

# The Role of XML in the CMS Detector Description

M. Liendl<sup>1</sup>, F. van Lingen<sup>1,2</sup>, M. Case<sup>3</sup>, T. Todorov<sup>1,4</sup>, P. Arce<sup>1</sup>,  
A. Furtjes<sup>1</sup>, V. Innocente<sup>1</sup>, A. de Roeck<sup>1</sup>

**for the CMS Collaboration**

<sup>1</sup> European Laboratory for Particle Physics (CERN), CH; <sup>2</sup> Eindhoven Univ. of Technology, NL and Univ. of West of England, GB; <sup>3</sup> Univ. of California Davis, US; <sup>4</sup> IReS Strasbourg, F

## Abstract

Offline Software such as Simulation, Reconstruction, Analysis, and Visualisation are all in need of a detector description. These applications have several common but also many specific requirements for the detector description in order to build up their internal representations. To achieve this in a consistent and coherent manner a common source of information, the detector description database, will be consulted by each of the applications. The role and suitability of XML in the design of the detector description database in the scope of the CMS detector at the LHC is discussed. Different aspects such as data modelling capabilities of XML, tool support, integration to C++ representations of data models are treated and recent results of prototype implementations are presented.

Keywords: XML, data modelling, detector description

## 1 Introduction

CMS offline software (Reconstruction ORCA, Full Simulation OSCAR, Fast Simulation FAMOS, Visualisation IGUANA [1]) are all in need of a detector description in order to build up their internal representation of the detector for efficiently fulfilling their specific tasks. For example, OSCAR needs a highly granular GEANT4 description of all sensitive and non sensitive parts of the detector whereas track reconstruction in ORCA would only need positions and shapes of sensitive tracker elements accompanied with their average material-budget and some additional (tracking-detector) specific information. To achieve coherent internal representations all applications should fetch detector information from a common source, the detector description database (DDD).

The detector description itself consists of two main parts: the ideal detector description and the conditions database (time dependent alignment corrections and calibration data of the various subdetectors are provided herein). The ideal description is defined using the below described XML techniques and formats whereas the conditions database format can be different for different subdetector domains. Further XML techniques are applied when transforming or merging existing detector description information into the new format. Various (mostly) standardised XML tools can be extended to fulfil these tasks.

So the role of XML in the CMS detector description is twofold: first serving as a description language for the ideal detector and second being a useful tool for data migration.

## 2 Basic Architecture of the Detector Description in CMS

Fig. 1 shows the basic architectural design of the CMS DDD. Applications are using the Common Interface (CI) which interfaces the DDD Core part. DDD Core consists of a transient object model structured according to the ideal detector description but extended with information from the conditions database. The design of the internal model reflects the requirements of the applications. Information from the conditions database may not be copied into the transient model but retrieved on demand.

On the data side there is one single defined entry point (interface) for the ideal detector description data realised as an XML Schema [3], the DDL (Detector Description Language).

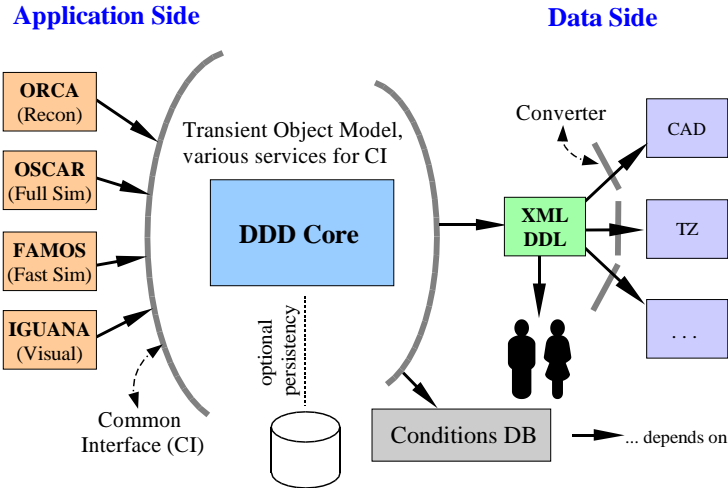
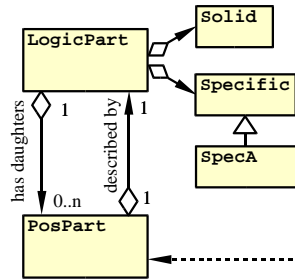


Figure 1: Basic Architecture of the CMS Detector Description Database

Ideal detector description covers the nominal detector specification (basically volumes, materials) and additional information attached to certain subdetector elements for specific demands of some of the applications. Documents using DDL are written by humans when a manual specification of subdetectors or of simple testbeam setups is the preferred way. If existing (semi-)structured data suggest an automatic transformation into the new format, converter programs can be used to write DDL conforming documents. In this case a transformation might not be complete in the sense that certain information is still missing being required by DDD Core to build up a consistent internal model. Then the DDL allows an appropriate extension of the transformed data.

**Description/Component Model**

- describes components
- ideal (=nominal) data (geometry, material, ...)
- ideal (=nominal) specific data
- DDL (XML) is used to describe the instances



**Application View - Model**

- access to fully expanded detector tree
- access to conditions data

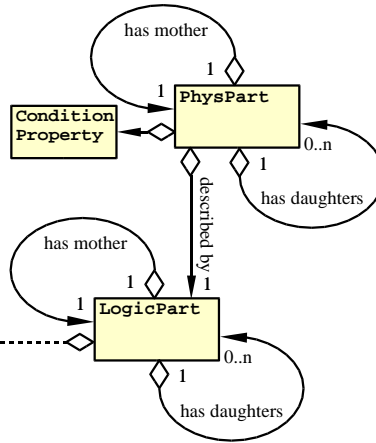


Figure 2: Outline of the two basic data models of the CMS DDD. Left: Description/Component Model as used when describing detector elements. Right: Application View - Model, which can be derived (immediately or on demand) from the Description/Component Model.

### 3 Basic Data Models

Two main data models are involved in the detector description. The outline of these two models is shown in the (simplified) analysis UML class diagram in fig. 2. The Description/Component Model serves as base for the definition of the DDL. It represents the detector as directed acyclic graph with LogicParts and PosParts being the vertices - closely following the ideas of GEANT4<sup>1</sup> [2]; LogicPart maps to G4LogicalVolume and PosPart to G4PhysicalVolume. Nominal (i.e. ideal) detector information such as solid specification, material definition, subdetector specific information, ... is attached to the LogicPart. Each LogicPart consists further of several (optional) components called PosParts which know their relative spatial position to their LogicPart and are themselves described by LogicParts.

From the Description Model the Application View-Model can be derived, which basically consists of the fully expanded detector tree with nodes represented as PhysParts. Naturally, because of efficiency and memory constraints, nodes in this tree will be provided on demand from the DDD Core. Further a derivation of the logical hierarchy is possible as well.

### 4 Properties of the Detector Description Language

The XML based DDL is itself defined in XML Schema [3]. The choice of XML Schema in favour of DTD was made, because more powerful and more typesafe XML constructs can be defined in XML Schema. XML parsers validating against a given XML Schema will set tighter restrictions on the XML documents than in in DTD case. Thus more 'correct' XML documents will automatically be enforced.

It is important that the the XML Schema for the detector description reflects the C++ transient Description/Component Model. This allows for an easy integration of data into the C++ environment. XML Schema is very well mappable to an object oriented world. It contains inheritance and abstract types. Within its hierarchical structure it provides a high degree of type safety. However, object oriented models contain typed pointers or references. This is only partly supported within XML Schema using the non-typed ID and IDREF(S) constructs. Furthermore, IDREF and ID are related to one single XML document. Within the CMS DDL special XML elements are introduced to describe type safe references across multiple XML files. The idea is based on the structure used in the Protégé project [4]. In principle an alternative approach using XLink, XPath and XPointer could be used therefore. However, currently appropriate tool support for these XML constructs is barely available, as the specification has just finished or will be finished in near future. Extending the XML schema with type safe references makes it easy to map to an object oriented world. Yet not every detailed object oriented aspect can be expressed in an XML Schema based data model. Therefore the transient model needs a module that contains a list of constraints that will be checked over the model. One such a constraint is that the LogicPart and PosPart in the detector description form an acyclic directed graph. Within the Protégé project [4] a first order logic language is used to formulate constraints on arbitrary models. Within this context we will focus only on the detector description domain, but will construct a structure for constraints that can be extended in the future if needed.

Apart from an ideal detector description, users/domains want to add specific data, to certain logical/physical volumes. For example: lightyield parameters to crystals, but not to muon chamber. Thus the model should be extendible with these domain specific characteristics. To cater for this an abstract node DomainSpecificParameters is introduced that links data files to specific logical volumes. Note that using XPointer and XLink this could be achieved as well.

---

<sup>1</sup>In fact the composition of LogicParts and PosParts is more restrictive in this approach than in GEANT4 as it does not allow to put a PosPart into another PosPart and thus preserves a strict component orientation. In GEANT4 a G4PhysicalVolume may be positioned into another G4PhysicalVolume.

## 5 XML as Data Migration Tool

There is a need to migrate the existing detector description information (e.g. TZ-file based data [5]) to the new Schema as required by the CMS DDD. A simple way to start this is to wrap existing data sources in XML. This first order XML is a one-to-one mapping of the original data and structured accordingly.

Once in the XML world, XSLT [3] is used to transform from this first order XML to the central XML Schema (DDL). XSLT allows function extensions when necessary to implement more complex transformations. Some information in the original files is somewhat hidden. For example in existing TZ-files, negative signs where non-negative numbers are normally expected may indicate further actions or relationships. Therefore it may be necessary to map such special flagged structures depending on their flag values to different XML types.

Finally, with the data in the common, central XML format, we can allow browsing and querying of the Detector Description using XPath and XQuery. Utilising these techniques on top of off-the-shelf XML parsers enables the DDD Core to build up the transient model more easily. In addition a well structured common format will allow further transformations from DDL based documents for other as yet un-foreseen uses. It will also allow users to learn about the detector description by easily searching for materials, components and relationships.

## 6 Integration into the CMS Computing Model

Current XML Schema definitions, XSLT transformations and related prototype work is located in the CMS repository. Configuration management is done using SCRAM [6]. First iterations of the DDD will be independent of the CMS software framework COBRA [1]. Future integration with this framework will be useful for exploiting COBRAs persistency and action-on-demand capabilities (as making the transient DD models persistent or interacting with the conditions database on demand).

## 7 Outlook

The current detector description model is based on XML Schema which provides a straightforward mapping to an object oriented environment. Furthermore, the architecture reflects a data warehouse approach in which different sources supply data to the detector description domain. The use of XML enables the usage of numerous off-the-shelf tools for data manipulation. The novel approach is to use XML files extended with certain management features instead of a database, for this data warehouse approach.

Migrating the full ideal detector description to our new Schema and providing a first iteration Common Interface to CMS applications is currently ongoing work. Successful migration of Muon geometries are already a first promising success.

## References

- [1] for further reference refer to following articles in these proceedings: 3-040 (IGUANA), 3-041 (ORCA), 5-015 (OSCAR, FAMOS), 5-014 (COBRA)
- [2] GEANT4 simulation toolkit, <http://wwwinfo.cern.ch/asd/geant4/>
- [3] XML, XML Schema, XPointer, XPath, XLink specifications: <http://www.w3c.org>.
- [4] The Protégé Project: <http://www.smi.stanford.edu/projects/protege>
- [5] GEANT Constants Definition Language for CMS, Yu. Fisyak, CMS Technical Note 95-080
- [6] Code Organisation and Configuration Management, S. Ashby et. al., these proceedings 8-004