

# XML for Modeling, Data Integration, and Meta data within CMS

Frank van Lingen<sup>1,4</sup>, Michael Case<sup>2</sup>, Martin Liendl<sup>3,4</sup>, and Ian Willers<sup>4</sup>

**Abstract-- Within certain domains of the CMS experiment at CERN, XML and related standards are used for construction of complex models, data exchange, transformation and data integration. Furthermore, meta data can be used to manage the transformations, integration models, and the related data sources. This paper gives an overview of the work that has been done on these subjects within a specific domain.**

**Index Terms-- XML, Data Integration, Meta data, Modeling**

## I. INTRODUCTION

At CERN, the fundamental structure of matter is studied using particle accelerators. The study of matter with accelerators is part of the field of high-energy physics (HEP). LHC [24] is a project for the construction of an accelerator that will be 1000 times more powerful than its predecessor. The CMS experiment [23] is building a detector in this accelerator that will be used for the study of particles after they have collided. Currently the CMS detector is being built. Such a detector is a machine that consists of millions of parts.

Simulation, visualisation, and other applications use a description of the CMS detector in order to get a better understanding of the detector when experiments start in 2006.

Because of the complexity of the detector this is a simplified description. However, data of this description currently resides in several sources (databases and flat files). Furthermore, different applications need to combine this simplified description data with more complex description data such as calibration sources, or other physics data sources.

The simplified description data does not change frequently but parts of this data will be used to build several "personal" detector descriptions to study various characteristics of the detector. The format for the detector description should therefore be easy to understand for the physicists working with it. Due to the size of the data nominal equal parts are

described by the same data within the simplified description. Other sources contain data that change frequently. These data are valid at a certain time or time interval.

Parts of this detector description will be transformed to other formats such as HTML, or RDF. It can then be used by applications that support these formats.

XML [18] has been chosen for several reasons:

- Most of the sources export XML.
- Several sources within this domain consist of flat files.
- XML applications based on standards such as Xpath [19] and Xquery [20] can be used for transformation and merging data.
- Tools are available that allow users to browse and edit XML data in a user friendly way.
- XML schema [16] allows for a flexible and type safe description of the data.
- XML parsers are available that validate the data, and can be used to import data in client applications.
- Data needs to be transformed to other XML formats.
- One of the aims is the use of standards that are supported by open source applications, such that the focus can be on modelling and management of data.

This paper is structured as follows: The next section discusses the architecture. Section III gives a global overview of the schema. Section IV discusses the mediator that merges data from the detector description with other data. Section V discusses the use of meta data for the management of transformations. Section VI discusses related work. Section VII summarises current work and highlights future research directions.

## II. ARCHITECTURE

Figure 1 shows the basic architecture. Sources will be wrapped in XML. Transformers based on XSL [17], Xpath, and Xquery transform the wrapped data to the XML detector description format. Users will be able to edit this data, either by using ASCII editors, but also dedicated XML editors. The mediator will provide a C++ interface to the XML data towards the clients, and merge this data with other sources. The internal model of the mediator represents a structure found in many data sources within CMS: Bill of Materials (BOM). The mediator will merge data from different sources

<sup>1</sup> Supported by Eindhoven University of Technology

<sup>2</sup> UC Davis, Department of Physics, One Shields Avenue Davis, CA 95616, United States, case@physics.ucdavis.edu

<sup>3</sup> Supported by Vienna University of Technology

<sup>4</sup> European Organization for Nuclear Research, CH-1211 Geneva 23, Switzerland, {fvlingen, martin.liendl, ian.willers}@cern.ch

containing such a BOM structure. The different client applications of the mediator all use the BOM structure.

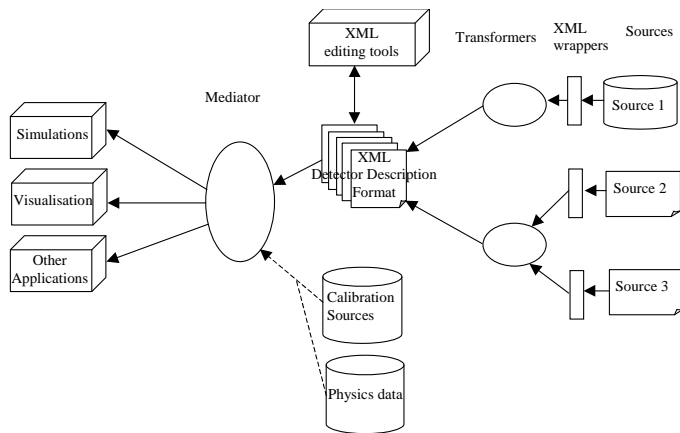


Figure 1. Basic architecture

Different clients need to merge data from the detector description with different sources.

Within the current architecture, XML is used as the storage format (ASCII files with tags). The XML files are instances of the XML schema defined for the detector description. The mediator transforms the data described by the XML files into a C++ model used by the clients.

### III. XML SCHEMA

The CMS detector description (CMSDD) consists of a hierarchy of geometry descriptions. Furthermore, this hierarchy contains parameters that give a detailed description about elements within this description. This hierarchy can be compared with a bill of materials (BOM). However, a BOM is based on a tree structure. The detector description is based on direct acyclic multi graphs (Figure 2. BOM). Acyclic multi graphs can be unfolded into a tree. Within this paper a BOM refers to a direct acyclic multi graph structure

To prevent an explosion of data, the schema consists of several layers of descriptions and instances based on these descriptions. The first layer contains descriptions of geometries and materials. Instances of these descriptions are in turn used as components of the description of parts (part types). The parts in the BOM are instances based on the part type descriptions. For example: part of the detector consists of 80,000 crystals. These are based on 5 types of crystals.

Figure 2 shows an example of a BOM. Within the XML description, part types have relative positions to other part types. A part type contains a geometry description and other information relevant for this part type. A requirement from the physics community was that data should not be duplicated if it is used within more than one XML node (one definition rule). Therefore the BOM provides one description for nominal equal parts. The references (arrows) between the XML descriptions are conceptual references. A requirement from the physics community was that the schema description

should be simple and intuitive for physicists who are not familiar with XML or the XML schema of the detector description. Furthermore, the types of references between data are known in advance. As a result of this requirement the XML schema has a set of predefined references that are interpreted by the mediator, instead of a generic Xlinks/Xpointer [15] description of the references. The data for the detector description is too large to fit into one XML file. Therefore references are needed between data of XML files. Based on this description the mediator constructs a direct acyclic multigraph, and expanded view on demand.

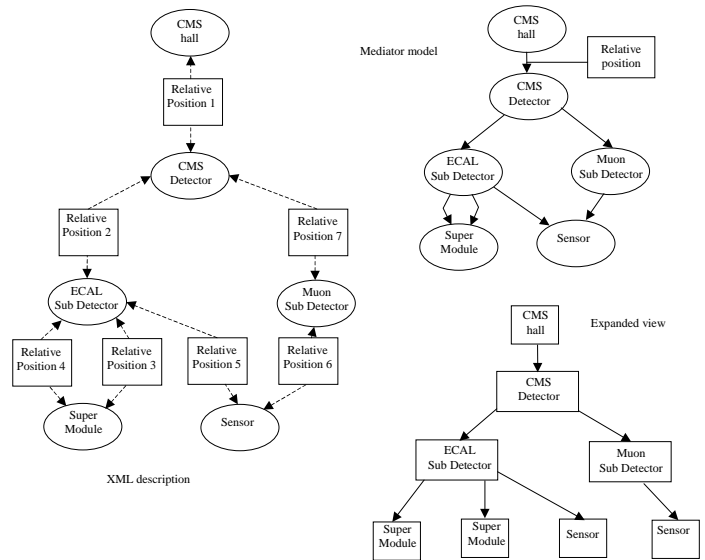


Figure 2. BOM

The XML schema standard provides several constraint mechanisms for data (key/keyrefs, the *choice* element, and maxOccur/minOccur attributes). But these constraint mechanisms were not sufficient for the detector description. Currently there is no constraint language standard for XML data, or applications that support a constraint language for XML. The Protégé project has defined a constraint language based on RDF [10]. However, this is restricted to RDF in XML format. [1] discusses a constraint interchange format in XML. Within the detector description schema a simple constraint language is used. It is used to define constraints on simple parameters (floats, doubles, and strings), using binary relations (<, ≤, >, ≥, =, ≠) and binary operators (∧, ∨).

Constraints are related to parameters of parts and part types within the BOM. The constraints reference directly to an element in the BOM, or a path is specified within the BOM that selects a set of parts, which are related to this constraint. The path specification is based on the Xpath specification that is used to navigate through XML trees. The following shows an example of a constraint:

```
//Part[@name=CMSDetector]//[Part[@name=SuperModule ^ @weight<2000 ^ @weight>1000 ^ @volume>500]
```

The part selector points to a set of parts (super modules that are part of the detector). The parameters *weight* and *volume* of these parts are used in constraints using an AND operator.

Within the detector description there are dependencies between data. For example: The value of attribute "width" of geometry X is three times the value of attribute "width" of geometry Y plus 2 times the value of attribute "width" of geometry Z. Currently it is not possible to specify this straightforward in an XML schema. Within our current prototype an attribute value is specified by attribute expressions are used. An attribute expression consists of numbers, operators (+,-,\*) and references to other attributes. The references are based on an Xpath notation. A constraint is that the Xpath should select one, and only one, attribute. If this is not the case, it would lead to ambiguities within attribute specifications. The following shows the preceding example in this notation:

```
<Geometry name="X"
width="3*Geometry[@name="Y"]/@width+
2*Geometry[@name="Z"]/@width"/>
```

A description of the detector is always an approximation. As such, different users construct different versions of the detector. Different versions can be based on the same parts. Within the schema of the detector description, the smallest versioned entity is an XML file. Within an XML file it is possible to define only one part or multiple parts (thus the smallest versioned entity can be a part). Users can construct configurations of XML files. This is a list of files that creates a description of "their" detector. Within these configurations it is also possible to "deselect" certain information from the XML files used within the configuration.

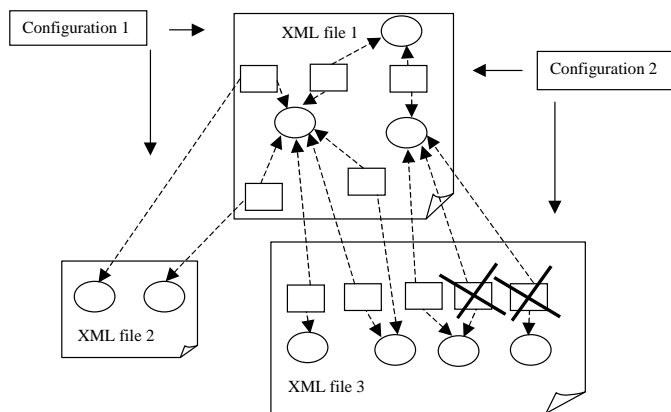


Figure 3. Different configurations

Figure 3 shows an example of two configurations. Configuration 1 uses XML file 1 and 2 to construct a BOM. Configuration 2 uses XML file 1 and 3. Furthermore, it deselects two parts in XML file 3. Constraints are also specified within XML files. Therefore the same mechanism for configuring parts is applied on constraints.

Client applications extract detector description data based on a configuration file. But parts of this detector description data

need to be accessed more frequently. Therefore, different client applications define different index structures on the same parts and part types within the detector description data.

#### IV. MEDIATOR

Clients of the detector description use this description data to build up an internal representation to perform their specific task more effectively. For example, simulation software needs to know the exact specification of the solids making up the geometrical hierarchy to be able to perform particle tracking through this hierarchy. On the other side, for visualization software, an approximation of solids is sufficient for displaying them (polygonalisation of curved solids). Both simulation and visualization make use of the geometrical hierarchy as specified in the XML BOM. Thus different clients have different AND common requirements on how to access the description data.

The BOM is represented as an acyclic directed multigraph its nodes and edges are used to attach description data. Clients will navigate this hierarchy, select the nodes and edges required by them and perform the necessary calculations and transformations to generate derived data. Tasks of the mediator are:

- a) Provide generic services for effectively navigating the BOM.
- b) Selecting nodes and edges, applying filter operations on them.
- c) Performing common used calculations and transformations for derived data.
- d) Merging non XML-based data and binding it to the BOMs nodes, edges and derived quantities thereof.
- e) Providing a Factory for instantiating its objects.

Mainly because of c) and d) XML based declarative query languages are currently not suitable to implement the mediator on top of them. The design of the mediator is independent of any XML technology. As a consequence an interface towards the XML based data has to be provided (e). This implies that the mediator manages its own internal representation of the detector description - its own BOM (CompactView). Towards the client software the mediator offers a set of domain specific interface-classes reflecting the BOM nodes and edges and data attached to them. The BOM becomes the CompactView of the detector. Many of these classes also have corresponding XML types and could in principle be generated from the schema. Data attached to the BOM nodes is again modeled by providing an appropriate class interface. As the client software is C++ software, the mediator itself is also implemented in C++.

XML instance documents containing data of the detector description are parsed by a standard XML parser (Apache Xerces). The content of XML instance documents can be directly converted into the mediators' representation of the BOM using the mediators' parser related factory interface. In the first stage XML elements are tranformed 1:1 to the

mediators representation, which keeps the semantic structuring of nodes and edges. In the next stage the mediators' representation is transformed to a type safe and validated BOM structure. References from edges to nodes are resolved.

To cover common requirements of the clients the mediator offers generic services to navigate and access data by:

*One to one BOM navigation.* The mediators representation of the BOM can be navigated directly using the usual graph navigation possibilities.

*Navigation in the expanded view.* In order to address individual parts of the detector (absolute positioned volumes) the BOM has to be expanded into a real tree. The BOM is expanded into a tree by first building its transposed graph and then unfolding it by transforming every possible path between two nodes into corresponding branches of the tree (expanded view Figure 2). The expansion to the tree is only done on demand. The mediator is used to navigate the expanded view using usual tree navigation where at each navigational step (first-child, next-sibling or parent) the corresponding node is calculated on demand from the BOM. In addition derived data such as the absolute spatial position of physical parts (nodes in the expanded view) can be calculated using the information of the relative positioning in the BOM edges.

*Selecting nodes (the PartSelector).* The mediator offers also services concerning the XPath like expressions (PartSelectors) for selecting nodes in the expanded view. Given a Partselector the mediator provides a list of expanded nodes corresponding to the selection. If the XML specification has defined a PartSelector with attached data, the mediator can provide this data for the appropriate nodes in the expanded view.

*Constraints checking.* The node selection capabilities are also used together with specified constraints among these selected nodes to validate against these constraints. As the constraints are specified in the expanded view of the detector it is currently not possible to use standard XML technologies for this kind of validations.

The current implementation of the mediator is C++ code that presents an interface to the clients. Data is retrieved by iteration over the nodes in the XML trees in different XML files. Data retrieval can also be done using a declarative approach. Xquery can transform the nodes of the schema instances into nodes that describe objects. These objects can than be de serialized into the applications.

Consider the following example based on two XML snippets:

```
<Section name="PartTypes">
  <PartType name="ECAL_sub_detector">
    <Geometry reference="Box1"/>
  </PartType>
  <PartType name="SuperModule">
    <Geometry reference="Trap1"/>
  </PartType>
</Section>
```

```
.....
</Section>

<Section name="Geometry">
  <Geometry name="Box1" type="box">
    <Dimensions x="3" y="4" z="5"/>
  </Geometry>
  <Geometry name="Trap1" type="trapezoid">
    <Dimensions Alp1="2" Alp2="6" B11="8"
    B12="8" Dz="1" Phi="0" H1="4" H2="1"
    Theta="50" Tl1="2 Tl2="3"/>
  </Geometry>
.....
</Section>
```

A simulation application needs data about parts with corresponding geometries. Within its internal model it uses an object SimPart that contains the part and the geometry. Furthermore, the attributes Theta and Phi define angles in degrees, while the simulation uses radians. An extension deg2rad performs this conversion. The following Xquery specifications translate the data into an object description in XML that can be de serialized into this application:

```
FOR $a IN
DOCUMENT("description1.xml")/Section[@name="PartTypes"]/PartType
  $b IN
DOCUMENT("s.xml")/Section[@name="Geometry"]
WHERE $a/Geometry/@reference=$b/@name
RETURN
<Object name="SimPart">
  <Attribute name="name" value=$a/@name
  type="string"/>
  IF ($b/@type="box") THEN
    <Attribute name="width" value=$b/@x
    type="float"/>
    <Attribute name="length" value=$b/@y
    type="float"/>
    <Attribute name="depth" value=$b/@z
    type="float"/>
  IF ($b/@type="trapezoid") THEN
    <Attribute name="Alp1" value=$b/@Alp1
    type="float" />
    <Attribute name="Alp2" value=$b/@Alp2
    type="float" />
.....
</Object>
```

Other applications need this data in a different form, because different objects are used within these applications. This leads to different Xquery specifications.

Currently data retrieval, transformation and de-serialization is done within the C++ code using an XML parser. A declarative approach splits this into data retrieval/transformation and de serialization. Xqueries can be stored using the XqueryX format and become more manageable [2]. A problem with Xquery is the problem of expressing data dependencies as discussed in section III.

## V. META DATA

The detector description will be stored in XML based on the XML schema described in section III. However, there are other applications that use different XML formats. For example:

- HTML. Used for displaying and browsing purposes.
- RDF. Used by Protégé.
- AGDD [3]. A format used by the ATLAS experiment for their detector description

If data can be transformed from the CMSDD format, in a coherent manner to other formats, it will be possible to use applications that support this other format. For example, Protégé has a powerful constraint language. This could be used to formulate constraints that cannot be expressed in the constraint language currently within the CMS detector description.

The transformations will be specified using XSL, Xpath and Xquery (since source and target are XML). If there are several transformations from/to different formats, users would like to get an overview of the possibilities for transformation. Furthermore, different groups within the CMS experiment can develop their own transformations.

The following meta data is attached to transformations:

- (Unique) transformation name
- Location
- Source format
- Target format
- Information loss. This is currently a parameter with one of the following values: None, Some, A lot
- Documentation. Describes the loss of information in human understandable terms.

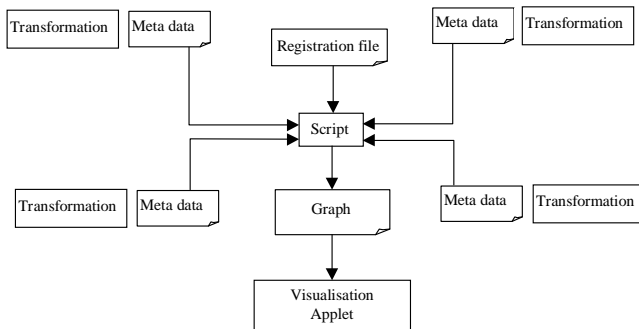


Figure 4. Meta data visualization

Applications such as HTML browsers, and Protégé can be seen as special kinds of transformers (transforming XML to a graphical representation for users). Within the meta data there is an attribute 'application' that is set true for these applications. Furthermore, the target format is empty. Figure 4 shows the architecture for visualizing meta data. The transformations are registered in a registration file. If a user requests an overview of the transformations, the script will consult the registration file and collect the meta data of the transformations. This meta data is combined into a graph.

There are two types of nodes in the graph. A type that represents transformations (applications without target included), and a type that represents different formats (e.g. HTML, RDF). Arrows connect formats and transformers. Figure 5 shows an example. Notice that CMSDD, RDF, HTML, and AGDD are all XML formats. This dependency information is not yet specified within the meta data. Furthermore it is assumed that transformations have one input source and one output source. Currently this is valid for the CMS domain.

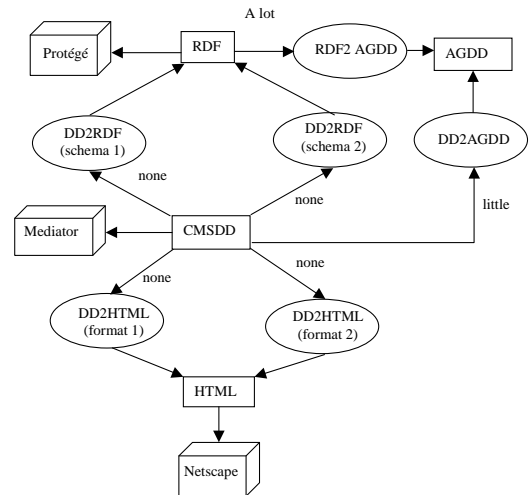


Figure 5. Combining meta data about transformations

Transformations will create new sources of "derived" or replicated data. The following meta data is attached to data sources:

- (Unique) source name
- Location
- Source format
- Source type (derived, original, replicated)
- If derived, from which sources, and which transformations
- If replicated from which source
- If it is a derived or replicated source when was it last updated

The architecture for visualizing this information is similar to Figure 4 (transformations are replaced by data sources). Figure 6 shows an example. Source 1 is the "original" source containing CMDD data. Source 2 is derived from this source based on the DD2RDF transformation (see Figure 5). Source 3 is a replica. Source 4 and 5 are derived sources based on RDF2AGGD and DD2HTML respectively (see Figure 5). The script creates a graph that is displayed via a Java applet.

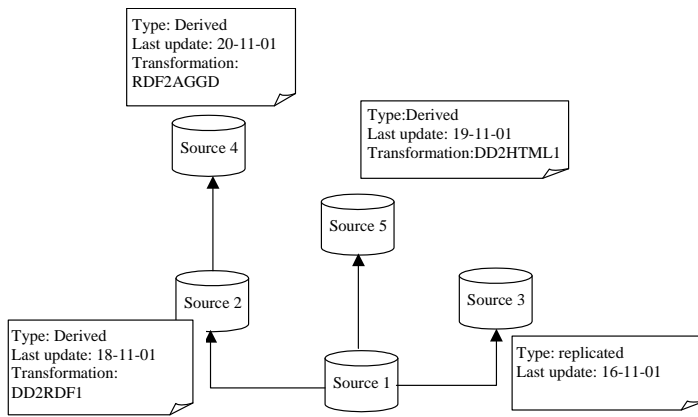


Figure 6. Combining meta data about data sources

These two graphs give the user an overview of what sources and transformations are available.

## VI. RELATED WORK

Within the High Energy Physics community, several projects deal with a detector description [3], [5], [6]. [3] is based on DTD instead of XML schema. This made it difficult to create models using an object oriented approach, and to enforce type safety. None of the approaches exploit the BOM structures that are found in detector description data. Furthermore, meta data (management) has not been identified.

Numerous projects dealt with data integration and mediators. Tsimmis [11] focused on semi structured data and rapid construction of mediators. MIX [12] used XML for information integration and exchange. A query language (XMAS) has been developed to define views. Tukwila [13] focuses on adaptive querying in XML documents. Several of the results are used in the current XML standards. The object exchange model for semi structured data is a predecessor of the XML format. The Xquery standard has been developed to define queries and integration specifications over multiple XML files.

The approach discussed in this paper uses these results based on XML standards. However, not every integration model used within this domain, can be described using standards such as Xquery or Xpath, (domain) specific extensions are needed for this.

Meta data is used in many areas. Dublin core [14] is a specification for meta data. It originated from the library community. RDF is a W3C standard for describing meta data.

## VII. FUTURE WORK

A first implementation exists of the CMSDD XML schema and mediator. Furthermore, a Java applet has been developed that visualizes graphs that are constructed from the meta data. A transformation has been developed to transform data from the initial sources to the CMSDD schema. Another transformation has been developed that transforms instances

of the CMSDD schema into HTML. Currently meta data is only used for visualization

Future work will focus on specifying transformations using XSL and developing a mediator (or mediators) that are based on Xquery specifications. Furthermore, a transformation will be developed to transform CMSDD schema instances to an RDF model that can be used within Protégé.

A problem encountered during the development of transformations is that not every transformation can be specified using the standard Xpath and Xquery language. Language extensions are needed to support the transformations specific for this domain. An example of such an extension is changing from Euclidean to polar co-ordinates.

Meta data is specified in plain ASCII files. This will be migrated to RDF. Currently meta data is used to visualize information about data sources to users. Future research will focus on the development of a constraint language for updating data. Such a specification will be part of the meta data. An example of such a constraint: "update this data source every day", or "update data source X if data source Y changes".

XML files are used as a storage format to store a detector description. However, this is difficult to manage if there is a large amount of data. A DBMS enables better management of this data. A relational or object oriented DBMS can export data using an XML interface [4], [8], [9]. An alternative for this is a DBMS for managing XML files such as Tamino [22].

An important requirement from the physics community is a DD format that is easy to understand and edit. This compromised the design of the schema: The references currently used to link data within different XML files are not based on Xlink/Xpointer. Furthermore, the configuration files "link" pieces of XML that form a "detector description" are not based on Xlink.

Future work will concentrate on defining a schema that exploits these standards. This would allow for the use of more open source software for managing and manipulating XML data. Physicists would edit/manipulate detector description data using the current schema definition. The data for this description is transformed from a description based on general XML concepts of describing data and relations between data (Figure 7).

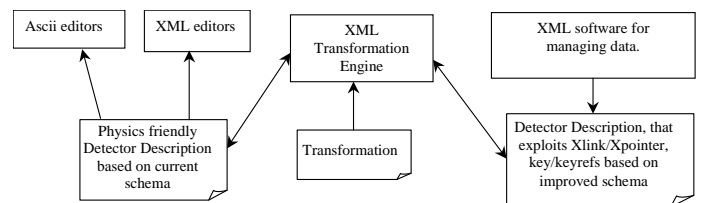


Figure 7. Transforming detector descriptions

XML schema is used to describe a model for the detector description domain. Furthermore, XML instances of this schema are used to store detector description data. Xpath and Xquery specifications are used for transformation and integration of data. Meta data from transformations and data sources is used for management of these different sources and transformations.

Numerous open source tools are available for XML (parsers, schema validators, transformation engines, and Xquery engines). This allows to focus on model development, data transformation/integration and meta data management without re implementing various services these tools offer.

#### ACKNOWLEDGMENTS

The authors like to thank the CMC group at CERN for numerous constructive discussions and feedback.

#### REFERENCES

- [1] Peter Gray, Kit Hui & Alun Preece, "An Expressive Constraint Language for Semantic Web Applications" IJCAI-01 Workshop on E-Business and the Intelligent Web, pp 46-53, 2001.
- [2] Frank van Lingen, Richard McClatchey, Peter van der Stok, Ian Willers, *XML for domain viewpoints* in proceedings of SCI 2001
- [3] Christian Arnault, Stan Bentvelsen, Steven Goldfarb, Marc Virchaux, Christopher Lester, "A generic approach to the detector description in Atlas", CHEP 2000
- [4] Carey, M. et al. 2000 "XPERANTO: Publishing Object-Relational Data as XML", Int. Workshop on the Web and Databases, Dallas, Texas
- [5] J. Bogart, D. Favretto, R. Giannitrapani, "XML for detector description at GLAST" Chep 2001
- [6] Radovan Chytracek, "The Geometry Description Markup Language" Chep 2001
- [7] M. Liendl, F. van Lingen, M. Case, T. Todorov, P. Arce, A. Furtjens, V. Innocente, A. de Roeck, "The Role of XML in the CMS Detector Description", in proceedings of CHEP 2001
- [8] Frank van Lingen, "XML interface for object oriented databases", in proceedings of ICEIS 2001
- [9] Bhatti, N. Le Goff, J.M. Hassan, W. Kovacs, Z. McClatchey, R. Martin, P. Stockinger, H. Willers, I. 2000 "Object Serialisation and Deserialisation Using XML", 10th International Conference on the management of Data (COMAD 2000) India,
- [10] N. F. Noy, R. W. Fergerson, & M. A. Musen. The knowledge model of Protege-2000: Combining interoperability and flexibility. 2th International Conference on Knowledge Engineering and Knowledge Management
- [11] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, J. Widom, "The TSIMMIS project: Integration of heterogeneous information sources", Proc. IPSJ Conference, pp 7-18, 1994
- [12] C. Baru, B. Ludäscher, Y. Papakonstantinou, P. Velikhov, V. Vianu, "XML-based Information Mediation", (position paper at QL98/W3C)
- [13] Zachary G. Ives, Alon Y. Levy, Daniel S. Weld, Daniela Florescu, Marc Friedman. Adaptive Query Processing for Internet Applications. IEEE Data Engineering Bulletin, Vol. 23 No. 2, June 2000.
- [14] Dublin Core Metadata Element Set, Version 1.0. Available as IETF RFC 2413
- [15] Xlink, <http://www.w3.org/XML/Linking>
- [16] XMLSchema, <http://www.w3.org/XML/Schema>
- [17] XSL, <http://www.w3.org/Style/XSL/>
- [18] XML, <http://www.w3.org/XML/>
- [19] Xpath, <http://www.w3.org/TR/xpath>
- [20] Xquery, <http://www.w3.org/XML/Query>
- [21] RDF, <http://www.w3.org/RDF/>
- [22] Tamino, <http://www.softwareag.com/tamino/>
- [23] CMS Technical Proposal. CERN/LHCC94-38 LHCC/P1.

[24] The LHC project Conceptual Design Report. CERN/AC/95-05(LHC).  
General information: <http://lhc.web.cern.ch/lhc/>