The Data Quality Monitoring Software for the CMS experiment at the LHC: past, present and future

Virginia Azzolini[1], Broen van Besien[2], Dmitrijus Bugelskis[3], Tomas Hreus[4], Kaori Maeshima[5], Javier Fernandez Menendez[6], Antanas Norkus[3], James Fraser Patrick[5], Marco Rovere[2], Marcel Andre Schneider[2]

[1] Massachusetts Inst. of Technology (US)
[2] CERN
[3] Vilnius University (LT)
[4] Universitaet Zuerich (CH)
[5] Fermi National Accelerator Lab. (US)
[6] Universidad de Oviedo (ES)

E-mail: `marcel.andre.schneider@cern.ch`

Abstract. The Data Quality Monitoring software is a central tool in the CMS experiment. It is used in the following key environments: (i) Online, for real-time detector monitoring; (ii) Offline, for the prompt-offline-feedback and final fine-grained data quality analysis and certification; (iii) Validation of all the reconstruction software production releases; (iv) Validation in Monte Carlo productions. Though the basic structure of the Run1 DQM system remains the same for Run2, between the Run1 and Run2 periods, the DQM system underwent substantial upgrades in many areas, not only to adapt to the surrounding infrastructure changes, but also to provide improvements to meet the growing needs of the collaboration with an emphasis on more sophisticated methods for evaluating data quality. We need to cope with the higher-energy and -luminosity proton-proton collision data, as well as the data from various special runs, such as Heavy Ion runs. In this contribution, we will describe the current DQM software, structure and workflow in the different environments. We then discuss the performance and our experiences with the DQM system in Run2. The main technical challenges which we have encountered and the solutions adopted during Run2 will also be discussed, including efficient use of memory in multithreading environments. Finally, we present the prospect of a future DQM upgrade with emphasis on functionality and long-term robustness for LHC Run3.

1. Introduction

In the CMS experiment, the Data Quality Monitoring group (*DQM*) serves three related, but subtly different purposes: (i) it provides *monitoring*, that means it raises alarms when some part of the detector malfunctions; (ii) it provides *data certification*, that means it records for which fractions of taken data parts of the detector were faulty; (iii) it aids *debugging*, by providing detailed information that helps experts to find the cause of spotted issues. The output for each of these tasks is very different: for the first two, it can be reduced to a single bit indicating the data quality as "good" or "bad". The difference between monitoring and certification is that for monitoring the results have to be available in real time, while some false positive and even

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

false negative rate can be tolerated. Instead for certification, the error rate should be as low as possible, while a delay of several days is acceptable. In contrast, as a debugging tool, the output of DQM consists of a large amount of plots called *monitor elements*. Usually, these are histograms in ROOT [1] format of some quantity derived from event data for a subpartition of the detector and some window of time. Detector experts can browse these monitor elements to understand what is happening in the detector.

The DQM group cooperates with each of the CMS *subsystems* to collect data about the subsystem and display it in a useful way. Subsystems are usually the different detectors in CMS, but also groups that are not directly represented in the CMS detector, such as the trigger systems and also DQM itself.

The DQM system consists of modules in the CMS data processing software [2] as well as independent applications, which are run on hardware infrastructure dedicated to DQM. This software can roughly be decomposed into a data collection step, which consumes CMS event data and produces monitor elements, which are no longer connected to individual events; a visualization part, which makes the monitor elements browsable in a convenient form; and tools to handle metadata, including the final certification results. Each of these parts also has a *user interface*: The data collection software is *provided* by subsystem experts, which are users of DQM; therefore the APIs in this part of the DQM software are a part of the user interface. The visualization of monitor elements is part of the user interface provided for debugging, but also used internally by the monitoring and certification team. This team then uses tools to record their findings, which are presented to the collaboration on another set of interfaces and user-facing APIs. This user interaction turns out to be one of the biggest challenges in DQM. We will take a look at how these systems are currently implemented in CMS in section 2.

So far, DQM has performed well in LHC Run2 (2015-2018) and always reliably performed its tasks; we will have a closer look at the past performance in section 3. However, achieving this required handling some major challenges. We will go into detail of how we solved one of these, the high memory consumption of DQM in multithreaded reconstruction, in section 4.

Finally, to ensure smooth operation in Run3 (starting 2021) and beyond, some major revisions of the DQM system and procedures have to be considered. We will talk about plans for future-proofing the DQM system in section 5.

2. The DQM system

The DQM system spans the entire CMS readout and reconstruction chain. Figure 1 shows a schematic overview of most of the components of DQM, with the software components of the DQM system highlighted in gray. DQM code runs as part of CMSSW for most places where CMSSW is used: in the High Level Trigger (DQM at HLT), in the dedicated online DQM machines, in the data reconstruction jobs in the CMS computing centers, and in the simulation and reconstruction jobs used for release validation and Monte-Carlo production. The monitor elements produced by all these jobs are aggregated in the *DQMGUI*s, which archive the data and provide a web-based interface to browse and view the DQM output. The DQMGUIs are then queried by the DQM shifters, certification and validation teams and subdetector experts, sometimes using additional tools. One such tool is *historic DQM* (HDQM) which provides a more time-based view of DQM data to spot trends over longer periods of time. Finally, the decisions made by the certification team are collected in the *Run Registry*, which then provides the list of data that is suitable for physics analysis to the collaboration. Usually, this takes the form of a *golden JSON file*, which lists all *good* runs with data quality certified at a granularity of ~ 23 seconds (CMS *Lumisections*).

In this section, we will discuss each of the software components in DQM.

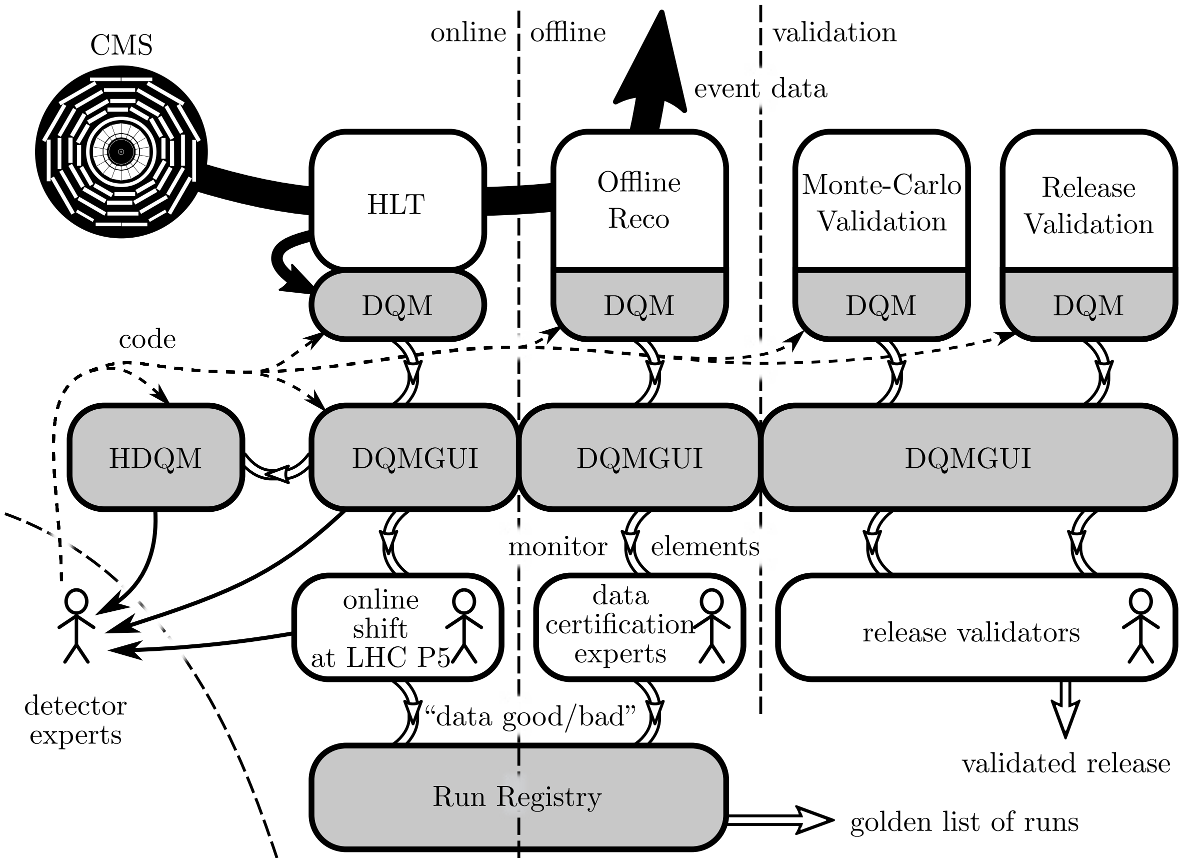

Figure 1. Schematic overview of the DQM systems.

2.1. DQM as part of CMSSW

CMS event data is processed by *CMSSW* [2], using the CMSSW event data model (*EDM*) framework specifically built for this purpose. This EDM framework allows to decompose the processing into small plugins, which are connected into *sequences* that event data is piped through.

The DQM plugins are developed by subsystem experts, using a framework provided by DQM. This *DQM framework* [3] touches some corner cases of the EDM framework, since it consumes event data but stores and outputs non-event data.

Since the functionality of the DQM framework within CMSSW to produce and save non-event data is also useful for other applications, there are a few plugins which use DQM infrastructure, even though they are not within the scope of DQM as outlined in the introduction. Primarily such plugins are used by the Alignment and Calibration group to save data required to *update* the detector conditions, while *providing* the detector conditions to all CMSSW plugins is handled by CMSSW EDM directly.

2.2. The online DQM system

As special source of monitor elements for DQM is the *online DQM system* [3], a number of machines running at the experiment site at LHC Point 5 dedicated to run DQM jobs. The online machines are a small fraction of the HLT farm, and managed in the same way. The input data stream is produced by HLT specifically for DQM. The main DQM stream is a mixture of physics data at a rate of 50-100Hz. For event displays, some events are fully reconstructed and then shown on screens in the control room, this reconstruction is also handled in the online DQM system. A number of additional special-purpose streams are received and consumed by online DQM: some DQM code runs in the HLT jobs themselves, their output is transferred to and unpacked by the online machines. Another stream is dedicated to a beamspot reconstruction run at online DQM, which feeds back into HLT. This is not a typical DQM responsibility, but the same infrastructure is reused efficiently.

The CMSSW jobs run in the online system using completely free-form, stand-alone configuration files maintained by the respective subsystem experts, which provides a lot of flexibility to use online DQM for any sort of online monitoring. Usually, a client reads *raw* data from the DQM stream, performs local reconstruction for one subdetector, and then runs DQM on the produced quantities. To allow this flexibility, the online system is designed to be robust and keeps the jobs independent, such that one failing configuration does not affect the others.

The production online system is complemented by a *playback* [4] system, which is an accurate replica of the main system that is fed from stored data. It is used to validate new developments before they are put into production. The playback system plays an important role in allowing much faster validation of changes compared to CMSSW itself. This allows subsystem developers to add ad-hoc changes to online DQM as needed.

The playback as well as the production system is monitored by a dedicated monitoring application called *DQM squared*, which is completely independent of the DQM and DAQ software on the online DQM machines. It allows watching the flow of event data through the system as well as the health and logging output of the subsystem DQM processes. This tool is intentionally kept simple, to allow troubleshooting problems that could appear at any step of the online DQM processing. All information in *DQM squared* is archived, to also allow debugging transient issues after they disappear.

2.3. Offline DQM and Release Validation

In the CMS data processing jobs, the CMSSW parts of DQM run to monitor various quantities. This includes the input raw data, which is only monitored with full statistics in offline processing, temporary reconstruction quantities that are never saved to files, and the output quantities. Apart from the offline processing of recorded experiment data, there are also jobs that are used for validation: Release Validation runs a representative set of workflows on old data to check for software changes and regressions, which are spotted by comparing DQM outputs. In Monte-Carlo validation, the software is run on simulated events where additional quantities known from the simulation are added to the events, such as the "ground-truth" tracks of the simulated particles. The Validation DQM running in these jobs can perform comparisons between this ground-truth data and the quantities reconstructed from the simulated raw data to assess the performance of the reconstruction algorithms implemented in the software.

The *DQM sequence* for each of these applications is different, and depends on the specific type of data to be processed. Since many subdetector quantities can only be monitored properly in few of the datasets, it would be useless and wasteful to produce the respective DQM output on the other datasets. To describe this, we define the *DQM matrix*, which defines which plugins or sub-sequences should run for which CMS dataset. However, in practice the DQM matrices for different types of jobs are not unified and spread over various configuration files. Maintaining the DQM sequences and keeping track of which sequences are used for which purposes turned out to be a very difficult task and is a major challenge of this software component right now. Currently, only prompt offline reconstruction uses a well-maintained matrix, while other jobs (express offline reconstruction running on a small fraction of events, re-reconstructions running months later with improved calibrations) use a dataset-invariant DQM sequence to avoid the complexity of maintaining a DQM matrix. The motivation to have a dataset-dependent configuration for prompt reconstruction, while other jobs can use a static sequence, is resource pressure, primarily

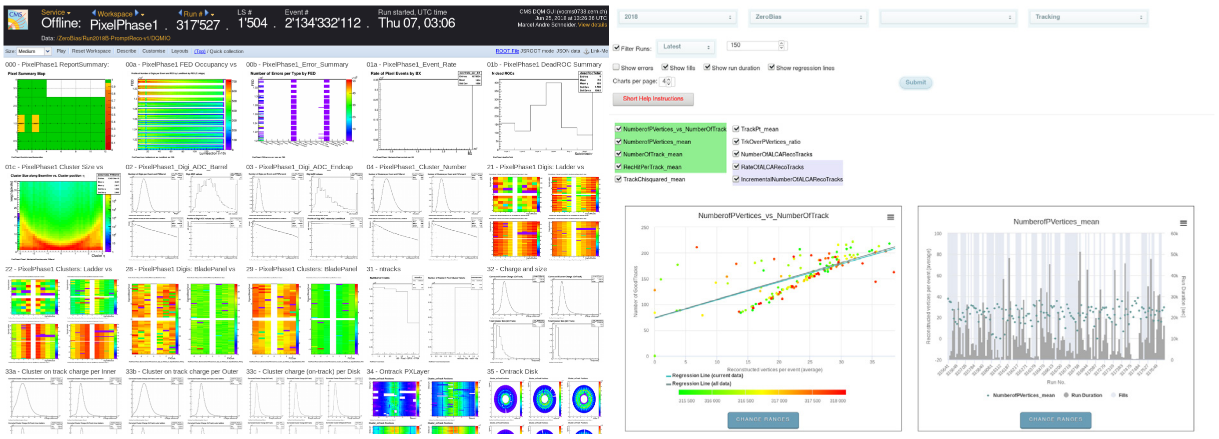
Offline: PixelPhase1 . 317'527 . 1'504 . 2'134'332'112 . Thu 07, 03:06
NumberofPVertices_vs_NumberOfTrack
NumberofPVertices_mean
CHANGE RANGES

Figure 2. The user interface of DQMGUI (left) and historic DQM (right).

regarding memory.

2.4. The DQM GUI

The monitor elements produced by CMSSW jobs are uploaded to the DQMGUI [5] for browsing and distribution. There are three main instances of the DQMGUI software, each running on dedicated hardware: (i) the *online DQMGUI*, which displays data from the online system and is physically located in the datacenter at LHC Point 5; (ii) the *offline DQMGUI*, which holds the DQM output from offline reconstruction and has the largest volume of data; (iii) the *RelVal DQMGUI*, which holds the DQM output for release validation.

Each DQMGUI consists of a web server hosting the web-based user interface (fig. 2), HTTP-based APIs to access DQM data, and a custom-built database called the *index*, which holds the actual monitor element data and can be queried via the APIs. Data enters the DQMGUIs as ROOT files containing the monitor elements. As a special case, the online DQMGUI also receives monitor elements from the online system over a custom network protocol: these *live mode* monitor elements are immediately shown in the web interface.

The detector experts contribute code to the DQMGUIs in the form of *render plugins*, which determine how monitor elements should be displayed, as well as *layouts*, which provide useful overviews tailored to specific use cases.

2.5. Historic DQM

As originally designed, DQM only shows aggregated information about blocks of data, usually *runs*. It was not designed to display time-series data or trends. This makes sense for the monitoring and certification use cases of DQM: only the *current* state of the detector matters (monitoring), or the quality of data in the Run to be certified. However, for the debugging use case of DQM, some subsystem experts wished to also see trends over time.

This was first covered by custom tools, which retrieve monitor elements from the DQMGUI and arrange the data ordered by time. Recently, one such tool was turned into the central *historic DQM* (HDQM) application, a webservice running as part of the DQM infrastructure (fig. 2). One reason for the rather slow adoption are the additional complexities in such a tool: many quantities are strongly influenced by detector settings and LHC conditions. Therefore, HDQM needs to import additional information, for example about LHC fills, and filters Runs to only include values that are comparable.

2.6. Run Registry
While most of the DQM software is concerned with handling event data and monitor elements, the role of *Run Registry* [6] is to collect meta data about the data taken by CMS, including the *certification flags* decided by the shift team and experts. Run Registry is implemented as a database-based web service, part of the central CMS ORACLE database and largely independent of the other DQM tools. It imports information from other CMS services (mostly *web-based monitoring, WBM* [7], but also the *CMS data catalog, DBS* [8] and the DQMGUIs), but it is primarily a tool where information is entered via user interfaces. Its data is then read via an API to produce the final certification output, the *golden JSON file* specifying good data suitable for physics analysis [9]. These JSON files are archived on the AFS file system at CERN and can be retrieved from there to do physics analysis.

In general, all certification decisions are taken by users. In a few cases, for example when detectors are turned off, the data is not presented for certification, but even these decisions can be overridden manually.

3. DQM performance
The DQM system has performed reliably over Run2 [10]. In some cases, manual interventions were required to ensure smooth operations; these are the cases where we want to improve things for Run3. Common problmes included incorrect configuration for DQM sequences, leading to software crashes in offline reconstruction jobs or useless or missing DQM outputs. Other issues included service instabilities, usually caused by overload from automated downstream tools, and problems caused by version incompatibilites in ROOT.

The DQMGUIs hold DQM data for almost all data recorded by CMS, all available in the web frontend. The database holding those histograms grew to a staggering size over the years, which the software handles without problems. The offline DQMGUI holds DQM results for more than 400,000 Runs and datasets, covering $3.8 \cdot 10^{10}$ CMS events processed by offline DQM (including events processed multiple times) in $4.4 \cdot 10^{10}$ monitor elements. These are stored in a 4.1TB database on a single machine. The online DQMGUI, which only orders data by runs, holds more than 22,000 runs in a 650GB database.

Of course, the DQM software is not static over this time. In the last year (July 2017 – July 2018), CMSSW received about 600 pull requests related to DQM (out of about 2000 total), more than any other CMSSW component. These changes are complemented by about 100 changes to DQMGUI layouts and render plugins over the same window of time. The flexibility to constantly adapt is crucial to make DQM a useful service for the detector teams.

4. The memory challenge
Since DQM is an easy way to make monitoring information available for large amounts of data, it is tempting for subsystems to add lots of monitor elements proactively, to have useful information available in case it is needed. This leads to an accumulation of monitor elements in the DQM system, which has an impact on the resource consumption of DQM. In general, this is not a problem: this is what DQM is designed for, and all components are designed to be efficient enough to handle large amounts of monitor elements.

However, in 2017 this led to a problem with the high memory consumption of DQM in prompt reconstruction. A big part of the problem there is the use of multi-threading. There are three possible approaches to make DQM work in a multi-threaded environment: (i) ensure monitor elements are only accessed by a single thread; (ii) make access to monitor elements thread safe, using atomic updates or fine-grained locking; (iii) exploit the fact that most monitor elements (typically histograms) provide an associative reduction operation.

The multithreaded DQM design from the beginning of Run2 used the third approach [3]: as the incoming data is split into multiple parallel *streams*, the monitor elements are duplicated into one instance per stream, which can be reduced into the final result by summation as needed. But, as the number of monitor elements grows, there are two problems with this approach: first, the memory consumption increases linearly with the number of threads, which partially negates the point of using multi-threading compared to running independent jobs concurrently. Second, the computing time required to merge the split histograms into the final result can become significant, especially when there are many monitor elements that need to be merged but only few events processed in between merges. This situation appeared for example in the HLT, which is monitored on a granularity of CMS lumisections, a period of 23 seconds.

When the memory use became a problem in 2017, some DQM plugins were changed to use monitor elements that allow concurrent access, the second approach. This avoids the need to keep per-thread instances of each monitor element in memory. However, the respective DQM code needed invasive changes to support the new design. In 2018, the first approach was chosen for the majority of DQM code. This might seem surprising, since essentially it means not using multi-threading at all. However, different types of DQM plugins that use distinct monitor elements can still run in parallel, with exclusive access to their monitor elements and without the need for per-thread copies. The changes to the DQM code base are minor and transparent for most plugins.

5. Future plans

As mentioned in the previous section, the DQM system is working well and there is not much need for structural changes for the future, but we do plan a number of improvements in details.

One long-standing issue that frequently requires manual work is the CMSSW configuration for DQM. Over time the DQM sequences were extended and new sequences added, without keeping track of the purpose and interdependencies of these modules. This makes many changes dangerous or even impossible today, since it is no longer clear which down-stream consumers will be affected by changes to the DQM sequences. Since some DQM infrastructure is shared with Alignment and Calibration, this can have far-reaching side effects. Mapping out all the use cases and consumers of DQM, and refactoring the configuration to clearly reflect the use cases, is a long-term project.

Another maintainability aspect that is currently being worked on is the certification system built around Run Registry, which itself poses severe maintainability problems. This is currently addressed by a redesign of Run Registry, along with plans to also include various stand-alone tools used for data certification into it, for example for the creation and distribution of the golden JSON files. A big push for this refactoring is the integration of machine learning based tools in certification [11, 12, 13], which require more elaborate interfaces and changes in the certification workflow in general. The new Run Registry will be based on a PostgreSQL database independent from other CMS services, using an append-only schema to ensure traceability.

In general, many auxiliary tools grew around the DQM system, using various intentional and ad-hoc APIs to DQMGUI and Run Registry. These need to be improved and extended. An idea there is to not only provide APIs, but also an environment to run custom tools; this could be based on Jupyter/SWAN to simplify development and sharing of tools.

Finally, we need to make sure that the entire DQM infrastructure can scale to the increasing volume of data to be handled. We want to provide equally good or better monitoring in Run3, for the old and the upgraded detectors, while compatibility with and accessibility of the existing DQM data are a big concern. The current infrastructure is in a good shape and no major changes are expected, however a redesign of the DQMGUI to handle the ever growing volume of data and more flexible time granularity might be required.

6. Conclusion

The DQM systems have, after their big redesign for Run2 [10, 3], performed well, but are not without flaws. The continuous evolution of the DQM software under time pressure during data taking caused a significant amount of technical debt, that we want to remove during the LHC long shutdown (2019-2020). The main domains are the software configuration, which grew in an uncontrolled way due to continuous adaptions for changing conditions, the data certification system, which unfortunately was built mostly on ad-hoc scripts, and the DQMGUI environment, which works very well but is also very technically complicated and hard to understand and maintain. For Run3, we are working towards incremental improvements on all components, mainly to improve maintainability, but also to make DQM more efficient and ready for the large amounts of accumulated historical data as well as the larger amounts of data expected in the future.

References

[1] Brun R and Rademakers F 1997 ROOT—an object oriented data analysis framework *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* vol 389 (Elsevier) pp 81–86

[2] CMS Collaboration 2006 CMS physics TDR: volume I, detector performance and software Tech. Rep. CERN-LHCC-2006-001 CERN-LHCC

[3] Rovere M 2015 The Data Quality Monitoring software for the CMS experiment at the LHC *Journal of Physics: Conference Series* vol 664 (IOP Publishing) p 072039

[4] Tuura L, Meyer A, Segoni I and Della Ricca G 2010 CMS data quality monitoring: systems and experiences *Journal of Physics: Conference Series* vol 219 (IOP Publishing) p 072020

[5] Tuura L, Eulisse G and Meyer A 2010 CMS data quality monitoring web service *Journal of Physics: Conference Series* vol 219 (IOP Publishing) p 072055

[6] Rapsevicius V, CMS DQM *et al.* 2011 CMS Run Registry: Data certification bookkeeping and publication system *Journal of Physics: Conference Series* vol 331 (IOP Publishing) p 042038

[7] Lopez-Perez J A, Badgett W, Behrens U, Chakaberia I, Jo Y, Maeshima K, Maruyama S, Patrick J, Rapsevicius V, Soha A *et al.* 2017 The web based monitoring project at the CMS experiment *Journal of Physics: Conference Series* vol 898 (IOP Publishing) p 092040

[8] Afaq A, Dolgert A, Guo Y, Jones C, Kosyakov S, Kuznetsov V, Lueking L, Riley D and Sekhri V 2008 The CMS dataset bookkeeping service *Journal of Physics: Conference Series* vol 119 (IOP Publishing) p 072001

[9] Batinkov A, Rovere M, Borrello L, De Guio F, Duggan D and Di Guida S 2013 The CMS Data Quality Monitoring Software: Experience and future improvements *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2013 IEEE* (IEEE) pp 1–5

[10] De Guio F 2015 The data quality monitoring challenge at CMS experience from first collisions and future plans Tech. Rep. CMS-CR-2015-329 CMS

[11] Pol A A 2018 Anomaly detection using deep autoencoders for the assessment of the quality of the data acquired by the CMS experiment Proceedings of CHEP2018, Sofia

[12] Pol A A 2018 Online detector monitoring using AI: challenges, prototypes and performance evaluation for automation of online quality monitoring of the CMS experiment exploiting machine learning algorithms Proceedings of CHEP2018, Sofia

[13] Azzolini V 2018 Improving the use of data quality metadata via a partnership of technologies and resources between the CMS experiment at CERN and industry Proceedings of CHEP2018, Sofia