Operational experience with the new CMS DAQ-Expert

Jean-Marc Andre[8], *Ulf* Behrens[4], *James* Branson[1], *Philipp* Brummer[3,11], *Sergio* Cittolin[1], *Diego* da Silva Gomes[3] , *Georgiana-Lavinia* Darlea[6], *Christian* Deldicque[3], *Zeynep* Demiragli[6], *Marc* Dobson[3], *Nicolas* Doualot[8], *Samim* Erhan[5], *Jonathan Richard* Fulcher[3], *Dominique* Gigi[3], *Maciej* Gladki[3], *Frank* Glege[3], *Guillelmo* Gomez-Ceballos[6], *Jeroen* Hegeman[3], *Andre* Holzner[1], *Michael* Lettrich[3], *Audrius* Mecionis[8,9], *Frans* Meijers[3], *Emilio* Meschi[3], *Remigius K.* Mommsen[8], *Srecko* Morovic[8], *Vivian* O'Dell[8], *Luciano* Orsini[3], *Ioannis* Papakrivopoulos[7], *Christoph* Paus[6], *Andrea* Petrucci[2], *Marco* Pieri[1], *Dinyar* Rabady[3], *Attila* Racz[3], *Valdas* Rapsevicius[8,9], *Thomas* Reis[3], *Hannes* Sakulin[3,*], *Christoph* Schwick[3], *Dainius* Simelevicius[3,9], *Mantas* Stankevicius[8,9], *Cristina* Vazquez Velez[3], *Michail* Vougioukas[3], *Christian* Wernet[3], and *Petr* Zejdl[8,10]

[1]University of California San Diego, San Diego, USA
[2]Rice University, Houston, USA
[3]CERN, Geneva, Switzerland
[4]Deutsches Elektronen-Synchrotron, Hamburg, Germany
[5]University of California Los Angeles, Los Angeles, USA
[6]Massachusetts Institute of Technology, Cambridge, USA
[7]National Technical University of Athens, Athens, Greece
[8]Fermi National Accelerator Laboratory, Batavia, USA
[9]Also at Vilnius University, Vilnius, Lithuania
[10]Also at CERN, Geneva, Switzerland
[11]Also at Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract. The data acquisition (DAQ) system of the Compact Muon Solenoid (CMS) at CERN reads out the detector at the level-1 trigger accept rate of 100 kHz, assembles events with a bandwidth of 200 GB/s, provides these events to the high level-trigger running on a farm of about 30k cores and records the accepted events. Comprising custom-built and cutting edge commercial hardware and several 1000 instances of software applications, the DAQ system is complex in itself and failures cannot be completely excluded. Moreover, problems in the readout of the detectors, in the first level trigger system or in the high level trigger may provoke anomalous behaviour of the DAQ system which sometimes cannot easily be differentiated from a problem in the DAQ system itself. In order to achieve high data taking efficiency with operators from the entire collaboration and without relying too heavily on the on-call experts, an expert system, the DAQ-Expert, has been developed that can pinpoint the source of most failures and give advice to the shift crew on how to recover

* Corresponding author: Hannes.Sakulin@cern.ch

in the quickest way. The DAQ-Expert constantly analyzes monitoring data from the DAQ system and the high level trigger by making use of logic modules written in Java that encapsulate the expert knowledge about potential operational problems. The results of the reasoning are presented to the operator in a web-based dashboard, may trigger sound alerts in the control room and are archived for post-mortem analysis - presented in a web-based timeline browser. We present the design of the DAQ-Expert and report on the operational experience since 2017, when it was first put into production.

1 Introduction

The Large Hadron Collider (LHC) at CERN typically provides collisions of protons or heavy nuclei around the clock on about 200 days per year. In order to obtain the statistics necessary for precision physics analyses, the experiments have to continuously record these collisions with as little loss as possible. The Compact Muon Solenoid (CMS) experiment [1,2] is operated by a crew of physicists from the CMS collaboration who typically dedicate only a few weeks per year to operating the experiment while doing other tasks during the rest of the year. Five operators are needed to operate the experiment. They perform their duties during 8-hour shifts. A crew of on-call experts (who typically do shifts of half a week to a week) supports the operators. In this scheme of operations it is crucial to automate data taking as much as possible and to design tools that can help the shift crew diagnose and resolve problems swiftly and without relying on the on-call experts too heavily.

The CMS Run Control and Monitoring System (RCMS) [3,4,5,6], includes automation at various levels [7,8]. Cross-checks ensure that all configuration changes are applied to relevant components of the system and in the correct order. At the start and end of an LHC fill when high voltages are ramped up and down, necessary actions in the data acquisition such as the disabling or enabling of payload suppression are automatically triggered, so that operators can start a run long before stable beams are declared. The over-all configuration of all components is determined by a top-level *Run Mode* that is normally chosen automatically based on the state of the LHC. Recovery of frequent and well-known problems local to a single sub-system - such as single-event upsets in the electronics - has been automated using the so-called soft error recovery mechanism, which dramatically decreased down times since the end of Run-1 of the LHC [7] (46 hours of down time avoided in 2012 alone). More recently, another layer, the Level-0 Automator [8] has been added: it allows operators to perform complex recovery actions with a single command. It commands sub-systems in the most streamlined sequence and automatically attempts to recover from additional problems occurring during the recovery.

There remain certain types of data taking problems that cannot be recovered by the above automatisms because they affect multiple sub-systems or because recovery requires expert intervention. Most of these problems can be diagnosed to a certain level by examining the monitoring information from a few central systems: the Central Data Acquisition system (CDAQ), the Timing and Control Distribution System (TCDS) [9] and the high-level trigger running on the file-based filter farm (F3) [10]. Domain specific human experts can pinpoint the origin of the problem by looking at the available monitoring clients that display the monitoring information. However, for the operators it can be difficult to interpret the information and draw conclusions, especially when under time pressure. For some problems it is difficult even to find out whether it is caused by a subsystem, sending data to the CDAQ, by the CDAQ itself or by a system down-stream of the DAQ such as the F3 or the storage and transfer systems. An expert system that

continuously examines the monitoring data can help to quickly pinpoint the problem. It can either give recovery instructions to the CDAQ / run control operator or determine what on-call expert needs to be called to solve the problem. During Run-1 of the LHC the Perl-based DAQ Doctor [7] fulfilled this task. It gave advice to the shift crew in a text-based terminal window. Due to the introduction of the TCDS system, the new DAQ-2 [11] system and the file based filter farm for Run-2 of the LHC, major changes to the expert tool were necessary. A new tool, the DAQ-Expert [12,13] was developed. We report on this new tool and on operational experience in the present paper.

2 The DAQ-Expert tool

As in its predecessor, reasoning logic in the new DAQ-Expert is implemented in a procedural language. This approach was chosen since deep knowledge about procedural languages is more widespread in the CDAQ group compared to declarative languages. Java was chosen for the DAQ-Expert tool since Java is already used in the Run control system and other tools developed by the group. Figure 1 shows the main components of the new DAQ-Expert tool, explained in detail in the following sub-sections.

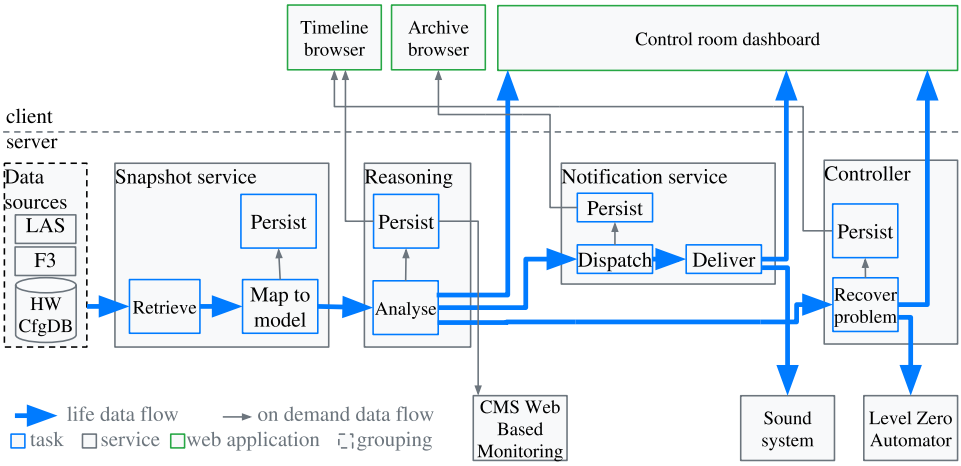

Fig. 1. Main components of the new DAQ-Expert tool. LAS: Live Access Server of the XDAQ Monitoring and Alarming System (XMAS [14]). F3: elastic-search based monitoring of the File-based Filter Farm. HWCfgDB: CMS DAQ Hardware and Configuration Database. The server side consists of four micro-services: Snapshot Service, Reasoning Service, Notification Service, Controller.

2.1 The Snapshot Service

The snapshot service continuously retrieves monitoring data form the monitoring systems of the CDAQ, TCDS and F3 and maps these data to the current DAQ system configuration, retrieved from the CMS DAQ Hardware and Configuration database [15]. A snapshot of all relevant monitoring data, including the DAQ system structure, is produced every 2-3 seconds and persisted in JSON format on an NFS filer. The typical snapshot size is 1.5 MB (or 100 kB after compression). A snapshot contains the state of the system at the time it is taken including active error messages. The produced snapshots are used by the Reasoning Service of the DAQ-Expert and by the DAQView React monitoring client. The snapshot service was introduced to homogenize monitoring information coming from

different monitoring systems, to provide a common persistency strategy and to include the DAQ system structure so that subsequent tools do not need to access the DAQ system configuration database.

2.2 The Reasoning Service

The Reasoning Service executes the reasoning logic, encapsulated in Logic Modules (LMs) written in procedural language. Each LM defines a condition that may either be satisfied or not. Optionally a LM may provide context information that characterizes the detected problem in more detail. LMs may use all monitoring information of a snapshot and the result of other LMs to determine whether their condition is satisfied. They may also internally store information from previous snapshots. Lower level LMs define simple conditions like "Run Is Ongoing" or "LHC is Providing Stable Beams". Higher level LMs detecting an error condition provide a description of the problem that can be parameterized with actual values from the problem analysis, and actions to be taken to recover from the problem. Logic Modules declare which other LMs they require information from. Based on these declarations the reasoning service determines the order of execution. The result of the reasoning logic is persisted in a relational database and displayed in the web-based timeline browser (Figure 2). The reasoning service dispatches events to the Notification Service: these can be single events like the LHC entering a new beam mode, or the start, the end or an update of a condition (i.e. the result of a logic module).

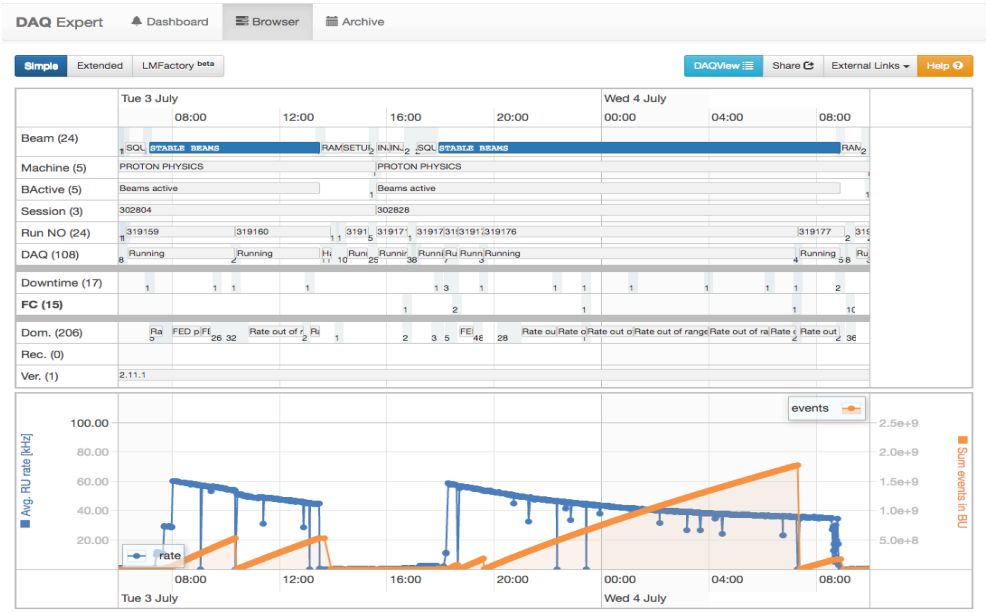

Fig. 2. The web-based timeline browser. The top frame shows states of the LHC, and the DAQ system. The middle frame shows the conditions of the reasoning logic. The bottom frame shows the history of the trigger rate and number of events collected in a run. The user may freely zoom from showing periods of months down to seconds. The information is downscaled to the appropriate resolution at the server.

2.2.1 Determining the root cause

Some conditions may be either the root cause of a problem, or – in the presence of another condition – the symptom of the other condition. In order to determine the root

cause, LMs declare their causality relationship with other conditions. From these declarations the reasoning service builds a causality graph as shown in Figure 3, which is used to determine the root cause. As an example, the following conditions may be satisfied (from the more inclusive to the less inclusive): *Deadtime, CriticalDeadtime, TTSDeadtime, PartitionDeadtime, FEDDeadtime FEDGeneratesDeadtime*. All these conditions are linked in the causality graph. The leftmost condition *FEDGeneratesDeadtime* would in this case be found as the root cause and presented to the operator meaning that a Front-End Driver (FED), i.e. an input to the Central DAQ, is generating dead time above a certain threshold. The problem would in this case need to be further investigated by a subsystem expert of the subsystem containing this FED. In a different situation, all the previously mentioned conditions may be satisfied but in addition also *RateTooHigh* and *VeryHighTcdsInputRate*, meaning that the first-level trigger is generating an input rate far above the expected rate. In this case *VeryHighTcdsInputRate* would be considered the root cause and the dead time generated by the FED a symptom. In this case a first-level trigger expert would need to investigate the high input rate.

The causality graph may also indicate multiple root causes for a given problem. For example dead time may at the same time be caused by a FED and by backpressure from the filter farm / high-level trigger (HLT).

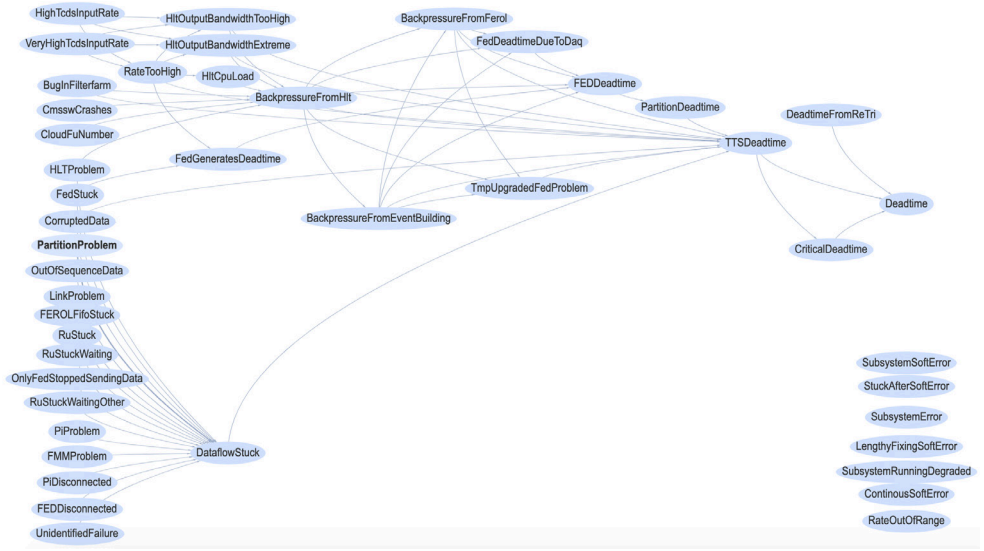

Fig. 3. Causality graph of the Logic Modules (partial). Conditions on the right hand side may be the symptom of conditions on the left hand side.

2.3 The Notification Service

The Notification Service receives events from the Reasoning Service and some external sources. It dispatches these events to the web based Dashboard (Figure 4), to the sound system, and optionally by e-mail and SMS. It also archives all events and makes the archive searchable through a web interface.

2.4 The Recovery Controller

The latest addition to the DAQ-Expert is the Recovery Controller that can trigger recovery actions by sending commands to the top-level control node of the run control

system or to the Level-0 Automator. It can be used to recover from a number of problems that prescribe a simple recovery action or a number of actions to be tried in sequence. Typically these actions include issuing a Hard Reset command via the TCDS system; stopping and starting the run; stopping the run, re-configuring or re-initializing a subsystem and re-starting the run. The recovery actions are defined in a custom mark-up language embedded in the list of actions suggested by the logic module. These actions can be manually executed by a command on the top-level run control node or by a command on the Automator tool. Via the Recovery Controller they can be triggered directly from the web-based Dashboard of the DAQ-Expert tool by clicking a button next to the prescribed recovery action ("Execute Step" button in Figure 4). Full automation of issuing these recovery steps for certain types of problems is under way.

2.5 Technologies used

The server side of all the above components including the Logic Modules is written in Java and running in an Apache Tomcat [16] server implementing a micro-service architecture with RESTful web services. Hibernate [17] is used for persistency to an Oracle database. De-/serialization from/to JSON is handled by Jackson [18].

The client side running in the web browser is implemented in Javascript using web sockets, Bootstrap [19], ReactJS [20] (for the dashboard) and vis.js [21] (for the timeline browser).

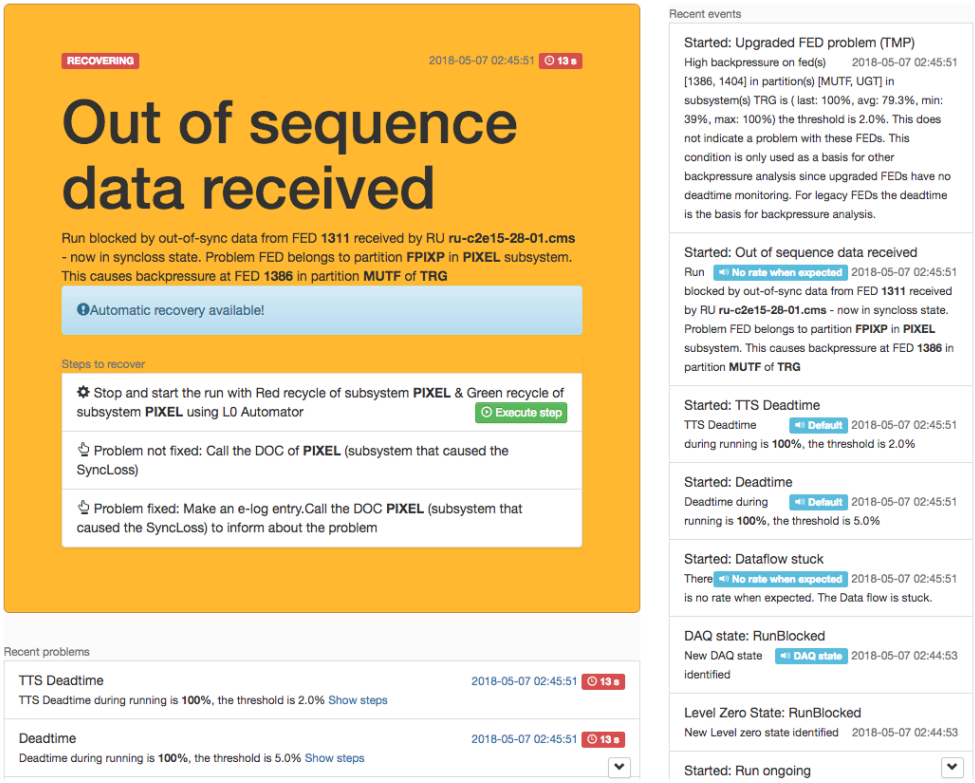

Fig. 4. The web-based Dashboard shows the current main problem with steps to follow for recovery, concurrent problems and the history of problems (left frame). It also shows the history of all notifications received and whether a sound alert was triggered (right frame).

3 Operational experience

The new DAQ-Expert tool has been gradually introduced since mid-2016. Coverage for problems causing down time reached 78 % in mid-2017 and 95 % in mid-2018. Even though the tool is called the DAQ-Expert and draws monitoring data mostly from the central DAQ and other central systems, the vast majority of down times are actually caused by the sub-systems. However, these sub-system problems manifest themselves in the central DAQ system and – before the introduction of the new DAQ-Expert - were sometimes difficult to discern from genuine problems of the CDAQ, especially for inexperienced operators. Table 1 shows the luminosity lost by down times caused by all online systems and caused by the CDAQ only from 2015 to 2018. There is a clear trend towards better efficiency by luminosity. We believe that the DAQ-Expert significantly contributed to this improvement. However, since the on-line systems of CMS as well as the experience of the shift crew are evolving it is not possible to quantify this effect.

Another interesting measure is the requirement for help form the central DAQ on-call expert. Before the introduction of the DAQ-Expert tool, the on-call expert frequently needed to help interpret the monitoring data to find out the root cause of a problem while this is now handled by the DAQ-Expert in 95 % of the cases. Figure 5 shows the number of the calls to the CDAQ on-call expert from the control room of CMS over time. In order to measure only calls related to data-taking problems, only night-time calls between 23h and 8h are considered. A clear trend towards fewer calls is visible. While part of this result may be due to other improvements in on-line systems, we believe that the DAQ-Expert tool significantly helped to reduce the load on the on-call expert.

Table 1. Luminosity lost in down time events from 2015 to 2018 for down-time events caused by any online system (including CDAQ) and down-time events caused by CDAQ. Periods with zero trigger rate for > 30 seconds are considered as down time. Shorter interruptions of data-taking are counted as dead time. Luminosity lost due to infrastructure problems, tests, commissioning and due to dead time is not shown. The latter amounts to a similar amount as the luminosity lost due to down time.

	Luminosity lost due to online systems (L1 trigger, sub-system DAQ, CDAQ)	**Luminosity lost due to CDAQ**
2015, no DAQ-Expert	3.7 %	0.20 %
2016, DAQ-Expert beta (since August)	2.6 %	0.08 %
2017 DAQ-Expert	2.4 %	0.02 %
2018 DAQ-Expert (up to July 6)	1.8 %	0.03 %

Conclusion

The new DAQ-Expert tool for the CMS DAQ has been presented. It detects the root cause of data-taking problems with 95 % coverage, greatly simplifying the task of the operator. It reduces the load on the human on-call experts and contributes to an improved over-all data taking efficiency.

We would like to acknowledge the help of our colleagues from the CERN IT/CS group who provided the raw data about the history of calls used to produce Figure 5.

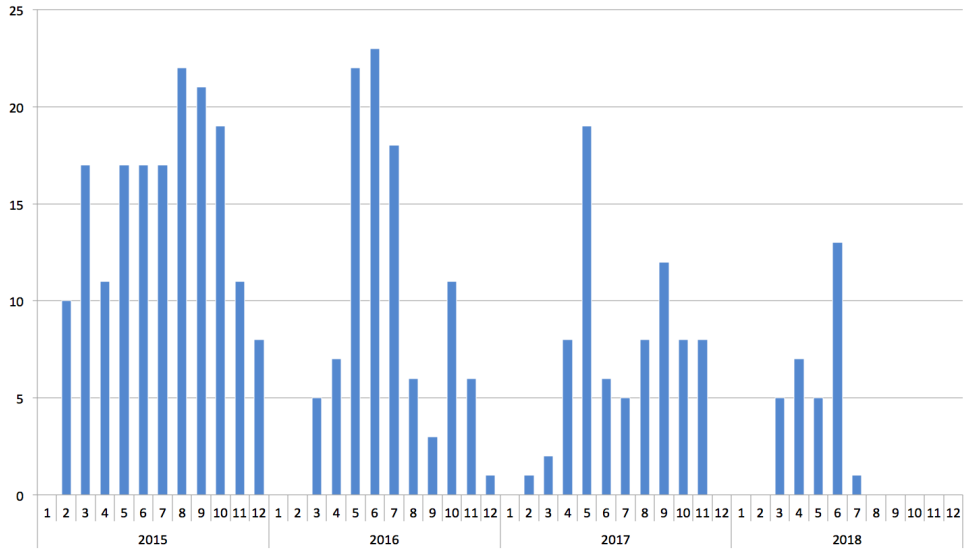

Fig. 5. Number of night-time calls to the CDAQ on-call expert per month from 2015 to 2018. Data provided by the CERN IT/CS group.

References

1. The CMS Collaboration, *CMS Technical Proposal*, CERN LHCC 94-38 (1994)
2. The CMS Collaboration, JINST 3 S08004 p. 361(2008)
3. M. Gulmini *et al*., arXiv:cs/**0306110** (2003)
4. A. Oh *et al*., J. Phys.: Conf. Ser. **119** 022010 (2008)
5. A. Petrucci *et al*., PoS ACAT **026** (2007)
6. H. Sakulin *et al*., IEEE Trans. Nucl. Sci. **59** 4 1597-1604 (2012)
7. H. Sakulin *et al*., J. Phys.: Conf. Ser. **513** 012031 (2014)
8. H. Sakulin *et al*. J. Phys. : Conf. Ser. **898** no.3, 032028 (2017)
9. J. Hegeman *et al*., 10.1109/NSSMIC.2015.7581984 (2015)
10. S. Morovic *et al*., J. Phys. : Conf. Ser. **664** 082036 (2015)
11. H. Sakulin *et al*., IEEE Trans. Nucl. Sci. **62** no.3, 1099-1103 (2015)
12. M. Gladki *et al*. J. Phys. : Conf. Ser. **1085** no.3, 032021 (2018)
13. CMS DAQ group, "DAQExpert" [software] version 2.15.3, available from http://daq-expert.cern.ch [Accessed 2018-10-30]
14. L. Orsini *et al*., J. Phys. : Conf. Ser. **219** 022042 (2010)
15. H. Sakulin *et al*., J. Phys.: Conf. Ser. **219** 022003 (2010)
16. The Apache Software Foundation, "Tomcat" [software] version 8.0.36 available from http://tomcat.apache.org/ [Accessed 2018-10-30]
17. Red Hat, "Hibernate" [software] version 3.6.10.Final available from http://hibernate.org/orm/ [Accessed 2018-10-30]
18. The Jackson project, "Jackson" [software] version 2.7.4 available from https://github.com/FasterXML/jackson [Accessed 2018-10-30]
19. The bootstrap project, "Bootstrap" [software] version 3.3.7 Available from https://getbootstrap.com/ [Accessed 2018-10-30]
20. Facebook, "ReactJS" [software] version 15.6.2 Available from https://reactjs.org/ [Accessed 2018-10-30]
21. Almende B.V., "vis.js" [software] version 4.16.1 Available from http://visjs.org/ [Accessed 2018-10-30]