A software framework for Data Quality Monitoring in ATLAS

S Kolos[1,4], A Corso-Radu[1], H Hadavand[2], M Hauschild[3] and R Kehoe[2]
[1]Department of Physics and Astronomy, University of California, Irvine, 4129 Frederick Reines Hall Irvine, CA 92697-4575
[2]Department of Physics, Southern Methodist University, Dallas, TX 75275
[3]CERN, CH-1211 Geneva 23 Switzerland

Abstract. Data Quality Monitoring (DQM) is an important and integral part of the data taking and data reconstruction of HEP experiments. In an online environment, DQM provides the shift crew with live information beyond basic monitoring. This is used to overcome problems promptly and help avoid taking faulty data. During the off-line reconstruction DQM is used for more complex analysis of physics quantities and its results are used to assess the quality of the reconstructed data. The Data Quality Monitoring software Framework (DQMF) which has been provided for the ATLAS experiment performs analysis of monitoring data through user defined algorithms and relays the summary of the analysis results to the configurable Data Quality output stream. From this stream the results can be stored to a database, displayed on a GUI, or used to make some other relevant actions with respect to the operational environment i.e. sending alarms, stopping the run. This paper describes the implementation of the DQMF and discusses experience from usage and performance of the DQMF during ATLAS commissioning.

1. Introduction

ATLAS [1] is one of the four experiments at the Large Hadron Collider (LHC) [2] at CERN. This experiment has been designed to study a large range of physics including searches for previously unobserved phenomenon such as the Higgs Boson and super-symmetry. ATLAS detector has about 140 million electronic channels and detects particles at 40 MHz bunch crossing rate which is then reduced by three levels trigger system down to the 200 Hz of recorded event rate. In order to monitor the status of the ATLAS detector and trigger system in an efficient manner it is necessary to have automatic tools for data checking. In the online environment, the outcome of such tool can alarm the shifter to prevent taking faulty data. In the offline environment, more complex checks can be done in order to classify events which are suitable for different types of physics studies.

Efficient, flexible and reliable automatic monitoring of data is especially important at the beginning of an experiment, when the environment is new, in order to help the experiment quickly determine problems and then proceed to solve them efficiently. DQMF has been designed to provide this functionality.

[4] On leave from Petersburg Nuclear Physics Institute, Gatchina Leningrad distr. 188300 Russia

2. General description
The Data Quality Monitoring Framework (DQMF) is based on the idea of applying analysis algorithms to various types of online monitoring data (histograms, messages, counters, etc.) according to a particular configuration, which is defined by detector experts. Each of these algorithms produces a piece of information called Data Quality Result. A DQ Result contains data quality assessment flag for a given monitoring data as well as some additional information which can be used to help understanding the result. DQ Results may be used for generating alarms when deviations from the standard are encountered. A summary of these results is displayed to the shifter and is also archived for future references. Using this information, detector experts can make a final data quality assessment for a given run. The archived results will permit a check or refinement of this assessment offline. The general architecture of the DQMF is shown on Figure 1.

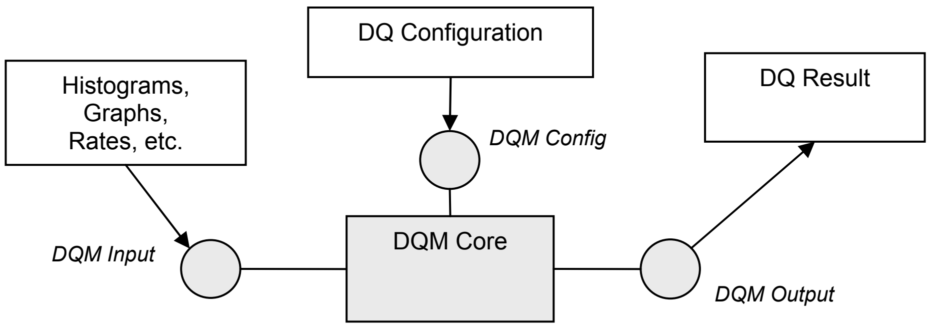


Figure 1. General structure of DQMF.

The functionality of executing DQ algorithms is provided by the DQM Core component which has three abstract interfaces for the communication with the external systems as:

- DQM Input is an interface for receiving monitoring data which is then passed by the DQM Core to the DQ Algorithms;
- DQM Output specifies a way of publishing DQ Results produces by the DQ algorithms;
- DQM Config interface provides a way of reading configuration information which defines behaviour of the DQM Core in a specific environment.

This approach allows reusing the same DQM Core in different environments where for example input data are coming from different sources.

3. DQ Configuration
The DQ Configuration is described as a hierarchical tree of objects of two different types: DQ Regions and DQ Parameters. This configuration approach is meant to be general in its accommodation of system description. For instance, while there are clear hierarchies of organization of data quality results for sub-detector elements, other vantages are possible. One can consider data quality results that are organized around some physics quantities like Energy and EtaPhi distribution for the Calorimeter as it is shown on Figure 2.
Another example for the ATLAS detector data quality, is a Jets hierarchy under an overall Jets DQ Region. There might be Calorimeter, Tracking, and Calibration DQ Regions under this. The lowest level under Calorimeter might include Width and EM fraction DQ Regions. The users, whether they are detector, offline reconstruction, or trigger algorithm experts will determine the specific hierarchies actually chosen.

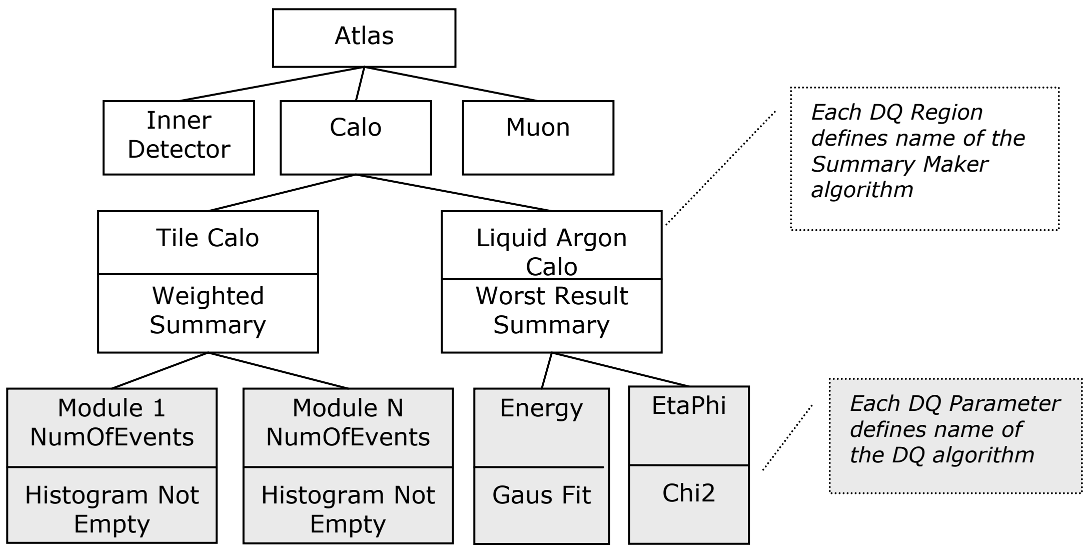


Figure 2. DQ Configuration example.

3.1.1. DQ Parameter

A DQ Parameter object represents the element of the ATLAS detector whose state can be assessed using single monitoring information item, i.e. a particular histogram or information object. A DQ Parameter object is responsible for applying the DQ algorithm defined for that object to a particular piece of monitoring data with the aim of producing the DQ Result for the corresponding system element. The description of the DQ Parameter is taken from the DQ Configuration and provides the following information:

- the location of the monitoring information that represents the state of a particular detector element;
- the weight (or degree of importance when calculating the DQ Result) for that DQ Parameter;
- the DQ Algorithm that has to be used for evaluating the DQ Result of that element;
- specific parameters and thresholds for the given DQ Algorithm e.g. fit parameters, minimum statistics, etc;
- the reference values or histograms;
- the actions which have to be taken depending on the results of the DQ Algorithm execution.

3.1.2. DQ Region

An object of the DQ Region class represents a certain subset of the ATLAS detector system elements and contains a set of DQ Parameters or low-level DQ Regions, which correspond to these elements. This class is used for representing a self-contained part of the system, like a detector, a sub-detector, a sub-farm, etc. For any particular DAQ partition all the available DQ Regions and DQ Parameters are organized into a tree with the root node representing the state of the whole system. This state is defined by the DQ Result calculated as a combination of the DQ Results for the contained DQ Parameters and DQ Regions. The rules for DQ Result calculation (for a given DQ Region) are provided by the DQ Summary Maker algorithm which is associated with that region. A possibility to define different summary algorithms for different DQ Regions is supported.

4. DQ Algorithms

The DQMF provides monitoring data analysis and DQ Results production be means of executing special analysis functions called DQ Algorithms. For supporting algorithms execution DQMF provides the DQ Algorithm generic interface which must be implemented by any specific kind of operation that can be applied to the monitoring data in order to evaluate a status of the corresponding

ATLAS system elements. DQMF provides a number of predefined DQ Algorithms for the most common operations like histogram comparison, histogram fitting, thresholds application, etc. DQ Algorithms are integrated into the DQM Framework in a dynamic plug-in manner which allows adding new algorithms on the fly without modifying the core software.
Each DQ Parameter has at least one DQ Algorithms associated with it. Such algorithm is executed whenever a piece of information which is associated with that DQ Parameter becomes available. Being executed a DQ Algorithm is responsible for producing the DQ Result object for the given DQ Parameter. In addition to that DQ Algorithm can also produce some extra information during its execution such as numerical results (e.g. fit results) or a difference histogram.

4.1. DQ SummaryMaker

The DQ SummaryMaker is a special implementation of the DQ Algorithm interface that evaluates the DQ Result for a given DQ Region. To calculate a new DQ Result the DQ SummaryMaker object uses the DQ Results produced by the DQ Regions and DQ Parameters that belong to the given region. The DQMF provides two predefined summary maker algorithms but custom ones can also be plugged into the framework in the same way as a normal DQ algorithm.

4.2. DQM Workbench

The algorithms which are running in the framework can also be used standalone in ROOT with the help of some ROOT scripts which are provided by the DQM Framework. These set of scripts is called DQM Workbench since it provides a convenient environment in which developers can easily test their custom algorithms without setting up any configuration database and without a necessity to set up the DQM Framework manually. The aim of the workbench is to help to debug custom algorithms and also to prepare the best configuration for a given DQ Parameter in terms of choosing the correct values for the algorithm thresholds and parameters. The tuning of the DQ Parameter objects can be very sensitive and developers need this experience to understand the behaviour before the algorithm is plugged into the DQMF.

5. Implementations

The DQMF has been implemented on C++ programming language using ROOT [3] for the DQ Algorithms implementation. Due to the fact that DQMF communicates with the external systems via well defined abstract interfaces it has being used without any modifications in both offline and online environments. The difference was the implementation of the DQM Input, DQM Output and DQM Config interfaces as it is shown in Figure 3.

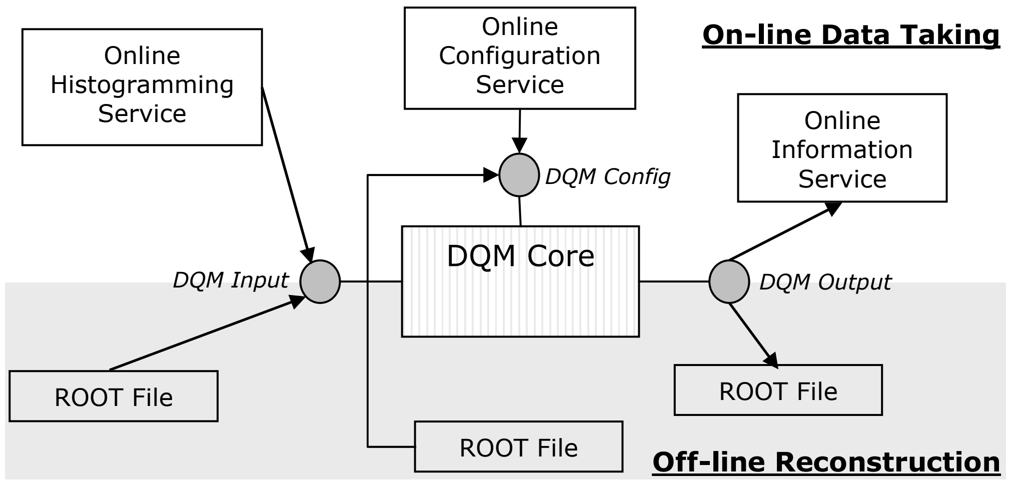


Figure 3. Online and Offline implementations of the DQMF.

5.1. Online DQM
In the online environment the implementation of the DQMF interfaces is based on the Online Monitoring [4] and Configuration [5] services provided by the ATLAS Trigger/DAQ system [6], namely:

- DQM Input interface is implemented using the Online Histogramming Service [4] which provides access to all the histograms produced by detector or DAQ applications during data taking;
- DQM Output implementation publishes DQ Results to the Information Service [4] from where they are accessible for any other application which is running within the ATLAS control network;
- DQM Config interface implementation is based on the OKS persistent object manager [5] which stores configuration information in XML files while providing C++ API for reading and modifying it.

5.1.1. DQMF Agent
In the online environment the DQM Core engine is hosted by the binary application called DQMF Agent. DQMF may contain one or more DQMF Agents with each of them responsible for a well defined subset of the whole ATLAS system. In terms of the DQ Configuration each DQMF Agent is responsible for one of the DQ Regions and naturally for all the sub-regions and DQ Parameters which belong to it. Each DQMF Agent instantiates appropriate implementations of the DQMF generic interfaces (i.e. DQM Input, DQM Output and DQM Config) and takes care of starting and stopping the DQM Core engine in appropriate moments. In the online environment the DQ assessment has to be started at start of run event and stopped when the run is finished and the DQMF Agent makes it possible by interacting with the ATLAS DAQ Run Control service [7].

5.1.2. Scalability
Since each DQMF Agent application is responsible for a well defined subset of the DQ Configuration, different agents are fully independent of each other. This makes possible to address scalability issue in quite an easy way by just increasing number of agent and deploying them on distinct computing nodes whenever an additional computing power becomes necessary.

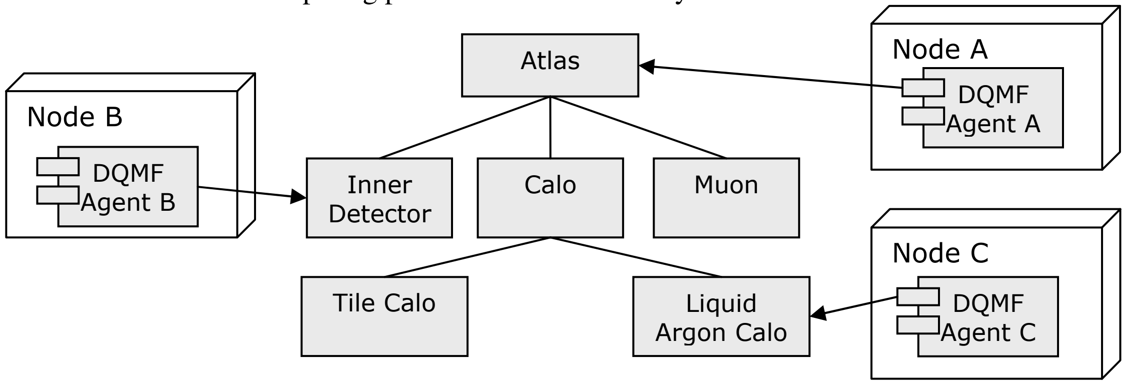


Figure 4. Example of DQMF Agents distribution on three computing nodes.

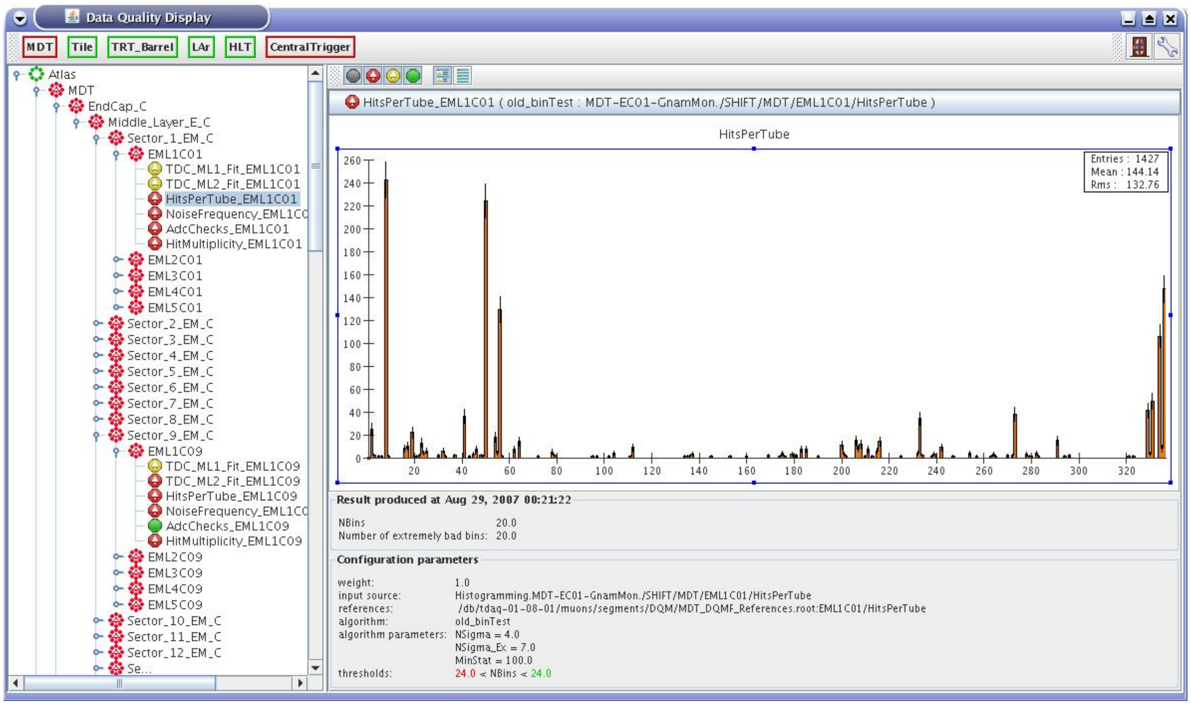
Figure 4 shows how responsibility can be distributed between different DQM Agent processes. For example initially there could be only one agent running on the Node A and handling the whole configuration. As number of DQ Parameters for different regions grows more computing power might become necessary. In order to address that the online implementation of the DQ Configuration allows defining one more agent that will be associated with the “Inner Detector” DQ Region and will run on the Node B. In this case the new agent will be responsible for executing DQ Algorithms for the “Inner Detector” DQ Region and for all its children. This work will be offloaded from the Top level DQMF Agent that will only get the final DQ Result from the DQ Agent B. In similar way the DQMF Agent C can be used for handling the "Liquid Argon Calo" DQ Region if more CPU will be required for analysing the DQ Parameters defined for that region. This process can go as far as assigning one DQMF Agent for each DQ Region which guarantees that scalability will never be an issue for the system.

5.2. Offline DQM

The offline implementation of the DQM framework is still under development for the purpose of being used at the ATLAS Tier-0 centre for the DQ assessment during offline event reconstruction. It is similar to the online version in functionality but uses ROOT file based implementation for the DQ Input, DQ Output and DQ Config interfaces.

6. DQMF at the ATLAS commissioning

The online version of the DQMF framework has been used during ATLAS combined cosmic runs which have been organised as part of the ATLAS commissioning procedure for the ATLAS sub-detectors integration, testing and calibration. The DQMF configuration which has been used in the August 2007 cosmic runs included more then 1000 DQ Parameters, which have been handled by a single DQMF Agent application. Results have been visualised to the shift crew by the prototype of the DQMF Display which has been developed for that purpose. The display was implemented on Java using JAIDA [8] library for histogram visualisation. Figure 5 shows a snapshot of it.



```
Result produced at Aug 29, 2007 00:21:22

NBins                         20.0
Number of extremely bad bins: 20.0

Configuration parameters

weight:                 1.0
input source:           Histogramming.MDT-EC01-GnamMon./SHIFT/MDT/EML1C01/HitsPerTube
references:              /db/tdaq-01-08-01/muons/segments/DQM/MDT_DQMF_References.root:EML1C01/HitsPerTube
algorithm:              old_binTest
algorithm parameters:   NSigma = 4.0
                        NSigma_Ex = 7.0
                        MinStat = 100.0
thresholds:             24.0 < NBins < 24.0
```

Figure 5. Snapshot of the DQMF Display from the ATLAS commissioning run.

On the left hand side the Display shows the hierarchy of the DQ Regions and Parameters as defined in the DQ configuration for the particular data taking session. On the right hand panel the configuration parameters and DQ status result for the selected DQ Parameter has been displayed.

7. Conclusion

The Data Quality Monitoring Framework which has been implemented for the ATLAS experiment provides an efficient, light weight and portable solution for the data quality assessment in both offline and online environments. The framework is very flexible due to the simple yet powerful configuration system which it is using for defining how to produce the DQ results. The online implementation of

DQMF is also fully scalable with respect to the number of checks which have to be performed on the monitoring data.
The first implementation of DQMF has been available since January 2007. Apart from the implementation of the DQM Core engine it also includes about thirty DQ Algorithms, two DQ Summary Makers, and DQM Workbench for developing and testing algorithms. The online DQMF has been used during ATLAS commissioning activities since March 2007. A prototype of the offline implementation has also been provided recently and started to be used in Tier-0 centre from August 2007.

8. References

[1] ATLAS Collaboration 1999 ATLAS detector and physics performance technical design report, *CERN Preprint* CERN-LHCC-99-014.
[2] ATLAS Collaboration 2004 LHC Design Report *CERN Preprint* CERN-2004-003-V-1
[3] Brun R and Rademakers F 1998 ROOT: An object oriented data analysis framework *Linux Journal* **51**
[4] Vandelli W et al 2007 Strategies and Tools for ATLAS Online Monitoring *IEEE Trans. Nucl. Sci.* **54** 609-615
[5] Soloviev I et al 2004 The Configurations Database Challenge in the ATLAS DAQ System *Proc. of Computing in High Energy Physics* (Interlaken, Switzerland) 101-104
[6] Gorini B et al, 2006 The ATLAS Data acquisition and High-Level Trigger: concept, design and status *Proc. of Computing in High Energy Physics* (Mumbai, India)
[7] A Kazarov et al 2007 A Rule-Based Verification and Control Framework in Atlas Trigger-DAQ *IEEE Trans. Nucl. Sci.* **54** 604 - 608
[8] Dönszelmann M, Johnson T, Turri M and Serbo V 2004 AIDA, JAIDA and AIDAJNI : Data Analysis using interfaces *Proc. of Computing in High Energy Physics* (Interlaken, Switzerland) 445