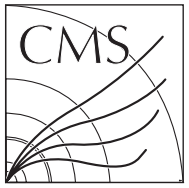
Available on CMS information server

**CMS CR -2010/090**



**The Compact Muon Solenoid Experiment**

**Conference Report**

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

**22 June 2010**

# First operational experience with the CMS Run Control System

Hannes Sakulin for the CMS Collaboration

**Abstract**

The Run Control System of the Compact Muon Solenoid (CMS) experiment at CERN's new Large Hadron Collider (LHC) controls the sub-detector and central data acquisition systems and the high-level trigger farm of the experiment. It manages around 10000 applications that control custom hardware or handle the event building and the high-level trigger processing. The CMS Run Control System is a distributed Java system running on a set of Apache Tomcat servlet containers. Users interact with the system through a web browser. The paper presents the architecture of the CMS Run Control System and deals with operational aspects during the first phase of operation with colliding beams. In particular it focuses on performance, stability, integration with the CMS Detector Control System, integration with LHC status information and tools to guide the shifter.

Presented at *RT2010: 17th IEEE NPSS Real Time Conference*

# First Operational Experience with the CMS Run Control System

Gerry Bauer, Barbara Beccati, Ulf Behrens, Kurt Biery, James Branson, Sebastian Bukowiec, *Member, IEEE*, Eric Cano, Harry Cheung, Marek Ciganek, Sergio Cittolin, Jose Antonio Coarasa Perez, Christian Deldicque, Samim Erhan, Dominique Gigi, Frank Glege, Robert Gomez-Reino, Michele Gulmini, Derek Hatton, Yi Ling Hwong, *Member, IEEE*, Constantin Loizides, Frank Ma, Lorenzo Masetti, Frans Meijers, Emilio Meschi, Andreas Meyer, Remigius K. Mommsen, Roland Moser, Vivian O'Dell, Alexander Oh, Luciano Orsini, Christoph Paus, Andrea Petrucci, Marco Pieri, Attila Racz, Olivier Raginel, Hannes Sakulin, *Member, IEEE*, Matteo Sani, Philipp Schieferdecker, Christoph Schwick, Dennis Shpakov, Michal Simon, *Member, IEEE*, Konstanty Sumorok and Andre Sungho Yoon

***Abstract*–The Run Control System of the Compact Muon Solenoid (CMS) experiment at CERN's new Large Hadron Collider (LHC) controls the sub-detector and central data acquisition systems and the high-level trigger farm of the experiment. It manages around 10,000 applications that control custom hardware or handle the event building and the high-level trigger processing. The CMS Run Control System is a distributed Java system running on a set of Apache Tomcat servlet containers. Users interact with the system through a web browser. The paper presents the architecture of the CMS Run Control System and deals with operational aspects during the first phase of operation with colliding beams. In particular it focuses on performance, stability, integration with the CMS Detector Control System, integration with LHC status information and tools to guide the shifter.**

## I. Introduction

THE Compact Muon Solenoid (CMS) experiment [1]-[2] at CERN's new Large Hadron Collider (LHC) is one of two large general-purpose detectors aimed at studying a broad range of physics at the TeV scale. Following an extensive phase of commissioning with cosmic muons, CMS saw its first proton-proton collisions in November 2009. Since then, routine operation with beams colliding at a center-of-mass energy of 7 TeV has been established. In this paper, we focus on the CMS Run Control System, which controls the sub-detector and central data acquisition (DAQ) [3] systems that are responsible for the data transport from the experiment's 55 million readout channels all the way to a 6,000 CPU-core computing farm running the on-line event selection algorithms and on to the storage system. The Run Control system provides the hierarchical control structure needed to control around 10,000 applications that in turn control electronics or handle the event building and processing. The applications themselves are developed using the C++ based XDAQ [4] data acquisition framework, that provides hardware access, powerful data transport protocols and services. Online event selection is based on the CMSSW [5] framework, which is also used for off-line reconstruction in CMS.

We first summarize the features and architecture of the CMS Run Control framework [6]-[7]. We then discuss the control hierarchy used in CMS and the top-level control node, the central entry point to CMS data taking operations, with a special emphasis on features that enable efficient operation of the experiment. In particular, we report on the recently achieved integration with the CMS Detector Control System (DCS) and with status information from the LHC.

## II. The Run Control Framework

The CMS Run Control System is built using web technologies. The Run Control framework runs as a web application in an Apache Tomcat servlet container. It is written in Java and makes use of Java Servlets, Java Server Pages, Tag Libraries and Web Services. The user interface runs in a web browser using technologies including HTML, Cascaded Style Sheets (CSS), Java Script and AJAX (Asynchronous Java Script and XML). User code is encapsulated in so-called *Function Managers (FMs)*, which are dynamically loaded into the web application. A Function Manager is a node in the control tree of the Run Control system. It controls a set of child resources and summarizes their state to the parent resource. Using the classes provided by the framework, Function Managers typically define a state

Manuscript received June 2, 2010. This work was supported in part by the DOE and NSF (USA) and the Marie Curie Program.

U. Behrens, D. Hatton and A. Meyer are with DESY, Hamburg, Germany.

B. Beccati, S. Bukowiec, E. Cano, M. Ciganek, S. Cittolin, J. A. Coarasa Perez, C. Deldicque, D. Gigi, F. Glege, R. Gomez-Reino, Y. L. Hwong, L. Masetti, F. Meijers, E. Meschi, R. Moser, L. Orsini, A. Racz, H. Sakulin (corresponding author. phone: +41 22 767 3506, fax: +41 22 767 8940, e-mail: Hannes.Sakulin@cern.ch), C. Schwick, and M. Simon are with CERN, Geneva, Switzerland. M. Gulmini, A. Oh, and P. Schieferdecker were with CERN, Geneva, Switzerland. M. Gulmini is now with INFN - Laboratori Nazionali di Legnaro, Legnaro, Italy. A. Oh is now with University of Manchester, Manchester, UK. P. Schieferdecker is now with Universität Karlsruhe, Karlsruhe, Germany.

J. Branson, M. Pieri and M. Sani are with University of California San Diego, La Jolla, California, USA. A. Petrucci was with University of California San Diego, La Jolla, California, USA. He is now with CERN, Geneva, Switzerland.

S. Erhan is with University of California, Los Angeles, California, USA and CERN, Geneva, Switzerland.

K. Biery, H. Cheung, R. K. Mommsen, V. O'Dell and D. Shpakov are with FNAL, Batavia, Illinois, USA.

G. Bauer, C. Loizides, F. Ma, C. Paus, O. Raginel, K. Sumorok and A. S. Yoon are with Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

machine and a set of parameters that can be set/read by the parent (Fig. 1). User code mainly consists of event handlers that handle state transitions, user input and asynchronous notifications from the child resources. The framework provides proxies to communicate with different types of child resources. Child resources are usually either Function Managers themselves, or XDAQ applications. Child function managers are accessed through a web service interface making use of the Web Service Definition Language (WSDL), the Simple Object Access Protocol (SOAP) and the Apache Axis library. Communication with XDAQ applications is performed through the SOAP interface provided by XDAQ applications. Function managers may also control or read the status of state machines and data points in the CMS Detector Control System by using the PVSS[1] SOAP eXchange (PSX) service of XDAQ.

Apart from the function manager infrastructure, the Run Control System offers a number of services:

The *Resource Service* [8] handles the configuration of both the Run Control system and the XDAQ applications in the system. Configurations are stored in an Oracle database. For the Run Control System, the configuration determines the Function Managers to be loaded, their parameters and the control structure in which they are arranged. For XDAQ applications, the configuration determines libraries to be loaded, applications to be instantiated, their parameters and their connectivity to other applications. At the start of a data-taking session, function managers are dynamically loaded and configured according to the configuration. The function managers then start up XDAQ applications by using the *Job Control* service (itself a XDAQ application) that is available on all the nodes in the cluster. Through this mechanism, the parameters and structure of the controlled DAQ systems (i.e. which applications run on which hosts) may easily be changed by switching to a different configuration.

Through the *DAQ Structure Service*, function managers may query information about the physical structure (modules, cabling, networks, etc.) of the DAQ system. Given the set of detector front-end drivers to be read out, the DAQ Structure Service determines the active sub-set of global DAQ resources.

Through the *Run Info* service, run conditions can be logged to a database.

The *Log Collector* service allows collecting log4j log messages from the Function Managers and log4cplus log messages from the XDAQ applications. The log messages are then written to file, to database or are available in a publish-subscriber infrastructure.

The Run Control framework also provides interfaces to XDAQ services such as the XDAQ Monitoring and Alarming Service (XMAS).

## III. The CMS Run Control Structure

As illustrated in Fig. 2, the entire control tree of the CMS Run Control System is controlled from the top-level Function Manger (Level-0 FM). During global data-taking, only this node is controlled by the operator through a web browser. At the next level, the Level-1 Function Managers provide the entry point to each of the sub-detectors, to the First-Level Trigger, and to the central DAQ system. The sub-detector Level-1 Function Managers implement a common state model and export a standard set of parameters for monitoring. Additional states exist for the Trigger and central DAQ systems. Further layers of Function Managers are sub-system specific. Each of the sub-systems has a dedicated Apache Tomcat server running an instance of the Run Control

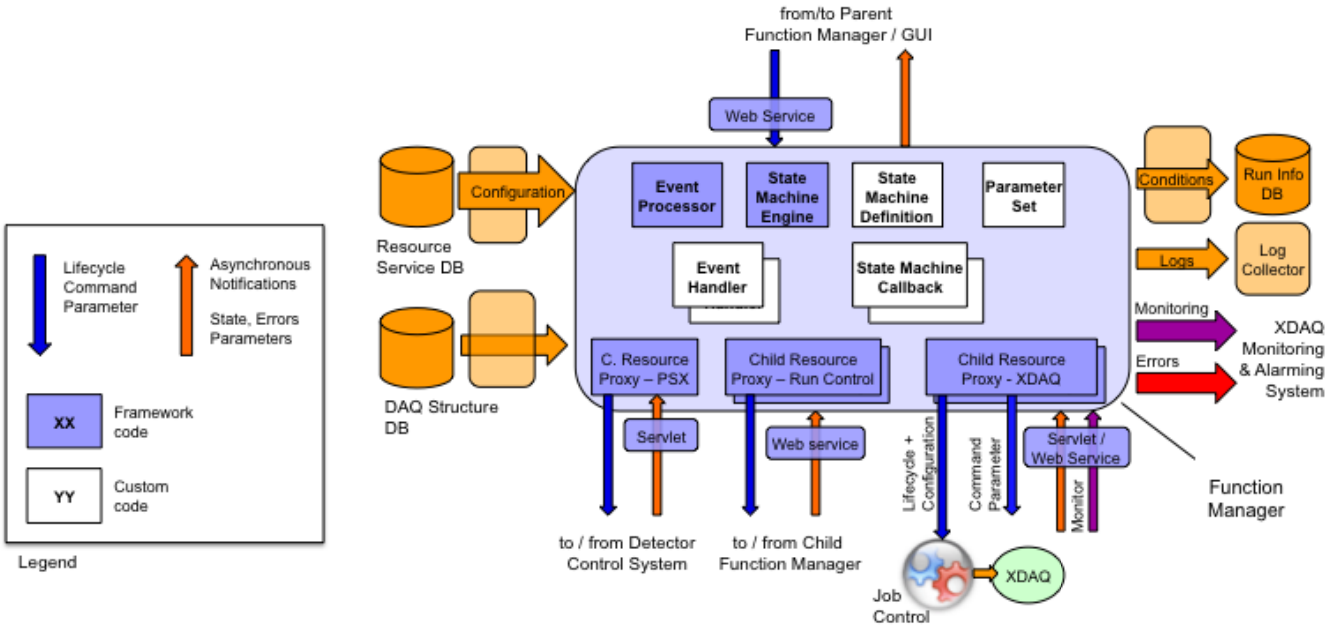



Fig. 1. Function manager framework and services provided by the CMS Run Control System.

[1] PVSS (Prozessvisualisierungs und Steuerungssystem), *www.etm.at*

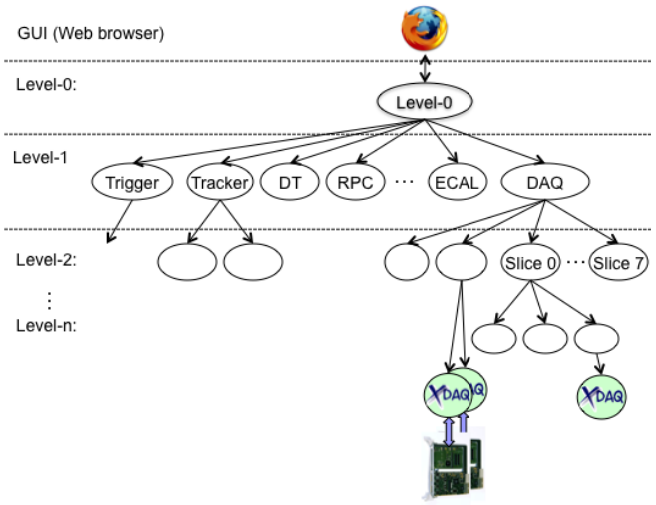



Fig. 2. CMS Run Control structure. The Level-0 Function Manager is the top-level control node. It provides the interface to the operator. The Level-1 Function Managers are the entry points to the CMS First Level Trigger (Trigger), Central Data Acquisition System (DAQ) and to the sub-detectors: Drift Tubes (DT), Resistive Plate Chambers (RPC), Si-Strip Tracker (Tracker), Electromagnetic Calorimeter (ECAL) and others (not shown).

framework. For the central DAQ system that controls the largest number of applications, five Apache Tomcat servers are used in order to optimize performance.

The state model defines the following main steps when starting a run: During Creation, the Level-1 Function Managers are loaded into the Run Control application and instantiated. During Initialization, all further levels of Function Managers are created and all XDAQ applications are started. During Pre-Configuration the clock source and periodic timing signals are set up. During Configuration, all sub-systems are configured. During Start, first the sub-detectors and central DAQ and then the First-Level Trigger are started. At the time of writing, the average time for a cold start of CMS (initialization, pre-configuration, configuration and start) is just over 4 minutes (Fig. 3). Having optimized handling of large numbers of applications (central DAQ system), configuration times are currently dominated by systems that configure a large number of front-ends such as the Silicon-Strip Tracker. Pausing and resuming are fast operations involving only some of the sub-systems while starting and stopping are more complex and time-consuming operations involving all of the sub-systems.

Sub-system configurations are handled by the respective sub-system groups. Each sub-system has a separate Resource Service (RS) database, which is filled by the general-purpose RS Manager tool or dedicated high-level tools such as the DAQ Configurator [8] used for the central DAQ system. Through a mechanism called the Global Configuration Map, sub-systems may register one of their configurations to be used for Global runs.

In order to operate a sub-detector or a group of sub-detectors independently, so-called *Mini-DAQ* setups may be used. These setups have their own control tree in Run Control with a separate top-level Function Manager and a separate Global Configuration Map. Triggers are typically provided by a Local Trigger Controller. Central Data Acquisition is provided by dedicated small-scale DAQ setups, which build events in the same way as the global DAQ system, but at a much lower rate. These setups have been indispensible during the commissioning phase of CMS. They are now mostly used in order to test new features or to debug problems.

## IV. The Top-Level Control Node

The Level-0 FM is the central entry point to CMS data-taking operations. It defines a global state machine that includes all the states of the sub-detector, Trigger and central DAQ state machines. In the simplest case, the operator will just step the global state machine through the various stages of the run start sequence in order to start a run with all sub-detectors. The Level-0 FM takes care of picking up the correct sub-system configurations, booking a data taking session, booking a run number and storing relevant information about the run to the Run Info database.

Through the Level-0 FM, the operator may also parameterize the configuration in many different ways. For each sub-system, a Run Key may be selected from a pre-defined set of keys indicating for example the level of zero-suppression. Furthermore the clock source and configuration keys for the First-Level and High-Level Trigger Systems may be selected. Entire sub-systems, sub-system partitions or individual Front-End-Drivers of a sub-system may easily be removed from the global run in case of needed local activities or problems. In case of the central DAQ system, slices corresponding to 1/8 of the system's capacity may be masked out in order to cope with computer or network problems.

Through the Level-0 FM, sub-systems may also be controlled individually. This feature helped to significantly shorten run startup time, as it allows for rapid recovery from sub-system errors at any stage of the start-up sequence. All parameters and masks set in the Level-0 FM are also taken into account when controlling sub-systems individually. It is therefore possible to quickly mask or unmask components of the readout without going through a full reconfiguration.

However, many dependencies need to be taken into account when operating sub-systems individually. Re-configuration of one sub-system may imply that re-configuration of another one becomes necessary. Changes of some parameters like the First Level-Trigger key or the included sub-detectors imply that multiple sub-systems need to be reconfigured. In order to increase operator efficiency, cross-checks have been added to the Level-0 FM, that keep track of all sub-system operations and parameter changes and indicates to the operator what sub-systems need to be re-configured or re-initialized. Thanks to the cross-checks even non-expert operators are able to recover from errors or to re-parameterize an already configured system with the least necessary steps.

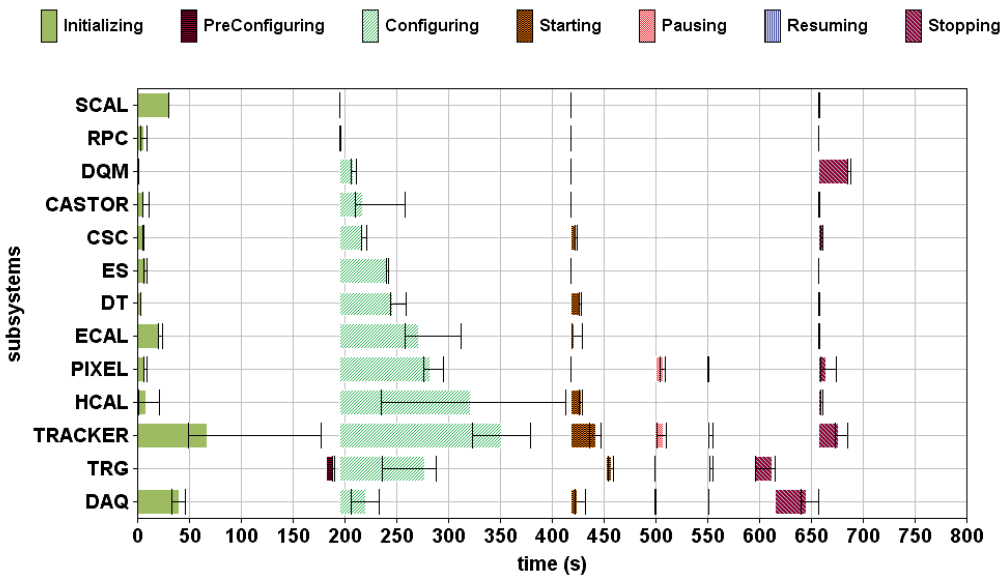



Fig. 3. Time needed by the CMS sub-systems for the actions defined by the Level-1 state machine. The solid bars indicate average time. Error bars are RMS errors separately calculated for larger and smaller than average configuration times. Data were gathered over two weeks of operation. Gaps of arbitrary duration have been added for better readability.

## V. Integration with the CMS Detector Control System and the LHC

During the commissioning of CMS operation with beams in the LHC, further requirements emerged. Some of the settings in the Detector Control System and Data Acquisition now depend on the state of the LHC. This especially concerns the domains of high voltages and clock stability. For collision data taking, the CMS clock needs to be synchronized to the LHC clock, which is guaranteed to be stable only in certain modes of operation of the LHC. Depending on the mode of operation of the LHC, CMS electronics may need to be re-configured or sensitive channels may need to be masked out. For reasons of detector safety, high voltages of some of the detectors may only be ramped up when the LHC is in a stable mode of operation. Criteria are more stringent for detectors close to the interaction point.

Operational experience showed that the best way of ensuring that CMS is ready for data taking, is to keep data taking runs going at all times, irrespective of the state of the LHC. Manually starting new data acquisition runs in order to follow changes in detector and/or LHC conditions was found to be inefficient and error prone. It was therefore decided to make the Run Control System aware of the states of the LHC and the Detector Control System and to implement automatic actions in response to state changes that would avoid any manual operations at critical times. A DCS Function Manager was developed to have a communication channel from DCS to Run Control. Through the PSX resource proxy and the XDAQ PSX service, it subscribes for notifications about DCS and LHC status changes. It summarizes status information and passes it on to the Level-0 FM. The Level-0 FM passes the status information to the concerned sub-systems when a run is started or when a run is resumed after a pause. The sub-detectors use this information in order to adapt their data acquisition setting to the LHC and DCS state, for example by masking out sensitive channels, by suppressing noisy data when high voltage is off or by re-programming thresholds in order to avoid high currents when high voltage is off. When the Level-0 receives a notification about a status change during an ongoing run, it briefly pauses the run and immediately resumes it with the new settings. Pausing and resuming was chosen over stopping and starting since it is faster (currently by a factor of 7) and more reliable as only some of the sub-systems need to be involved in the operation.

Due to the automatic actions a run can now be started with all detectors included as soon as the LHC provides a stable clock. During periods of LHC clock variations, sensitive channels are automatically masked out. High voltages are successively ramped up under the control of the DCS system when beam conditions allow. The Run Control System automatically triggers the necessary configuration changes in the sub-detector data acquisition systems. Data taken with all high voltages on is automatically marked by setting a flag in the trigger's data record.

The Level-0 FM also uses the information about the mode of operation of the LHC in order to automatically select the LHC clock or a local clock as the clock source for CMS. In order to protect against data integrity problems, additional cross checks remind the operator to reconfigure CMS sub-detectors whenever a clock source change or clock instability make it necessary.

## VI. Observations

The CMS Run Control System is now routinely operated by operators recruited from the entire CMS collaboration. Average CMS data taking efficiency (integrated luminosity

recorded by CMS with respect to integrated luminosity delivered by the LHC) has been at 95 % for all 2010 runs up to the time of writing (6 weeks in total). Streamlining of the procedure needed to start a run with colliding beams certainly has been an important ingredient in achieving this high efficiency. The tools built into the top-level control node aimed at optimizing operator efficiency and at minimizing time needed for error recovery also have been instrumental to achieving smooth operation.

Building the Run-Control System based on web technologies has the advantage that experts may easily connect to the system from anywhere in order to watch operations or to aid the operator. The stability of the server side (Apache Tomcat and Run Control web application) has been excellent with servers running for many weeks without a restart. Instabilities observed in the past were always traced back to faulty user code. Stability of the GUIs has been more of a concern since it depends on a third party tool - the web browser. Depending on the exact version of the browser used, crashes may occur more or less frequently. Fortunately, this is not a major problem since the Run Control System keeps its state independent of the GUI and the GUI may simply be reconnected.

In order to develop basic Function Managers, developers only need to know Java and need to learn how to use the classes provided by the Run Control framework. The framework provides a default web-based GUI through which the Function Manager's state machine may be controlled and handles all of the web technologies listed earlier. Developing more sophisticated user interfaces such as the top-level control node can however be cumbersome, as almost the full list of technologies needs to be mastered.

## VII. Summary and Outlook

The CMS Run Control System is based on Java and web technologies. The entire system and especially the top-level control node have been optimized for efficient operation. The top-level control node allows the operator to quickly recover from problems by commanding individual sub-systems. Built-in cross-checks ensure consistency of the over-all configuration and warn the operator about necessary actions. The procedure to start collision data taking has been streamlined by adding automatic actions based on the state of the Detector Control System and the LHC. Operators recruited from the entire CMS collaboration successfully use the system to control data taking operations of the CMS experiment which are currently proceeding with a data-taking efficiency of above 95 %.

Future work will focus on enhanced fault tolerance, automatic recovery procedures and further automation of operator tasks.

## References


[1] The CMS Collaboration, CMS Technical Proposal, CERN LHCC 94-38, 1994.

[2] The CMS Collaboration (R. Adolphi et al.), "The CMS Experiment at CERN LHC", *JINST* 3 S08004 p. 361, 2008

[3] The CMS Collaboration, CMS, The TriDAS Project, Technical Design Report, Volume 2: Data Acquisition and High-Level Trigger, CERN/LHCC 2002-26, 2002.

[4] G. Bauer et al., "The CMS data acquisition system software", *J. Phys.: Conf. Ser.* 219 022011, 2010

[5] C. D. Jones et al., "The new CMS data model and framework", *CHEP'06 Conference Proceedings*, 2007

[6] M. Bellato et al., "Run control and monitor system for the CMS experiment," presented at *Computing in High Energy and Nuclear Physics*, La Jolla CA, March 24-28, 2003

[7] G.Bauer et al., "The run control and monitoring system of the CMS experiment," *PoS(ACAT)* 026, 2007

[8] G. Bauer et al., "Dynamic configuration of the CMS data acquisition system", *J. Phys.: Conf. Ser.* 219 022003, 2010