

EVENT DATA MODEL IN ATLAS

E. Moyses, CERN, Geneva, Switzerland*
F. Åkesson, CERN, Geneva, Switzerland†

Abstract

The event data model (EDM) of the ATLAS experiment is presented. For large collaborations like the ATLAS experiment common interfaces and data objects are a necessity to insure easy maintenance and coherence of the experiments software platform over a long period of time. The ATLAS EDM improves commonality across the detector subsystems and subgroups such as trigger, test beam reconstruction, combined event reconstruction, and physics analysis. Furthermore the EDM allows the use of common software between online data processing and offline reconstruction. One important component of the ATLAS EDM is a common track class which is used for combined track reconstruction across the inner tracking detectors and is also used for tracking in the muon detectors. The structure of the track object and the variety of track parameters are presented. For the combined event reconstruction a common particle class is introduced which serves as the interface between event reconstruction and physics analysis.

INTRODUCTION

This report gives an overview of how the ATLAS EDM is constructed within the constraint of the ATLAS computing model and shows the benefits of the approaches taken. All subdetectors are represented in the EDM but here emphasis is given to the tracking and Inner Detector software, and these are used as examples.

THE ATLAS COMPUTING MODEL

The ATLAS detector [1] will produce up to one PetaByte of data per year, a vast amount of information which prohibits the wide distribution of raw data to worldwide collaborators. To enable physicists to analyse the data at remote sites two additional stages of datasets are introduced:

- The Event Summary Data (ESD) which contains the detailed output of the detector reconstruction and will be produced from the raw data. It will contain sufficient information to allow particle identification, track re-fitting, jet calibration etc. thus allowing for the rapid tuning of reconstruction algorithms and calibrations. The target size for the ESD is 500 kB per event.
- The Analysis Object Data (AOD) which is a summary of the reconstructed event, and contains sufficient information for common analyses. Several tailor-made

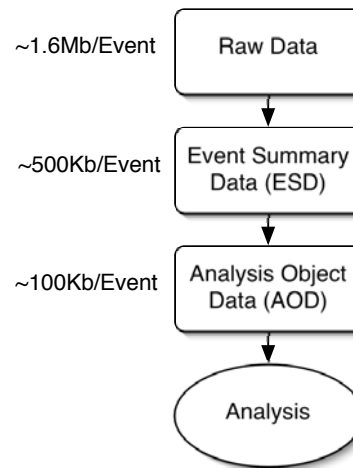


Figure 1: The layers of the ATLAS computing model, each smaller than the last.

streams of AOD's are foreseen for the different needs of the physics community. The AOD can be produced from the ESD and thus makes it unnecessary in general to navigate back and process the raw data, adding significant cost and time benefits. The target size for the AOD is 100 kB per event.

Inevitably, there will be some overlap between the different reconstruction realms: for example, some objects will exist in both AOD and ESD.

There will also be "tags" on each event, indicating some general features of the event, and thus allowing the quick access of the required events. The target size for the tags is 1kB per event.

REQUIREMENTS FOR THE EDM

The ATLAS EDM is shaped by many considerations: it must allow the correct level of modularity to fulfill the constraints of the computing model with respect to the differentiation between raw data, ESD and AOD.

It must also fulfill the requirements of the Athena software framework [4] (based on GAUDI) used by ATLAS. The ATLAS EDM must interact cleanly within this framework and the associated tools and services it provides. Moreover the EDM must follow ATLAS coding standards, such as enforcing the separation of event and non-event data - e.g. by avoiding having detector description¹ in the

* edward.moyse@cern.ch

† fredrik.akesson@cern.ch

¹ATLAS uses GeoModel for its detector description

event data. In fact in Athena there are different types of storage used for transient event data (i.e. data that only exists for an event) and data with a lifetime of the run. The ATLAS transient event store has two instances, the short term storage is called StoreGate, whilst the longer term store is called DetectorStore.

The EDM must be persistifiable: ATLAS uses POOL [5] to read and write data to disk, and therefore it must be possible to store all EDM object in POOL format. This is a non-trivial requirement and has set serious constraints on the allowed designs of the EDM. For example, links between persistified objects have to be possible. Normally this could be done with pointers and references (for example, linking a track to the measurements used to produce it) but these cannot easily be persistified and so "DataLinks" must be used instead. Links across levels (i.e. from AOD to ESD) are another necessary complication, and are restricted.

ATLAS has adopted CLHEP [5], and therefore the EDM must support it: this has then benefit of promoting code-readability, and portability, since all of ATLAS uses the same classes. However it has also caused problems with persistency (some CLHEP classes were not originally persistifiable, and were unnecessarily large when written to storage).

Finally, it must be possible to navigate from the EDM data object to the underlying simulated event (i.e. it must be possible to access the 'truth' from the EDM objects).

Above and beyond all these technical requirements, the EDM must promote code re-use by allowing the factoring out of common tools and common data objects. For example, data objects should, if possible, be shared between the online trigger (which has strong requirements, such as being able to run in a multi-threaded environment and speed) and the offline reconstruction software, as well as between the various sub-detector systems. At the same time the EDM must minimise unnecessary dependencies.

THE ATLAS DETECTOR

ATLAS has two types of sub-detector systems: trackers (the Inner Detector [6] and Muon Spectrometer [7]), which measure momenta of charged particles, and calorimeters (Tile [3] and Liquid Argon [2]), which measure energy depositions. As mentioned before, a major aim in the design of the EDM is to share as much code as is possible within these common sub-system types.

Calorimeters

The two types of calorimeter have different data formats at the raw data level, however for reconstruction the EDM uses one common calibrated input object, "CaloCell". CaloCells can be generated either from the raw data or simulation. For example, fig. 2 (which is a schematic representation of the calorimeter reconstruction chain) shows the raw data being fed to "CellMaker" algorithms, which

produce CaloCells. From this moment on data classes are common to both calorimeter types.

Neighbouring CaloCells are used (by "CaloTower-Maker") to produce calorimeter "towers", then these towers (as well as cells) are taken (by "CaloClusterMaker") to construct "clusters", collections of calorimeter elements (which can even contain clusters themselves).

A navigation scheme allows access to constituent data objects e.g. it is possible to retrieve all the CaloCells used to create an EnergyCluster.

All calorimeter data classes inherits from a four-momentum interface which allows the use of common tools only requiring kinematic information.

Tracking Detectors

As with the calorimeter, a basic requirement for the EDM is to support different tracking devices with shared code, e.g. the muon chambers and drift tubes, the inner detector transition radiation tubes and silicon detectors must all be provided for by common tracking software.

The most obvious requirement is a common track class, but more than that, the EDM needs standard definitions of:

- Track parameters (on all the various surfaces found along the track);
- Interfaces to hit-clusters, drift circles, etc;

Tracking must handle many different coordinate frames, as a track can span the entire detector and have measurements on many different surfaces (i.e. discs, planes, cylinders, and so on). However, the various tracking tools and algorithms must not be expected to handle the geometry of the detector. Generalised tools allow tracking to work on both the Inner Detector and the Muon Spectrometer tracks. This can best be explained with the aid of fig. 3, which shows an overview of the Tracking reconstruction chain.

Bytestream convertors take the data from the detector, and form the raw data objects. These are then used to create "prepared raw data", i.e. clusters (for example, from the pixel detector) or drift circles (for example, from the muon monitored drift tubes).

Some of the tracking sub-detectors return what are essentially one-dimensional measurements, so these must be combined to form two-dimensional "SpacePoints".

The "PrepRawData" (along with the SpacePoints) can then be used to find tracks. Finally, the tracks can be used to find vertices, or to create the TrackParticles (for physics analysis at the AOD level).

Common Track One of the most important elements in the ATLAS EDM is the common Track. It must work in a wide range of applications, from:

- online (where speed and the ability to work in a multi-threaded environment are important requirements);
- alignment studies (which need very detailed information);

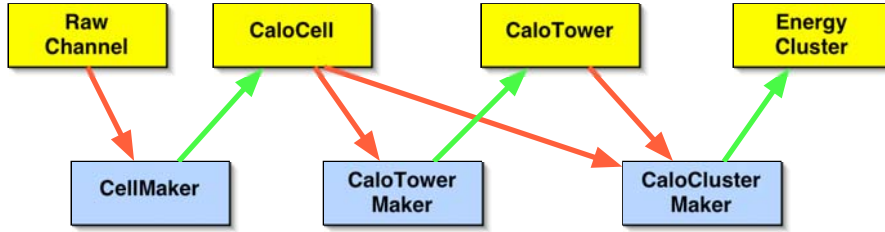


Figure 2: Schematic diagram of calorimeter reconstruction. The top line contains the data objects, whilst the bottom line shows the algorithms used to process them. Data flows from left to right.

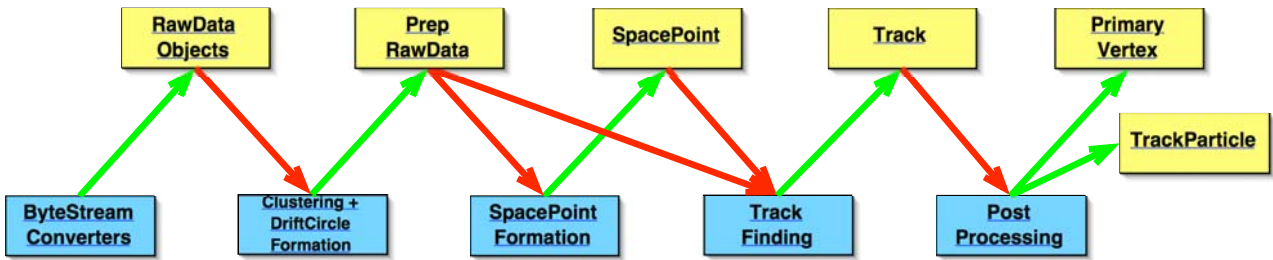


Figure 3: Tracking reconstruction chain. The boxes on the top represent data objects, whilst the boxes on the bottom show the algorithms which work on them. The arrows show the direction of data flow.

- general reconstruction.

Tracks at ESD level consist of fitted measurements on multiple surfaces, and are the output from the fitters, and the input to the combined reconstruction (all reconstruction packages should use the same track class).

These tracks are (necessarily) relatively large objects and for AOD something more lightweight is needed: therefore “TrackParticles” are created from Tracks. These objects contain summary information about parent track (number of hits on track etc), as well as the perigee parameters.

They are physics analysis objects with 4-momenta in the physics frame, and therefore (as with the calorimeter data objects) inherit from the common momentum interface, I4Momentum.

They can be used for vertex finding, but not re-fitting etc. (as the hits/measurements are missing).

Common Tracking Tools The common tracking tools are not really the focus of this report, but the ability to easily write them is both an important consequence and an important requirement of the EDM. Currently, there are:

- Track fitters - all fitting tools must inherit from a common interface. At the moment there are two fitters provided, but it is expected that more will follow;
- Extrapolator - this handles the propagation and extrapolation of track parameters (and their errors) to arbitrarily oriented surfaces [8];
- Vertexing - there are generic mathematical tools provided in Tracking, with specific implementation available in the InnerDetector area [9];

- Visualisation - it is possible to use visualisation tools, to examine the surfaces (i.e. using HepVis) and the tracks, and the tracking hits (i.e. with Atlantis), which is very useful for debugging purposes.

Fitters are inherently ESD-level tools (as they require the hits on the tracks) whilst the vertexing, visualisation and extrapolation tools can work on either AOD-level TrackParticles or ESD-level Tracks

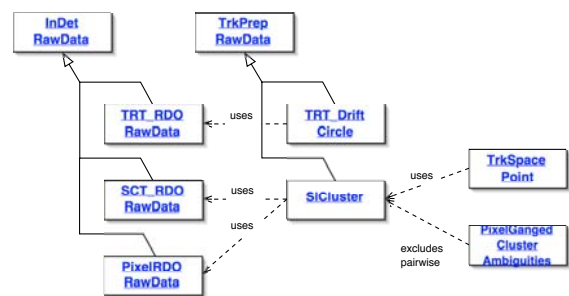


Figure 4: Inner Detector Input EDM. On the left are the various raw data classes, all inheriting from InDetRawData. They are used to create the Inner Detector “Prepared Raw Data” classes, TRT_DriftCircle and SiCluster. SiCluster is then used to form SpacePoints, and ambiguities in the SiClusters are resolved by PixelGangedClusterAmbiguities

Inner Detector Fig. 4 shows the EDM for the Inner Detector. Raw data classes (TRT_RDORawData, SCT_RDORawData and PixelRDORawData) inherit from a common base class (InDetRawData) and are used to create the “prepared raw data” classes, TRT_DriftCircle and

SiCluster, which are then used for track finding. Some SiClusters are then used to create SpacePoints (as explained above). Finally, because the readout from the ATLAS pixel detector elements are ganged together, there is some ambiguity between where a pixel hit actually is in space. This is resolved by track finding, and the result is stored using the PixelGangedClusterAmbiguities class.

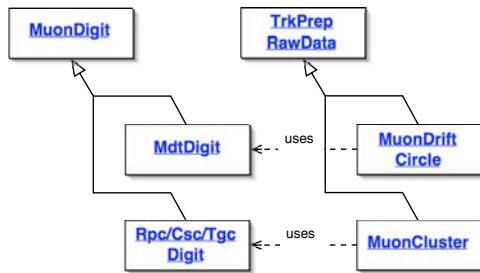


Figure 5: *Muon Input EDM*. On the left are the various raw data classes, all inheriting from *MuonDigit*. They are used to create the Muon Spectrometer “Prepared Raw Data” classes, *MuonDriftCircle* and *MuonCluster*.

Muon Detectors Fig. 5 shows the EDM for the Muon Spectrometer. As with the Inner Detector, the raw data classes (*MdtDigit* and the cluster digits — *Rpc,Csc* and *Tgc*) inherit from a common base class, *MuonDigit*. These raw data classes are used to create the “prepared raw data” classes, *MuonDriftCircle* and *MuonCluster*, which are used for track finding.

Analysis Objects

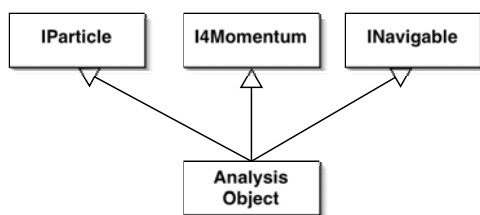


Figure 6: *Representation of an analysis object, inheriting from the *I4Momentum*, *IParticle* and *INavigable* interfaces. Examples of analysis objects would be muons, bjets, taus etc.*

Fig. 6 shows a generic analysis object. Since it represents a physical object, it inherits from *I4Momentum* and *IParticle*, whilst the *INavigable* interface allows (in the same manner as the calorimeter objects) navigation between constituent objects.

Tools which only require kinematic information will just use the *I4Momentum* interface, whilst other, more analyses might need more detailed information. In any case, the use of common interfaces dramatically simplifies the analysis code.

Trigger

The ATLAS trigger [10][11] is responsible for the online event selection. As such, a minimum requirement is that the EDM stores the trigger criterion (or hypotheses) which were passed for each level of the trigger (Level-1, Level-2 and Event Filter). Beyond that, if space permits it would be useful to store sufficient information to allow the trigger algorithms to be re-run, for example for Level-2 this is:

- Track parameters
- Spacepoints on the track
- Calorimeter Clusters
- EM shower information

CONCLUSION

ATLAS Event Data Model uses abstract interfaces and shared data objects, to allow the easy development of common tools. The EDM is now stable and fully supports the ATLAS computing model. Further refinements on the exact composition of the ESD and AOD will be needed, but the ATLAS EDM itself provides a solid basis for the further development of analysis code, and data-taking with ATLAS.

REFERENCES

- [1] ATLAS Collaboration, Technical Proposal for a General-Purpose pp Experiment at the Large Hadron Collider at CERN, CERN/LHCC/94-43
- [2] ATLAS Liquid Argon Technical Design Report , CERN/LHCC/96-41, December 1996
- [3] The ATLAS Tile Calorimeter Technical Design Report , CERN/LHCC/96-42, December 1996
- [4] Athena Developer Guide (draft), version 2, Athena website, <http://atlas.web.cern.ch/Atlas/GROUPS/ SOFTWARE/OO/architecture/General/index.html>
- [5] D. Düllmann *et al.*, The LCG POOL Project: General Overview and Project Structure, CHEP 2003 Proceedings, MOKT007
- [6] ATLAS Inner Detector Technical Design Report, CERN/LHCC 97-16, April 1997
- [7] ATLAS Muon Spectrometer Technical Design Report, CERN/LHCC/97-22, May 1997
- [8] A. Salzburger, The new Extrapolation Package in the ATLAS Tracking Realm, Presented at CHEP 2004
- [9] A. Wildauer and F. Åkesson, Vertex Finding and B-tagging Algorithms for the ATLAS Inner Detector, Presented at CHEP 2004
- [10] ATLAS First-Level Trigger Technical Design Report, ATLAS TDR-12, June 1998
- [11] ATLAS High-Level Trigger Data Acquisition and Controls Technical Design Report, ATLAS TDR-16, June 2003