A generic approach to the detector description in Atlas

Christian Arnault[1]*, Stan Bentvelsen*[2]*, Steven Goldfarb*[3]*,*
Marc Virchaux[4]*, Christopher Lester*[5]*, The Atlas collaboration*[6]

[1] LAL, CNRS, France
[2] NIKHEF, NL
[3] Harrison M. Randall Laboratory, University of Michigan, USA
[4] DAPNIA, Centre d'Etudes de Saclay (CEA-Saclay), France
[5] High Energy Physics Group, Cavendish Laboratory, University of Cambridge, UK
[6] `http://atlasinfo.cern.ch/Atlas/Welcome.html`

Abstract

The problem of defining and managing the detector description for the Atlas Experiment at CERN is being addressed by applying a generic approach to the software design. This approach is based on the development of a central model, which is independent of both the application (simulation, reconstruction, visualisation, etc.) and the specificities of the subdetectors being described. In particular, the model permits one to provide a parametric description of both the physical elements and the logical organisation of the subsystems, including a generalised identification scheme.

Because of the generality of the approach, the W3C (World-Wide Web Consortium) standard XML[1] (eXtensible Markup Language) has been chosen for the persistent form of the description. The structure of the model, including the representation of the descriptive parameters, such as shape, dimension and material, is defined by the DTD (Data Task Definition) syntax. Using the XML standard accelerates the development process by providing a simple, easily distributed user interface for editing (ascii) and by giving access to a wide range of commodity tools for visualisation and validation. In addition, construction of the transient C++ objects is automated to some extent by making use of the DOM (Data Object Model). Work is currently active on applying the model to describe the digitization for the various clients of the detector description.

Keywords: XML, DetectorDescription, Patterns

1 Introduction

Constructing the software components related with the management and operation of a large and complex detector such as Atlas requires the sharing among these components of a precise and unambiguous description of the detector. This description must cope with several different aspects such as the logical organisation (how the detector is structured), the geometrical attributes (dimensions and positions), the physical properties (the materials), the read-out organisation and characterstics, or the identification scheme for any item of the detector.

All software activities (simulation, event-building, reconstruction, event-viewing, analysis, etc.) will eventually make use of such a description, and the uniqueness of the values as well as of the data models used to structure the values is a key factor to the software quality.

The success of such a consistent approach resides in the fact that the description is designed to be independent of the client software (e.g. simulation). This implies that different applications will view the same detector description with their own eye, building a specific representation, based upon its internal algorithms. Typical examples are the use of Geant4 for the simulation, persistency schemes (these models are generally constrained by the persistency environment such as Objectivity), or reconstruction algorithms based on regions, roads, etc.

We therefore decided to build a generic and open architecture to address this challenge, including:

- a set of generic concepts for parameterising all values of the description, so that only the basic values (those that cannot be algorithmically deduced from others) are considered in the description.

- a generic abstract object model for specifying detector description parameters and building the various software items
- a set of design patterns (based on conventional patterns such as visitors, iterators or factories) meant to construct converters between the generic model and the specific models.
- a primary textual representation of the generic model using the XML syntax.

2 The generic model

The generic model is an abstract object model representing the concepts to be used in constructing the detector description. It is meant to receive various implementations, such as C++, XML, Objectivity, Java, etc. Some of the concepts are today fairly well defined (the geometry, the material) whereas others are only roughly identified and understood (the identification scheme, the readout description).

We first define a main manager named `AGDD`, in which all the detector description elements (`materials` for the material elements and `sections` for the geometry elements) are maintained and managed.

2.1 Materials

These elements describe the material of the detector constituents:

- Materials can be `elements` or `composites`.
- `elements` are identified using a long name.
- `composites` are identified using a long name and provide a density. They are made of a set of `addmaterial` elements, each providing an IDREF towards an already defined material and the percentage of this material in the composite.

2.2 Sections

A section forms the basic sub-division of the entire detector. It holds the implementation of the geometry for this particular piece of detector. A section is primarily made of `volume` elements but also may receive a set of named `parameters` (used to specify constants).

The `volume` type is generic and effective volumes are organized as a hierarchy of types :

solid volumes	A basic geometry unit; Specialized types provide for specific shapes (boxes, tube sections, trapezoids and cone sections). A solid references a `material`, and may be declared as `sensitive` (which opens the not-yet-implemented capability of producing hits).
`compositions`.	Construct a new volume which consists of positioning several other volumes w.r.t the reference frame of this new entity. Generally, there is a priori no defined volume corresponding to this entity i.e. it is a juxtaposition of volumes the envelope of which has to be computed from the juxtaposed volumes. An explicit envelope volume may be specified, in which all positioned volume will be placed.
boolean volumes	The positioning of volumes within boolean operations (`union`, `intersection` or `subtraction`) can only be done via single positioners (ie. `posXYZ` and `posRPhiZ`). The materials of the boolean volumes should be identical.
axis compositions	Constructs a new volume which consists of several other volumes or dummy gaps piled up along one given axis (either `compositionX`, `compositionY` or `compositionZ`). The positioning of volumes proceeds via `pos`, `gap` and `mpos` elements.

Stacks	A stack is defined as a stacking of several solids along the stack-axis (along either
`stackX`	the X, Y or Z axis).
`stackY`	Stack entries (element tag "stack_entry") place solid volumes next to each other
`stackZ`	along the X, Y or Z axis (c.f. stackX, stackY and stackZ)

2.3 Positioning of volumes

Each volume (Solid volumes, compositions, unions, etc) is positioned relatively to other volumes using the `position` elements. A given volume may be referenced in several position operations.

Position operators are organized as a hierarchy of types as follows

Single	compositions boolean volumes	`posXYZ` : single positioning of a 'generic' volume, in cartesians coordinates. The volume is rotated before it is placed. `posRPhiZ` : single positioning of a 'generic' volume, in cylindrical coordinates. The volume is rotated before it is placed.
Multiple	compositions	`mposR/X/Y/Z` : multiple positioning of volumes along the R, X, Y, Z directions respectively. `mposPhi` : multiple positioning of a 'generic' volume, around the Z-axis at a given radius R, with incremental values of phi.
Axis	axis-based compositions	`pos` : a single volume is added to the pile. The effective thickness along the axis must be specified. And a rotation (one angle) along the same axis may also be specified. `gap` : a dummy spacer (with a thickness) along the axis. `mpos` : a multiple position of a given volume. The effective thickness of this volume must be specified, and a pitch may also be provided.

2.4 Identifiers

Identifiers provide for a generalized identification scheme of positioned volumes in the detector, the primary motivation being to consistently identify parts of the detector which provide the data.

Generally, an identifier is composed of a set of numbers such as `/1/3/4/5/1/34`. The meaning of each field is conventional and reflects the local hierarchy of volumes. The `identifier` elements permit to freely specify which field(s) (using conventional symbolic names) will be affected by the positioning operation. When single positioning is used, a single value of one or several fields will be affected by the operation. When a multiple positioning operation is considered, one or several fields will be iteratively affected. In this case, it is possible to follow the iteration by specifying a first value and a step used to compute iterated field values.

3 The XML representation

A textual representation using the general XML [1] syntax as well as a corresponding C++ model has been built after the generic model. A framework of utilities, based on these implemented models, and featuring factories and visitors are available for constructing user applications meant to derive specific views from the generic model.

Various parsers found as public domain software (`Expat`[3] and `XML4c` from IBM) have been evaluated to form the core mechanism onto which model converters are built. The `XML4c` family currently happens to be the richest one, by providing syntax checkers and Java interfaces.

4 The associated tools

4.1 Persint

The PERSINT [2] program can show and interact with the geometries written in an xml file.

Detector elements as well as hits and digits (or reconstructed quantities for some of the sensitive detectors) can be visualized, and interactive selection of components is possible, giving access to a complete identification of selected components as well as a full interactive 3D displacement of the viewing point or of the direction of sight of any viewed object

4.2 GraXML

GraXML is a prototype of the standalone visualization program for the AGDD XML files. This relatively modest project permitted a quite positive evaluation of the Java technologies (Java2, Java3D, XML4J and VRML2) for the 3D detector geometry visualization. The use of standards gave us access to very good tools for the 3D picture rendering itself.

GraXML only depends on information stored in the AGDD XML files. Thus, while verifying correctness of those files it ensures that there are no hidden parameters present. GraXML shows the same 3D geometry as other tools (Persint and G4) with a superior graphical quality (depending on the used rendering engine).

Further developments of GraXML are expected, especially on integrations with the general Atlas Graphics Architecture, or to introduce the interactive features of Java3D + VRML2.

4.3 G4Builder

A Geant4 builder is created, which is a 'client' of the generic model (described above). Using the generic model, it converts the materials and geometries of the xml files to Geant4 materials and Geant4 objects. The geometries can subsequently be visualised using the Geant4 visualisation package (DAWN). The G4builder visualises most of the AGDD syntax, as long as the appearance of the volumes in the xml file is in order (NO forward referencing). It currently visualises whatever is put in the volume 'ATLAS'. It cannot deal with boolean volumes as yet.

5 Conclusions

Working on a widely generic approach for the detector description was (and still is) a challenge, due to the complex nature of the Atlas detector. This project should be still considered in its very early phases. However, it already showed that this approach was quite valuable especially since it yielded quite constructive thoughts and discussions between detector experts for gaining a clear understanding of general concepts involved in this domain.

The use of XML based tools was satisfactory and does not (*yet*) imply exceeding complexity, although the lack of real object orientedness was sometimes felt as a limitation.

References

1 "The XML Web page", `http://www.w3.org/XML/`
2 "The Persint Web page", `http://atlasinfo.cern.ch/Atlas/GROUPS/MUON/persint.html`
3 "The Expat Web page", `http://www.jclark.com/xml/expat.html`
4 "The XML4C IBM Web page", `http://www.alphaworks.ibm.com/tech/xml4c`