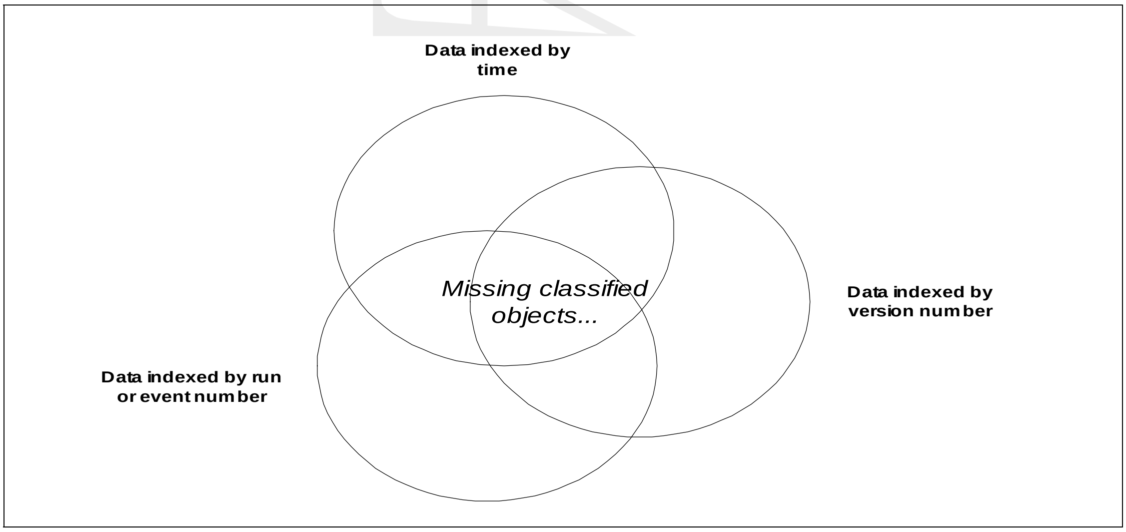

Figure 5: Versioning class sets and their relationship

1.4.5 Questions

1.5 Conditions

The conditions include the alignment data, calibration data, slow control data.

1.5.1 What are the conditions data?

- description of the conditions under which event data were taken
- necessary for the correct interpretation of event data
- necessary to configure the detector for the next physics run
- useful for detector debugging and monitoring
- irreplaceable measurements independent of physics events

1.5.2 Examples of conditions data

- detector environment: temperature, atmospheric pressure, ...
- beam and detector conditions: magnet currents, high voltage, gas temperature, gas purity, gas pressure
- front-end electronics calibrations: pedestals, gains, ...
- constants derived from event data: alignment, energy scale, ...

1.5.3 Questions

1. What is needed information to store for alignment (misalignment) database?
2. Selection criterias for versioning
   a. Data selected by time value
   – e.g. select data according to time of the given event
   – examples: pressure, temperature, HV
   b. Data selected by time range (validity period)
   –

1.6 Mapping

Very important information is mapping of channels, cells, pads and other parts of detector. For example mapping of readout channels to pads.

1.7 Field map

Information about magnetic field and/or the other types of fields.

1.8 Conversions

1. GEANT4
2. GEANT3?
3. ROOT
4. Graphics
5. WIRED, VRML, XML?
6. CAD
   a. use of STEP exchange data format (compatibility with CAD and GEANT4)
   b. use of EUGENIE (from/to EUCLID)

2 Software clients of DDDB

The very important thing to be done is to distinguish and identify the software applications (software clients) where DDDB will be used and the groups of users who will use it for their everyday work.

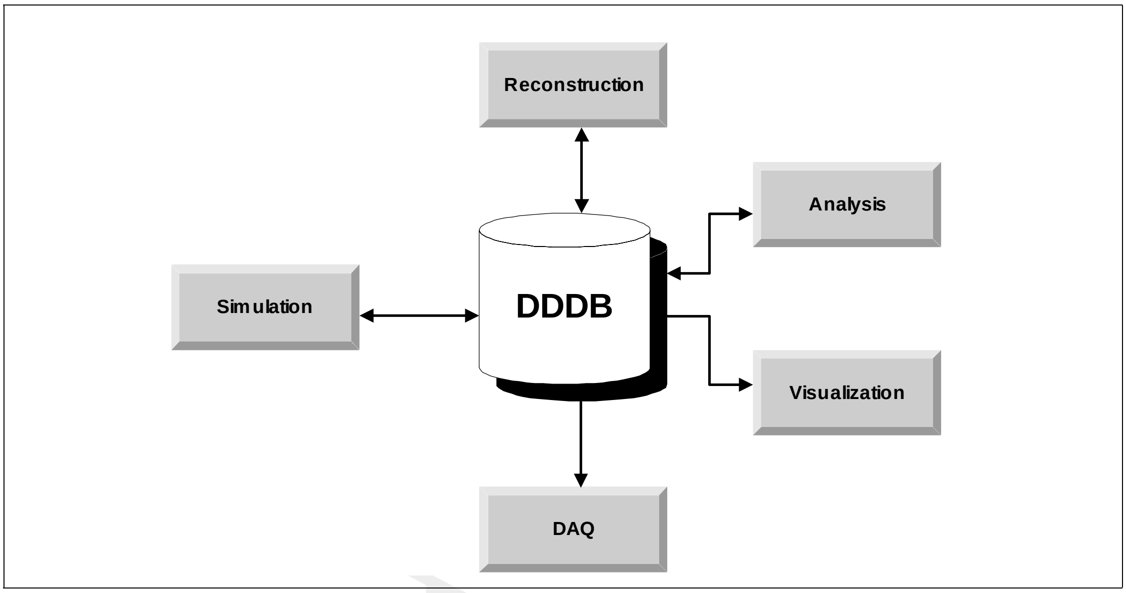

Figure 6: DDDB clients

Simulation

2.1 Muon detector group

2.1.1 Detector simulation (GEANT)

Input:

- All Event MC Kinematic data
- Muon detector station geometry data
- Muon detector station alignment data
- Muon detector chamber geometry and material data
- Muon detector shield geometry and material data

Ouput:

- Muon MC Hit data

Digitization

2.2 Muon detector group

2.2.1 Muon event raw hit data production

STEP A

Input:

- Muon event MC Hit data
- Muon detector pad/strip geometry data
- Muon detector pad/strip response data

Output:

- Muon event pad hit data

STEP B

Input:

- Muon event pad hit data
- Muon detector digitization parameter data

Output:

- Muon event raw hit data

Reconstruction

- Refinement and update of alignment and calibration data after reprocessing
- In some cases reconstruction may provide feedback for geometry, e.g. deformation of detector elements

2.3 Muon detector group

2.3.1 Hit reconstruction

Input:

- Muon event raw hit data
- Detector pad/strip geometry data
- Detector hit reconstruction data

Output:

- Muon event reconstructed hit data

2.3.2 Track segment reconstruction

Input:

- Muon event reconstructed hit data
- Muon detector station geometry data
- Muon detector station alignment data
- Muon detector track segment reconstruction parameter data

Output:

- Muon event reconstructed track segment data

2.3.3 Level 0 trigger

Input:

- Muon event raw hit data
- Muon detector trigger parameter data

Output:

- Muon event trigger track segment data

Analysis

2.4 Analysis

  – feedback to alignment

2.5 Data acquisition (DAQ)

2.6 Visualization (event display, geometry editor)

  – No feedback
  – Low level of details, just basic geometry information needed to draw detector elements including hierarchy relations and transformations

2.7 Users of DDDB[2]

1. People doing simulation
2. People doing reconstruction
3. People doing analysis
4. People from detector groups (on-line + off-line)
   – probably doing all the things above + their own stuff

2 Event display and visualization will be used probably by all the user groups

Table 1: Detector groups and contact persons

Detector group	**Contact person(s)**
Vertex	Chris Parkes
Tracker	Gonzalo Gracia
Rich	Guy Wilkinson, Roger Forty, Niko Neufeld
Calorimeter	Ivan Korolko
Muon	Andrei Tsaregorodtsev, Paul Colrain, Gloria Corti

3 Use cases

Use cases described for each of the selected user groups.

3.1 Muon detector group

3.1.1 Use Case M1: Given a MC Muon Hit in a given chamber, find the pad id(s) of the pad(s) traversed (Paul Colrain)

Input: A MC Muon Hit (chamber id, active area entry point (x,y) and exit point (x,y)), pad layout of the chamber, z position of pad layer within the chamber

Output: pad id(s) of the pad(s) traversed

3.1.2 Use Case M2: Given that a pad in a given chamber is traversed by a charged particle at a given time (time of flight), decide whether or not this pad or any neighbouring pads (cross-talk) are fired and if they are fired, determine to which bunch crossing they should be assigned (Paul Colrain)

Input: Pad id, time of flight, chamber response information (pulse height spectrums, arrival time spectrums, cross-talk, etc.), chamber conditions (calibration information - pulse height thresholds, etc)

Output: pads fired and their time stamps (bunch crossing assignment)

Additional information required: Was pad fired in previous event? This can introduce a dead time depending on the technology

3.1.3 Use Case M3: Given two raw hits, one in muon station 2 and one in station 3, find the predicted position (x,y) of the hit in station 1 (Paul Colrain)

Input: The two raw hits in stations 2 and 3 and the z positions of the pad layers in the chambers in station 1

Output: ID of the chamber hit and the position (x,y) of the hit on the pad layer of the chamber

Remarks:

- A raw hit will probably consist of a pad id and, in MC, a Time Stamp

3.1.4 Use Case M4: Given the predicted position (x,y) of a hit in any chamber of any station, find the nearest raw hit within the Field of Interest (Paul Colrain)

Input: Predicted position (x,y) within chamber, list of raw hits within the chamber

Output: Nearest raw hit

Additional information required: The (x,y) of a given raw hit (pad centre) given its pad id. This will be calculated using the pad id and the pad geometry data

3.1.5 Use Case M5: Given a raw hit in station 1 and a raw hit in station 2, calculate the transverse momentum, pt, of the candidate muon (Paul Colrain)

Input: The 2 raw hits, z position of the magnet bending plane and the average pxkick in the magnet

Output: pt of the candidate muon

Remarks:

- The (x,y,z) of a raw hit will be calculated from its pad id and the pad geometry data

3.2 Track reconstruction

3.2.1 Use case R1: Track reconstruction in active material (Gonzalo Gracia)

Input: wire (number), TDC or ADC counts (drift time)

Output: coordinate of the hit perpendicular to the wire/strip

Additional information required: calibration, alignment, z position (of the layer), eventually if the detector volume is tilted (angle)

Remarks:

- Inner tracker has 4 or 6 layers/station
- Outer tracker has 8 or 12 layers /station

3.2.2 Use case R2: Track reconstruction in passive material (Gonzalo Gracia)

Input: radiation length[3], z position (of the volume), entry point and exit point of the track in the volume

Output: extrapolation of the track in the given material with error calculation (track parameters are not changed)

Additional information required: length of the track inside the passive material, the implication is that to obtain this information the geometry database must provide information about the volume containing this passive material and its rotation and translation matrices in order to know exactly how the volume is positioned with respect to the track to allow for computation of the length.

3.2.3 Use case R3: Use of magnetic field (Gonzalo Gracia)

B_x, B_y, B_z coordinates at the given 3D (x,y,z) point are required, see Section 3.3.1. Since the magnetic field information is used very often that requires the magentic field map to present in the memory while the reconstruction job is running.

3.3 Analysis

3.3.1 Use case A1: Use of magnetic field (Gloria Corti)

Direct access to magnetic field information for the analysis jobs is needed. The information can be obtained either from the magnetic field map or calculated on the fly for the point in 3D space if the coordinates of the 3D point do not correspond to any point in the magnetic field map (either the closest point in the map or calculated values via interpolation). Another needed information is magnetic field polarity in the given event or run.

---

[3] of the material which the track goes through

3.3.2 Questions

3.4 Visualization

3.4.1 Questions

1. Where should be stored the information needed by event display and visualization? Should it be part of a volume or managed by some other parts of the Gaudi which take care of the visualization and event display?

4 Scenarios

Simulation

4.1 Muon detector group

4.1.1 Scenario SM1: (Paul Colrain)

1. Superimpose background events
   - Choose the background event generator and alter the normalisation.
2. Choose from different predefined
   - Muon Detector Station Geometry data
   - Muon Detector Station Alignment data
   - Muon Detector Chamber Geometry and Material Data
   - Muon Detector Shield Geometry and Material Data
3. Alter/overwrite the
   - Muon Detector Station Geometry data
   - Muon Detector Station Alignment data
   - Muon Detector Chamber Geometry and Material Data
   - Muon Detector Shield Geometry and Material Data
4. Within one job compare different
   - Muon Detector Station Geometry data
   - Muon Detector Station Alignment data
   - Muon Detector Chamber Geometry and Material Data
   - Muon Detector Shield Geometry and Material Data
   - Background Generators
5. Store, in Persistent Storage, a subset of the MC Event data produced, according to some selection criteria
6. Store, in Persistent Storage, ONLY the MUON Event MC Hit data and the associated MC Kinematic data
7. Store, in Persistant Storage, Muon Simulation histograms

Digitization

4.2 Muon detector group

4.2.1 Scenario DM1: (Paul Colrain)

1. Access, in Persistant Storage, a private set of Event MC data
2. Access, in Persistant Storage, a private set of Muon Event MC Hit data and the associated MC Kinematic data only
3. Perform Digitization for Muon Detector only
4. Choose from different predefined
   - Muon Detector Pad/Strip Geometry data
   - Muon Detector Pad/Strip Response data
   - Muon Detector Digitization Parameter data
5. Alter/overwrite the
   - Muon Detector Pad/Strip Geometry data
   - Muon Detector Pad/Strip Response data
   - Muon Detector Digitization Parameter data
6. Within one job compare different
   - Muon Detector Pad/Strip Geometry data
   - Muon Detector Pad/Strip Response data
   - Muon Detector Digitization Parameter data
7. Add Pad Noise hits at a chosen level
8. Store intermediate Muon Event Pad data in Persistant Data Store
9. Store, in Persistent Storage, a subset of the Raw Event data produced, according to some selection criteria
10. Store, in Persistent Storage, only the Muon Event Raw Hit data
11. Store, in Persistant Storage, Muon Digitization histograms

Reconstruction

4.3 Muon detector group

4.3.1 Scenario RM1: Hit reconstruction (Paul Colrain)

1. Access, in Persistant Storage, a private set of intermediate Muon Event Pad data
2. Access, in Persistant Storage, a private set of Event Raw data
3. Access, in Persistant Storage, a private set of Muon Event Raw Hit data only
4. Perform Hit Reconstruction for Muon Detector only
5. Choose from different predefined
   – Muon Detector Hit Reconstruction Parameter data
6. Alter/overwrite the
   – Muon Detector Hit Reconstruction Parameter data.
7. Within one job compare different
   – Muon Detector Hit Reconstruction Parameter data
8. Store, in Persistent Storage, a subset of the Reconstructed Hit Event data produced, according to some selection criteria.
9. Store, in Persistent Storage, only the Muon Event Reconstructed Hit data
10. Store, in Persistant Storage, Muon Hit Reconstruction histograms

4.3.2 Scenario RM2: Track segment reconstruction (Paul Colrain)

1. Access, in Persistant Storage, a private set of Event Reconstructed Hit data
2. Access, in Persistant Storage, a private set of Muon Event Reconstructed Hit data only
3. Perform Track Segment Reconstruction for Muon Detector only
4. Choose from different predefined
   – Track Segment Reconstruction code and
   – Muon Detector Track Segment Reconstruction Parameter data.
5. Alter/overwrite the
   – Track Segment Reconstruction code and
   – Muon Detector Track Segment Reconstruction Parameter data
6. Within one job compare different
   – Track Segment Reconstruction code and
   – Muon Detector Track Segment Reconstruction Parameter data
7. Store, in Persistent Storage, a subset of the Reconstructed Track Segment Event data produced, according to some selection criteria
8. Store, in Persistent Storage, only the Muon Event Reconstructed Track Segment data
9. Automatic change of Alignment data when run changes
10. Store, in Persistant Storage, Muon Track Segment Reconstruction histograms

4.3.3 Scenario RM3: Trigger (Paul Colrain)

1. Access, in Persistant Storage, a private set of Event Raw Hit data
2. Access, in Persistant Storage, a private set of Muon Event Raw Hit data only
3. Perform Trigger for Muon Detector only
4. Choose from different predefined
   - Trigger code and
   - Muon Detector Trigger Parameter data.
5. Alter/overwrite the
   - Trigger code and
   - Muon Detector Trigger Parameter data.
6. Within one job compare different
   - Trigger code and
   - Muon Detector Trigger Parameter data.
7. Store, in Persistent Storage, a subset of the Trigger Event data produced, according to some selection criteria
8. Store, in Persistent Storage, only the Muon Event Trigger data
9. Store, in Persistant Storage, Muon Trigger histograms

Visualization

5 Data model requirements

5.1 The common requirements

1. Support for multiple versions of detector components and setups
    a. What is the right depth of levels to provide versioning for, in sense of the components hierarchy (setup, detector, Vertex subdetector, layer, faces, modules, wafers, channels)?

5.2 Transient model

1. Optimized for run-time efficiency in terms of CPU usage and user friendly manipulation of data
2. Use of C++ pointers for navigation and information retrieval

5.3 Persistent model

1. Optimized for space efficiency[4]
2. No data duplicity
3. Use of logical links for navigation and information retrieval

[4] Speed of data loading from the persistent store is considered later on the TODO list

5.4 Data type requirements

5.4.1 Muon group

1. Muon detector station geometry data
   - nominal station positions and dimensions
2. Muon detector station alignment data
   - together with 1. will provide precise determination of station positions
3. Muon detector chamber geometry and material data
   - positions, dimensions and material of chambers within each station
4. Muon Detector Shield Geometry and Material Data
   - Positions, dimensions and material of the Muon shields
5. Muon Detector Pad/Strip Geometry Data
   - Pad/Strip geometry within each chamber
6. Muon Detector Pad/Strip Response data
   - Efficiency, timing, etc. properties of the pads/strips
7. Muon Detector Conditions data
   - Alignment, Calibration, Threshholds, etc.
8. Muon Detector Digitization Parameter data
9. Muon Detector Hit Reconstruction Parameter data
10. Muon Detector Track Segment Reconstruction Parameter data
11. Muon Detector Trigger Parameter data

6 Existing detector description databases

6.1 ALICE Geometry Database

6.1.1 Objectives

- definition of a set of data structures to hold the geometrical information about ALICE detector for
  - Simulation
  - Reconstruction
  - Visualization
  - ...everywhere the geometry of ALICE is needed
- having both disk and memory data structures
- handle multiple versions
- intarfeces to CAD, geometrical modeller and visualization
- provide a model for other detector databases

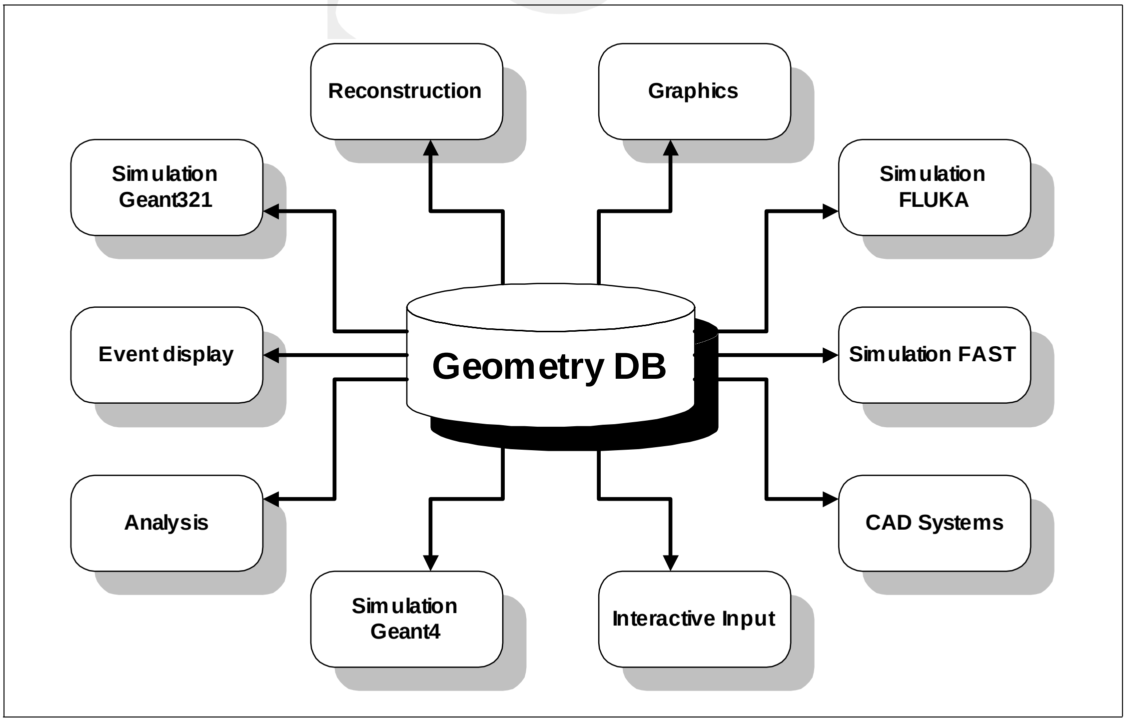

Figure 7: ALICE geometry database system overview

6.1.2 Memory structure (transient model)

Geometry hierarchy is represented as a tree of nodes[5]. The nodes are organized in an UNIX filesystem like structure. The node's items are lists of

- Transformations
- Materials
- Shapes
- Configurations

6.1.3 Disk structure (persistent model)[6]

This structure contains different versions of each node (**not clear to me how it is done**). Modular import/export of nodes or complete trees is allowed + editing (interactive)[7].

6.1.4 Current status

1. Writing/Reading geometry data into disk
   a. for each node is saved: Shape, Material, Configuration file (**what is it?**) and Transformations (applied to its sons).
   b. for each son node there is one subdirectory in the parent node
2. Translation from GEANT3 (performed by ROOT C++ macro)
3. Visualization (provided by ROOT)

6.1.5 Plans

After completing the conversion from GEANT3 to ALICE Geometry DB the next step will be to provide conversion for writing back to GEANT3. The same excercise will be done with GEANT4.

6.1.6 Comments

- They focus to detector geometry only
- They stick to one type of transient and persistent store (ROOT)
- What is done for identification and versioning is not clear from the presentation
- The need of convertors is the same as in GAUDI
- It is not clear how they handle calibration & alignment data, field information and their multiple versions.
- Considering versioning it seems they can have multiple versions for each node.
- How does correspond the ALICE GDB node to the term of logical/physical volume in GEANT4?

---

[5] Essentially ROOT TDirectory
[6] ROOT data files
[7] Enabled by ROOT visualization

7 Existing geometry models in use

There are a few geometry models used by HEP community:

- **CSG -** Constructed Solid Geometry (GEANT3,GEANT4,ROOT)
- B-REP **-** Boundary REPresentation (GEANT4,CAD)
- CG **-** Combinatorial Geometry (FLUKA)

This chapter focuses on the use of these geometry models by the particular software packages and not on the explanation of the models themselves.

7.1 FLUKA

FLUKA system uses modification of the package developed at ORNL called MORSE. The two fundamental concepts in CG are **bodies** and **regions**.

Bodies are defined as convex solid bodies (finite portions of space completely delimited by surfaces of first or second degree, i.e. planes and quadrics). FLUKA extends the definition to include infinite cylinders (circular and elliptical) and planes (half-spaces).

Regions are defined as combinations of bodies obtained by boolean operations: **union**, **subtraction** and **intersection**. Repetition of sets of regions is possible and allows to model in detail only a single cell of a calorimeter and to replicate it in the entire volume.

FLUKA predefines 12 basic body types (i.e. box, sphere, circular cylinder, elliptical cylinder, infinite half-space and others). The region volumes then can be defined by applying the boolean operations to bodies.

1. Transient FLUKA model is not probably what is desirable for designing flexible transient geometry data model. FLUKA implementation heavily relies on the FORTRAN code and the drawback of that is that is certainly not object oriented and is missing required level of data abstraction and modularity for easy to use geometry data manipulation. There are some thoughts recently about adding FLUKA functionality into GEANT4 and go on with GEANT4 geometry data model.
2. Another problem in FLUKA is the use of **absolute coordinates** in the whole system due to which the user must recalculate the coordinates for all geometry objects in case of a translation or rotation of some objects in the system. There is a workaround for that by using a technique which tries to avoid the recalculation of coordinates by preallocating some volume space in the system to ensure independence of the other objects in the system on changes inside the preallocated volume.

7.2 GEANT3

GEANT3 uses CSG model for its geometry data representation. Logical representation of geometry data is a tree of **volumes**. The root of the tree is so called **mother volume**. The reference frame of the mother volume is the master reference frame of the whole system.

The volume is defined by the given:

- name
- shape identifier
- shape parameters
- a local reference system
- the physical properties, given set of constants describing the homogeneous **material** which fills the volume
- additional properties (tracking medium, environment constants/magnetic field,....)
- a set of attributes for the drawing and detector response packages

The volume has no relationship to other volumes until is placed into the other volume or becomes a mother volume itself. Placing is done by translation and rotation of the volume inside its **parent volume** relative to reference frame of the parent volume and assigning material definition and rotation matrix to it. Any volume can contain its **daughter volumes** which can contain the other volumes etc. This way can be built the whole tree hierarchy describing some part of detector.

The user should make sure that no volume extends beyond the boundaries of its mother. When a volume is positioned the user gives it a **number**. Multiple copies of the same volume can be positioned inside the same or different mothers as long as their copy numbers differ. The contents of the positioned volume are reproduced in all copies.

In addition to that the volumes can be created by **divisions** and/or **boolean operations**. The general rule is that the result of the division be still a GEANT shape. A volume can be partially or totally divided. The division process generates a number of **cells**, which are considered as new volumes.

7.3 GEANT4

A detector geometry in GEANT4[2] is made of a number of **volumes**. The largest volume is called the **World volume**. It must contain all other volumes in the detector geometry. The other volumes are created and placed inside previous volumes, including the World volume.

Each volume is created by describing its shape and its physical characteristics, and then placing it inside a containing volume. When a volume is placed within another volume, we call the former volume the **daughter volume** and the latter the **mother volume**. The coordinate system used to specify where the daughter volume is placed is the coordinate system of the mother volume.

To describe a **volume's shape** we use the concept of a **solid**. A solid is a geometrical object that has a shape and specific values for each of that shape's dimensions. For example a cube with a side of 10 centimeters and a cylinder of radius 30 cm and length 75 cm are solids.

To describe a **volume's full properties** we use a **logical volume**. It includes the geometrical properties of the solid and adds physical characteristics: the material of the volume, whether it contains any sensitive detector elements, the magnetic field, etc.

What remains to describe is **to position the volume**. To do this you create a **physical volume**, which places a copy of the logical volume inside a larger, containing, volume.

Geometry, Magnetic Field and CAD-Interface

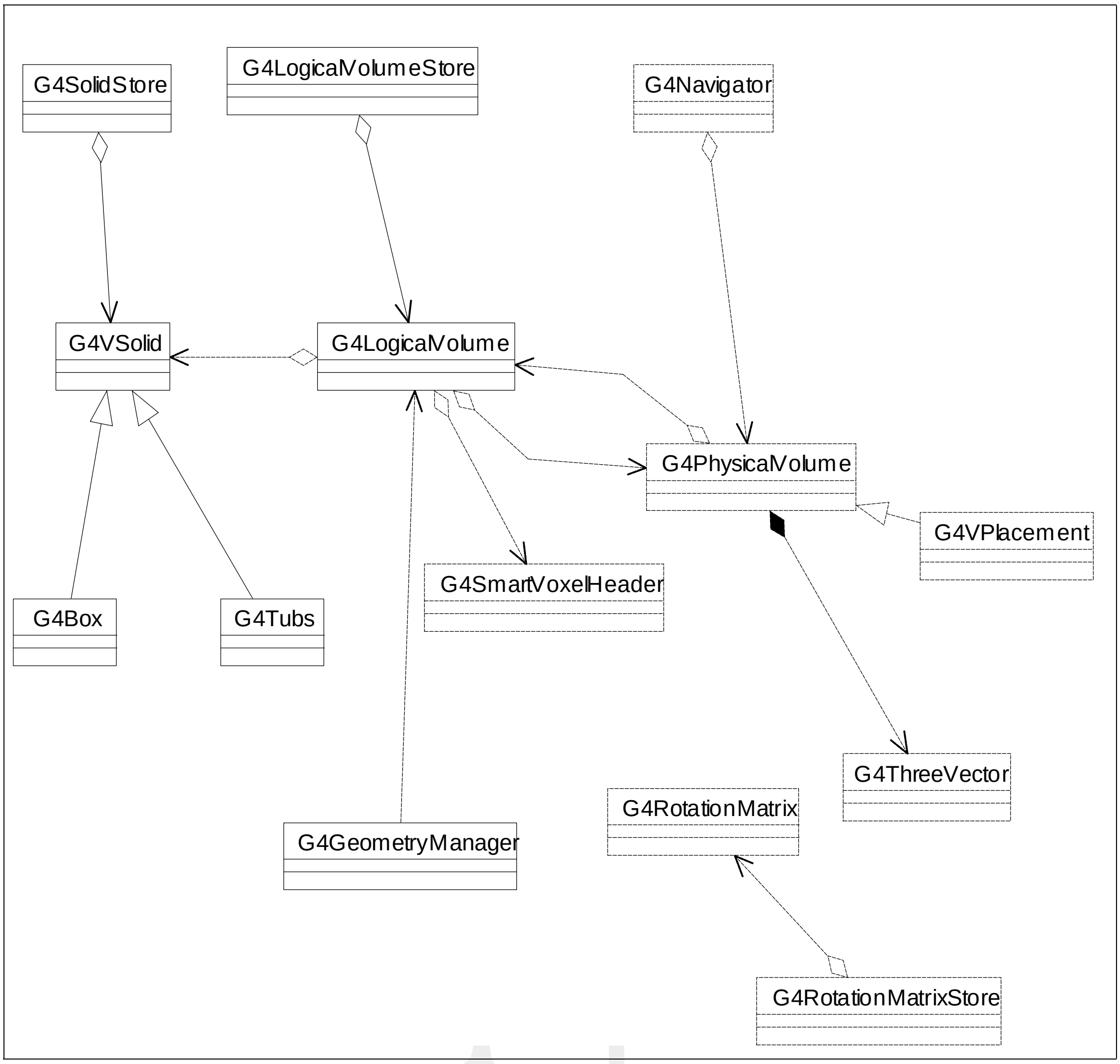

Figure 8: GEANT4 geometry data model

These three categories manage the geometrical definition of a detector (solid modeling and interactions with CAD systems) and the computation of distances to solids (also in a magnetic field). The GEANT4 geometry solid modeler is based on the **ISO STEP** standard [7] and it is fully compliant with it, in order to be able to exchange geometrical information with CAD systems. **A key feature of the GEANT4 geometry is that the volumes definitions are independent from the solid representation. By this abstract interface for the G4 solids, the tracking component works identically for various representations.** The treatment of the propagation in the presence of fields has been provided within specified accuracy. An OO design allows us to exchange different numerical algorithms and/or different fields (not only B-field), without affecting any other component of the toolkit.

For more information about GEANT4 geometry please see the URL:

http://wwwinfo.cern.ch/asd/geant/geant4_public/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/PracticalApplications/geometry.html

Questions to be answered:

1. What is the role of all the stores (logical volume store, solid store,...) in GEANT4 model?

7.4 ROOT

ROOT contains an embryon of geometry package that today provides only some basic graphics functions only. GEANT3 geometries can be translated to ROOT TNodes via the g2root utility. The g2root utility can also be used to automatically generate the C++ classes corresponding to the GEANT3 sets and dets.

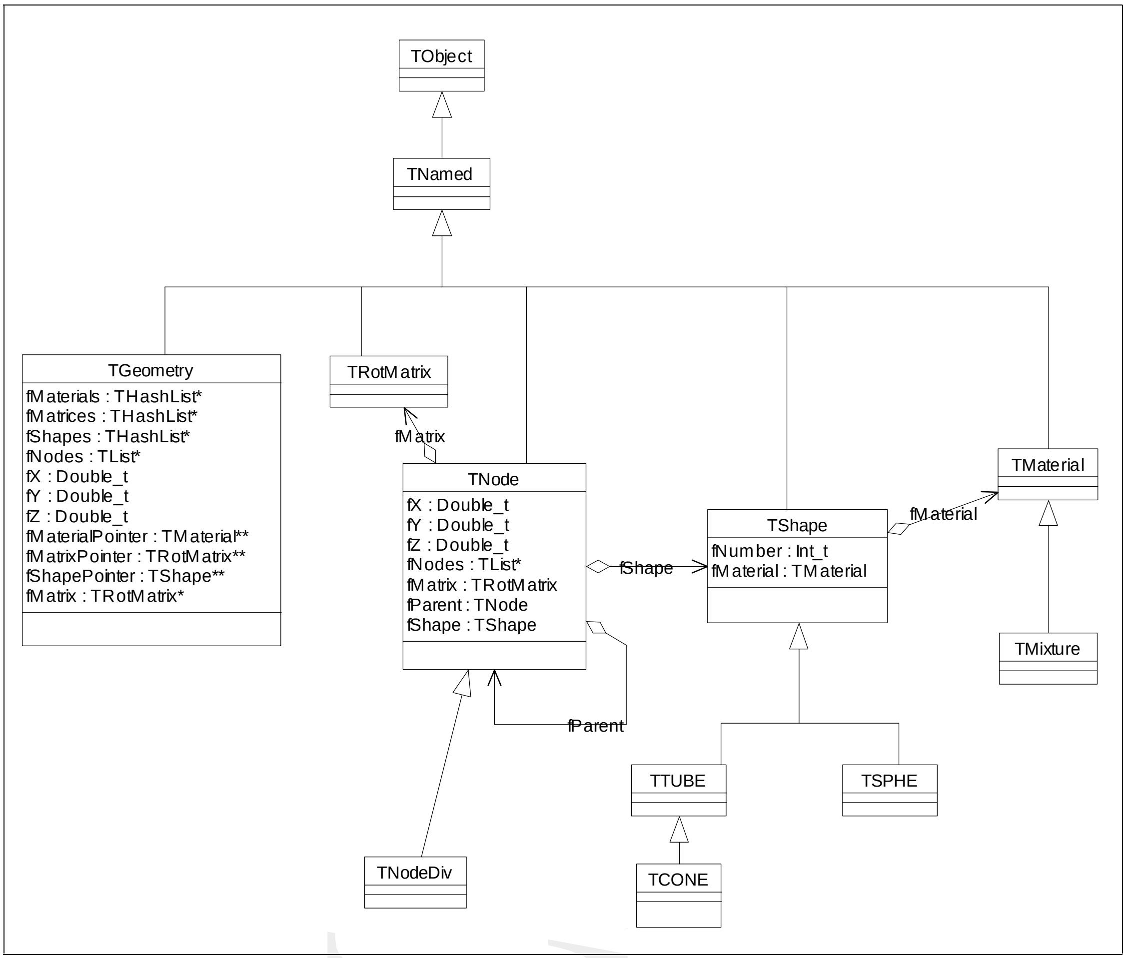

Figure 9: ROOT geometry data model.

The ALICE collaboration has implemented a complete interface to GEANT3 via the class TGeant3. An upgrade of this class has recently been provided by Pasha Murat from the CDF collaboration. The TGeant3 class provides a complete interface to the GEANT3 functions.

A set of classes AliXXXX have been developed to replace the hits functionality from GEANT3 and also in view of GEANT4. More details about the GALICE framework can be found in [3].

Using the TGeant3 and Ali classes, a complete C++ interface has been developed. Kinematics, Hits&Digits are stored in the form of ROOT Trees and are used directly by the ALICE reconstruction framework based on ROOT and entirely written in C++. Work is currently in progress to interface the Ali classes to GEANT4. This scheme provides an easier transition from GEANT3 to GEANT4 or GEANT3 to Fluka.

ALICE is also developing a geometry data base, see section 6.1, (independent from simulation programs) based on ROOT classes and a new set of classes (ALICE independent) to handle complex detectors. This data base tool includes geometry editors and viewers and automatic interface to GEANT3, GEANT4, Fluka, reconstruction/analysis programs and DAQ. This geometry data base tools is expected to be operational early next spring. For more details, contact Federico.Carminati@cern.ch.

The ROOT interface to GEANT4 still requires more work. The GEANT4 team is planning to replace the current RogueWave container classes by STL. CINT has some problems to digest RogueWave classes. The transition to STL should help. But the plan is to automatically generate the ROOT dictionaries for GEANT4 classes such that an interactive interface will be available and support for persistency provided via ROOT Trees.

Glossary

active material. wires, strips, e.g. in tracker or vertex detectors

passive material. blocks of material which does not produce information about hits, e.g. rich detector

component. A component can be a layer, module, wafer, strip channel,...

detector. A detector unit.The unit can contain subdetectors and/or its various components. Subdetector is considered as a detector as well.

setup. Setup is a configuration of a whole detector valid during some period of time. Setup includes all subdetectors and their components. There may be multiple versions of the setup.

version. Each setup, detector or detector component can have its multiple versions.

version tag. An human readable tag by which are logically labeled all the various versions of the parts of a detector or a setup belonging to the given logical version (release, run, time period,...).

volume. A general term for the object in the detector geometry

logical volume. An entity representing an element in the detector geometry with assigned shape definition, material properties

physical volume.

material.

field.

configuration. may be useful term to be added into versioning definition (setups, configurations, ...)

References

1 Y. Kolomensky, Configuration Database for BaBar Online System, CHEP'98, Chicago, IL

2 GEANT4 User's Guide
URL http://wwwinfo.cern.ch/asd/geant/geant4_public/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/index.html

3 Galice Framework
URL http://www.cern.ch/ALICE/Projects/offline/Simulation/galice/