PROCEEDINGS OF SCIENCE

The Run Control and Monitoring System of the CMS Experiment

G. Bauer[g], V. Boyer[d], J. Branson[f], A. Brett[d], E. Cano[d], A. Carboni[d], M. Ciganek[d], S.Cittolin[d], S. Erhan[de], D. Gigi[d], F. Glege[d], R. Gomez-Reino[d], M. Gulmini[ad], E. Gutierrez Mlot[d], J. Gutleber[d], C. Jacobs[d], J. C. Kim[c], M. Klute[g], E. Lipeles[f], J. Lopez Perez[d], G. Maron[a], F. Meijers[d], E. Meschi[d], R. Moser[di], S. Murray[h], A. Oh[d], L. Orsini[d], C. Paus[g], A. Petrucci[*a], M. Pieri[f], L. Pollet[d], A. Racz[d], H. Sakulin[d], M. Sani[f], P. Schieferdecker[d], C. Schwick[d], K. Sumorok[g], I. Suzuki[h], D. Tsirigkas[d] and J. Varela[bd]

[a] *INFN - Laboratori Nazionali di Legnaro, Legnaro, Italy*
[b] *LIP, Lisbon, Portugal*
[c] *Kyungpook National University, Daegu, Kyungpook, South Korea*
[d] *CERN, Geneva, Switzerland*
[e] *University of California, Los Angeles, Los Angeles, California, USA*
[f] *University of California, San Diego, La Jolla, California, USA*
[g] *Massachusetts Institute of Technology, Cambridge, Massachusetts, USA*
[h] *FNAL, Batavia, Illinois, USA*
[i] *Vienna University of Technology, Vienna, Austria*

The CMS experiment at the LHC at CERN will start taking physics data towards the middle of 2008. To control and monitor the experiment during data-taking the Run Control and Monitoring System (RCMS) was developed. This paper describes the architecture and the technologies used to implement the RCMS, as well as the deployment and commissioning strategy. The RCMS framework is based on a set of web-applications implemented with Web Service and Java Servlet technologies. AJAX and JSP are used for the user interfaces, and MySQL and Oracle are supported as the DB backend. A hierarchical control structure organizes the Run Control into subsystems. The RCMS was successfully used in the "Magnet Test & Cosmic Challenge of CMS" in 2006. The goal of this exercise was to integrate and operate a sub-set of the components of CMS in order to record cosmic rays.

XI International Workshop on Advanced Computing and Analysis Techniques in Physics Research
April 23-27 2007
Amsterdam, the Netherlands

*Corresponding author: Andrea Petrucci (Andrea.Petrucci@cern.ch)

PoS(ACAT)026

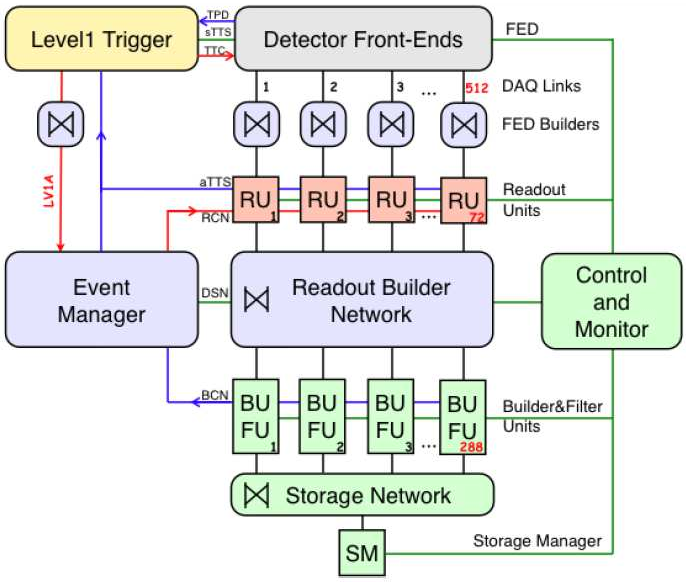

Figure 1: The CMS data acquisition architecture.

1. Introduction

The Compact Muon Solenoid (CMS) [1] is a general purpose particle detector currently under construction at CERN in Geneva, Switzerland. It is scheduled to start taking data from the proton-proton collisions produced at the Large Hadron Collider (LHC) [2] at $\sqrt{s}$=14 TeV in 2008. The CMS Data Acquisition System (DAQ) [3] (Fig. 1) is responsible to build and filter events from about 600 data sources at a maximum trigger rate of 100 KHz. The Run Control and Monitor System (RCMS) has to provide the interfaces to configure, control and monitor the system. The RCMS architecture is designed to hide the diversity of the involved hardware and to be highly scalable, as the CMS DAQ system is foreseen to be extended by almost an order of magnitude during the lifetime of the experiment.

Current Internet applications have requirements similar to those of the RCMS: many distributed clients must interoperate with a set of servers and databases; scalability in terms of database transactions per second and connectivity to the services; and providing secure access. Most Internet applications have adopted Web technologies and in particular Web Services [4]. The Extensible Markup Language (XML) data format [5] is commonly used for data exchange and the Simple Object Access Protocol (SOAP) [6], for communication. Since the main requirements of the RCMS are similar to those of Internet applications, CMS has therefore chosen to adopt commonly used technologies in order to profit from existing Information Technology (IT) developments.

The RCMS software is developed in collaboration with the Grid enabled Remote Instrumentation with Distributed Control and Computation (GRIDCC) [7] project. The GRIDCC is a project funded by the European Community, aimed to provide access to and control of distributed complex instrumentation. One of the main applications of the GRIDCC is the RCMS of CMS.

The next section presents requirements and architecture and gives a general overview of the software technologies and design of the RCMS. The RCMS experience at the Magnet Test & Cosmic

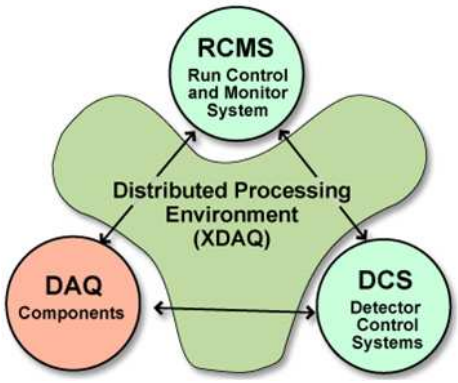

Figure 2: The RCMS integration with the CMS On-line system.

Challenge (MTCC) of CMS is presented, followed by a summary.

2. Run Control and Monitor System (RCMS)

As a component of the CMS online software, the RCMS is designed to interoperate with the other online software components, like the Detector Control System (DCS), and the XDAQ cross-platform DAQ framework [8, 9] (see Fig. 2).
The RCMS views the experiment as a collection of setups, where a setup is a configurable group of resources. Multiple setups can be active concurrently, sharing resources if necessary, allowing several sections of the experiment to run independently.
Users have to be authorized and authenticated to get access to the RCMS functions. Multiple users can access the experiment concurrently, performing subsets of possible functions exported by the system. Hierarchies of distributed control applications will be used to control the $O(10^4)$ objects of the CMS DAQ system.

2.1 Requirements

The RCMS is the master controller of the CMS DAQ when the experiment is taking data. The three main requirements are:

- to ensure the correct and proper operation of the CMS experiment.
- to control and monitor of the data acquisition system.
- to provide user interfaces.

A number of Sub-Detectors is involved in data-taking. The DCS sub-system, from the point of view of the RCMS, is an external system with its own independent partition.

2.2 Architecture

The logical structure of the RCMS is shown in Fig. 3. The system is organized into a tree structure of controllers and a Graphical User Interface (GUI). The Top Controller (TC) is the unique

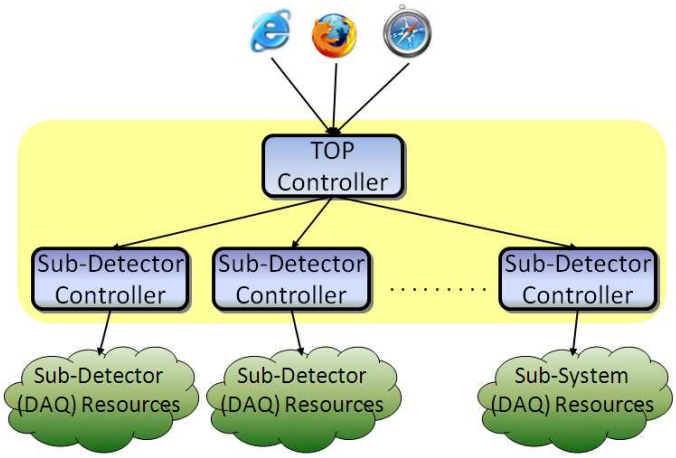

Figure 3: The RCMS logical structure.

entry point for the CMS system and drives the Sub-Detector Controllers (SDCs). Sub-Detector DAQ Resources are managed by the respective SDC. Every Controller has a set of services that are needed to support specific functions like security, logging and resource management.
During data taking, the RCMS deals with the DCS to set up and monitor the partitions corresponding to the detector setups involved in the acquisition.
Each setup is associated with a TC that coordinates user access to the setup and propagates commands from the users to the SDCs. High-level commands are interpreted and expanded into sequences of commands for the DAQ Sub-Detector resources.

2.3 Software Technologies

Web technologies and related developments play a fundamental role in the implementation of the RCMS design. The XML data format and the SOAP communication protocol have already been mentioned in the introduction. The Web Services Description Language (WSDL) [10] is used to export service interfaces to clients.
A rich choice of tools and solutions based on the previously mentioned Web technologies is available. Apache Axis [11] is widely used to implement Web Services. The Grid [12] community use the same technologies.
Relational database management systems provide the required level of robustness and scalability. For security issues the RCMS is going to adopt one of the existing solutions available in the scope of large distributed systems.

2.4 Design

The RCMS has been developed using the Java programming language [13]. It makes use of the Apache Axis software tools. All the services are developed as Web Services that run in a Tomcat servlet container [14].
The RCMS is currently used in some application scenarios of the CMS experiment. Permanent

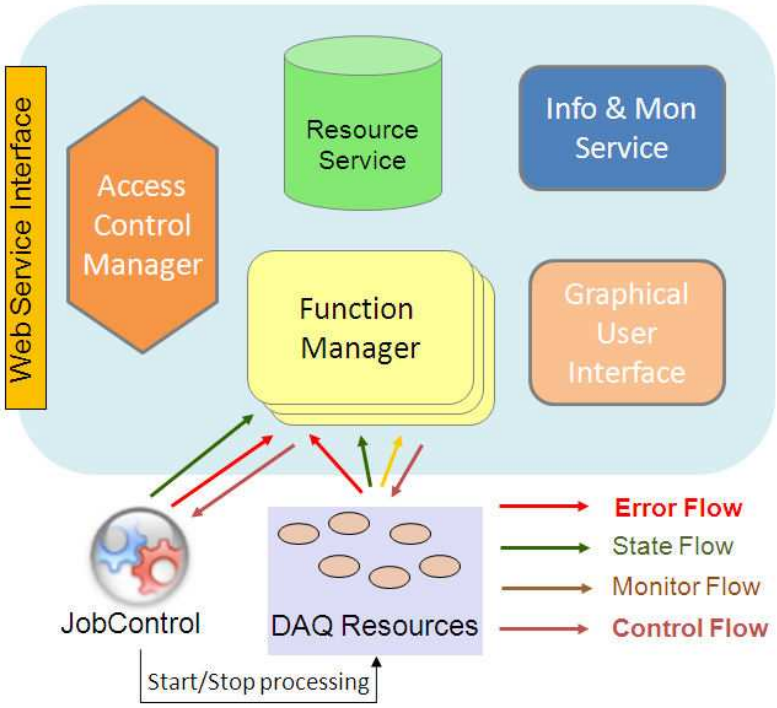

Figure 4: The block diagram of the Run Control and Monitor System.

installations are used for the test and validation of the CMS muon chambers [15] and for the tracker system test [16]. Other installations are used for testbeam DAQs, both for tracker, ECAL [17] and muon detectors, and for DAQ global and local tests. Figure 4 shows a block diagram of the controller, which consists of components of six different types: Function Managers (FMs), Resource Service (RS), Information and Monitor Service (IMS), Graphical User Interfaces, Job Control (JC) and Access Control Manager (ACM).

2.4.1 Function Manager

The FM is responsible for performing the communications with the controlled set of resources. It acts as a protocol adapter that implements the resource-specific protocols for accessing their functions and reading their status. FMs can be organized into a hierarchical control structure in order to break down the complexity and scalability of the system.

The FM comprises the following components (see Fig. 5):

- *"Input Handler"*: this module handles all the inputs to the FM, including commands from external entities (e.g. GUI) and error messages or states from the resources. Messages are then passed to the Event Processor.

- *"Event Processor"*: this module handles the messages coming from the Input Handler. According to a user defined behaviour, such messages can trigger a state transition in the FSM module and/or act on the Resource Proxy to directly control the associated resources.

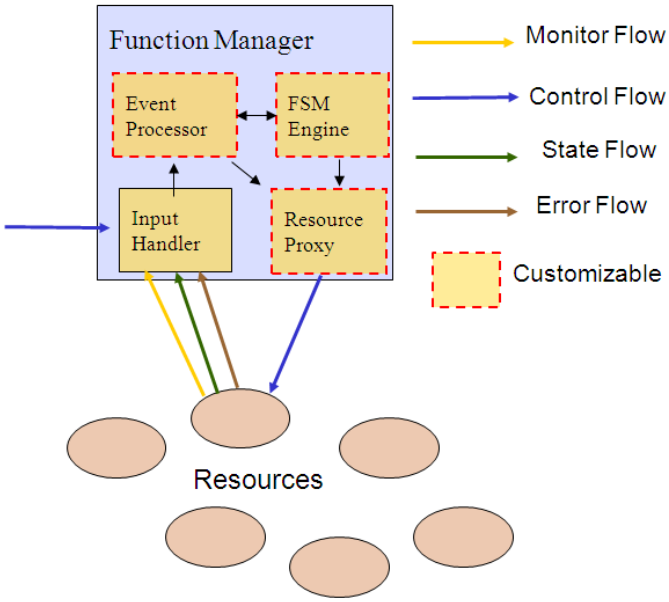

Figure 5: The Function Manager architecture.

- *"Finite State Machine (FSM) Engine"*: this module is a user customizable state machine. It performs state transitions and executes the actions allowed by the current state through the Resource Proxy.
- *"Resource Proxy"*: this module deals directly with the resources, having the knowledge on how to communicate with them (i.e. which protocol is used), where and how many they are. The module is a plug-in module that can be easily replaced according to the type of Resources that should be controlled.

The typical use case is to receive inputs from the users who are supervising the resources. Since this component also receives inputs like states or errors that come from the set of resources it can react to the behaviour of the controlled resources, allowing automatic recovery procedures to be started. One or more Function Managers are commoonly used for each controller.

2.4.2 Resource Service

The Resource Service (RS) is responsible for the organization and management of resources available in the system and about configurations which reference these. A set of controllable resources and FMs, organized in a hierarchical tree structure, constitutes a Configuration. A resource can be any hardware or software component involved in a setup. Resources can be allocated and queried and setups can only use available resources.
The main requirements for the RS are listed below:

- RS ahould be able to extract information on available configurations from storage.
- RS should store the information about new or modified configurations.

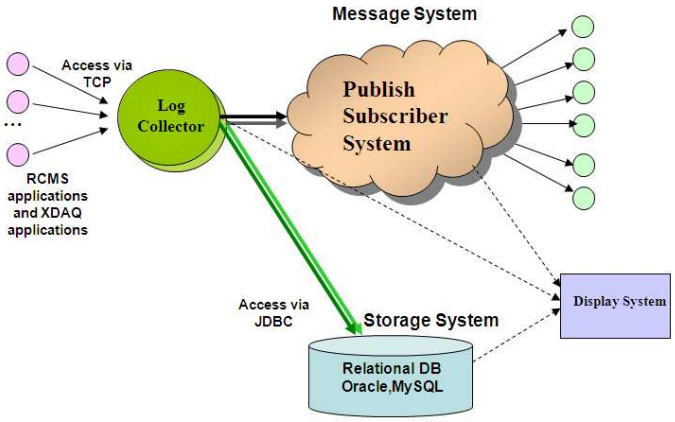

Figure 6: The Information and Monitor Service.

- A Configuration must contain all the details to ensure that Resources are properly initialized and working correctly.

It is the responsibility of the RS to check resource availability and contention with other active partitions when a resource is allocated for use in a setup.
The solution designed for the Resource Service uses a Web Service interface that guarantees a high interoperability and a standardized access to the Resource Service component in order to "read" any resources configuration from or to the resource catalog.
The persistent storage is based on a DBMS (Database Management System) such as MySQL [18] or Oracle [19]. The connection between the repository database and the Resource Service backend relies on Java Database Connectivity (JDBC) technology [20].

2.4.3 Information and Monitor Service

Information concerning the internal status of DAQ resources and RCMS components are sent to the IMS as messages, which are reported to the operator or stored in a database.
The messages are catalogued according to their type, severity level and timestamp. The IMS collects and organizes the incoming information in a database and publishes it to subscribers. Subscribers can register for specific messages categorized by a number of selection criteria, such as timestamp, information source and severity level.

This service uses a publish/subscribe system to disseminate monitoring data to the interested partners (see Fig. 6). The IMS system manages the received information in different ways:

- It publishes data via a Java Message Service (JMS) [21] system arranged in topics.
- Subscribers can register themselves for specific messages that can be selected on the basis of the information source, the error severity level, the type of messages or a combination of these tags.
- It stores data in a persistent repository (database). In this case event data are permanently recorded and interesting information can be retrieved by a client via simple query procedures to the database.

- It can distribute data via SocketAppender (a simple TCP/IP communication).

Clients can publish messages to IMS using Log4C [22] and Log4J [23].

2.4.4 Graphical User Interfaces

A number of web applications are employed to operate the system. GUIs are very important to provide a set of intuitive functions to control and monitor the experiment. The Resource Service has to provide GUIs for storing new resources, defining and browsing partitions. The Information and Monitor Service provides displays for alarm messages, error statistics, system status and monitoring.
The Controller GUI is based on JSP [24] and Ajax [25] technologies, it can run in current web browsers. It is composed of a generic framework, released together with the services, containing the basic functionalities common to any CMS DAQ. These include the interface to the Resource Service for browsing configurations and the ability to command DAQ applications. To fit the specific requirements of each Sub-Detector, the GUI can be extended by new JSP pages.

2.4.5 Job Control

The Job Control (JC) is a utility that allows the remote execution and supervision of any software process involved in data-taking. RCMS needs a way to get around the access/authentication mechanisms of the infrastructure of our cluster, in order to be able to operate the whole cluster remotely. JC is a daemon process which needs to be up and running on each of the nodes to allow RCMS to start/stop arbitrary processes without the need to authenticate. This component is provided by XDAQ.

2.4.6 Access Control Manager

The Security Service provides facilities for user authentication and authorization, including data encryption if necessary. The ACM uses the Apache Tomcat Realm [26]. It is a simple authentication service like the Unix operating system based on usernames, passwords and roles. One or more roles can be associated to an username, the access to a web application is allowed to all users possessing a particular role. It uses a relational database to store the data associated with user profiles and access privileges.

3. RCMS at the Magnet Test & Cosmic Challenge (MTCC)

During summer and autumn 2006 a subset of the CMS detector including the 4 T magnetic field was operated to detect cosmic muons. The MTCC was a milestone of the CMS experiment as it completed the commissioning of the magnet system (coil and yoke) before its lowering into the cavern and demonstrated the working of the full read-out and control chain.
The main goals of the Cosmic Challenge were: Commissioning of several sub-detectors (Muon Detectors, Hadron and Electromagnetic Calorimeters, Tracker) and parts of the Trigger system; Test Muon alignment systems; Readout all available detectors, demonstrate cosmic ray reconstruction across CMS.

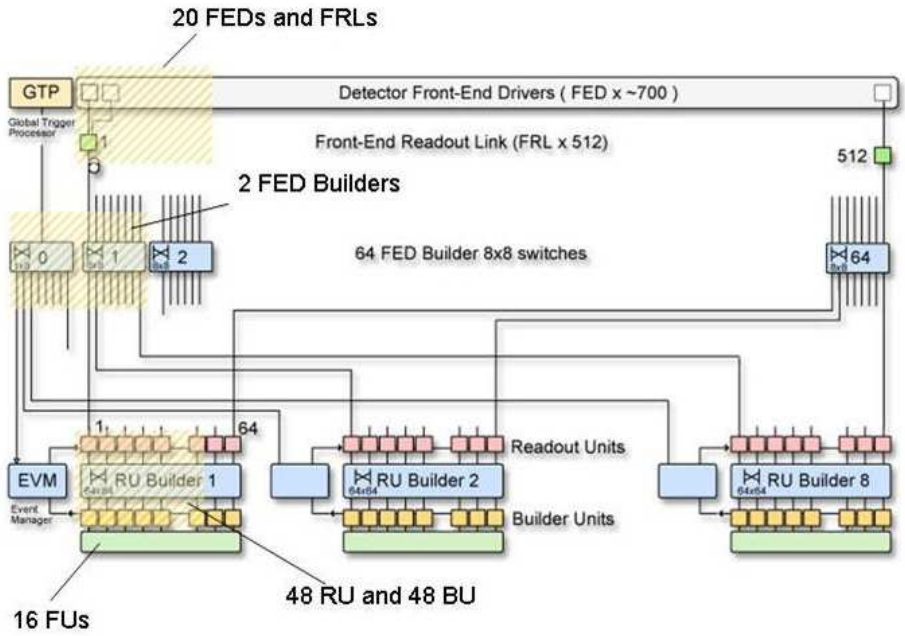

Figure 7: The CMS DAQ system layout.

3.1 DAQ Setup

The DAQ group of CMS has built a MTCC prototype of the final DAQ system, which includes RCMS, DAQ and DCS components.

The Event Builder is implemented in two-stages, the FED Builder and the Readout Builder stage. An overview of the data flow is illustrated in Figure 7 proceeding from top to bottom. At the top are the Front-End Drivers (FEDs) which are the sub-detector specific data sources. The FEDs feed the Front-End Readout Links (FRLs) which merge the data of up to two FEDs into one stream. The outputs of the FRLs are merged into Readout Units (RUs). The RUs read the data from FEDs and perform a first concatenation of the event data fragments into larger data blocks.

The Readout Builders consist of three types of software components: RU, Builder Unit (BU) and Filter Unit (FU). The RUs receive the event fragments and distributes them to the BUs which assemble full events and pass them to the FUs. The BUs constitute the last stage of the event building process and hold complete events in their memory. The event data are handled by the FUs for processing, and dispatching to the Storage Manager (see Fig 1).

Figure 7 shows the layout of all the components highlighting the ones participating in the MTCC system. The prototype consisted of 20 FEDs and 20 FRLs, the ensemble of 8 FRLs and 8 RUs is called the FED builder. In the final system 64 FED Builders with a total of 512 FRLs and about 600 FEDs will be operating, whereas in the MTCC only 2 FED Builders were operational.

The second stage of the DAQ system comprised 48 RUs and 48 BUs, an Event Manager (EVM) and a Trigger system. In the final system 576 RUs, about 1600 BUs/FUs and 8 EVMs are foreseen. Eight of the BUs were connected to 16 FUs and delivered their event data over Gigabit Ethernet.

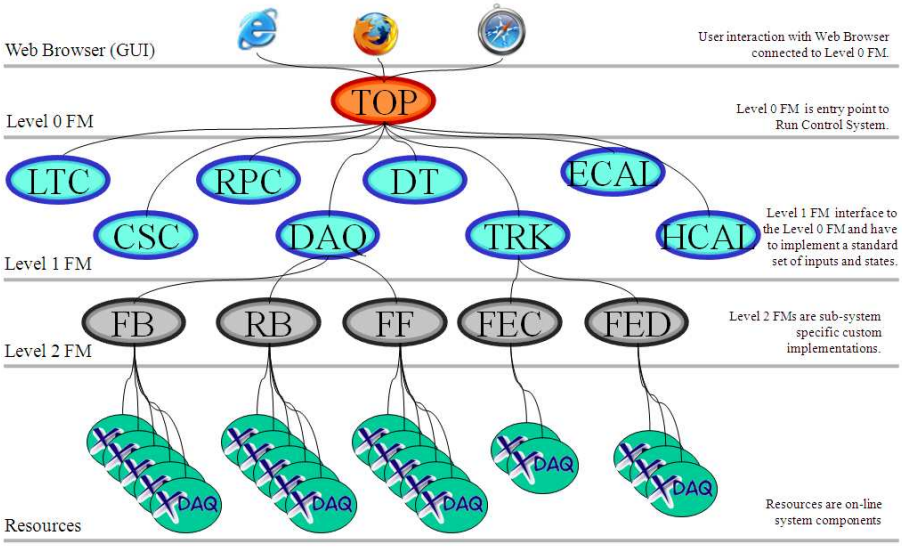

Figure 8: FMs Control Structure at MTCC I & II.

3.2 RCMS Deployment

At the root of the controller tree, a dedicated master controller is used to control the SDCs. Each SDC has a RS, one or more FMs and a Log Collector, the basic implementation of the IMS.
The tree structure of the control flow system is highlighted in figure 8, the Top is the entry point for control: user commands to the system are issued to the Top. The second level of the tree gives access to the sub-systems, namely to all the sub-detectors, to the trigger and data acquisition system (DAQ), and to the Detector Control System (DCS).
The sub-detectors configure and control the related FEDs to produce the data related to that part of the detector.
The DAQ, in the center part of the Figure 8, supervises 3 children: the FB (Fed Builder) controls the FRLs and RUIs, the RB (Ru Builder) controls the RU, BU, EVM and TA and the FF (Filter Farm) controls the FU components.
The Top and the DAQ run on a single machine, all the sub-detectors run on another machine. For each subsystem and the Top there is a Tomcat application server, together with the JSP servlet for the Run Control web interface generation and a logging service. Subsystem logging services deliver log messages to a central logging collector service that stores them in an Oracle database and allows for on-line log visualization on a GUI (Chainsaw [27]).
Each node hosts a Job Control service to start and stop DAQ components remotely. Another component of the control system is the Resource Service which uses an Oracle database to store configuration data of the control system and of the application nodes running the DAQ applications. All nodes run under Scientific Linux CERN 3 (a CERN customized version of RedHat Enterprise Linux 3).

4. Summary

In this paper, the architecture and the design of the Run Control and Monitor System for the CMS experiment have been described. Facing similar requirements as Internet applications, Web Service technologies have been adopted.
The final system is currently under development. Current and previous releases have been successfully used in test beam DAQ systems, in the central DAQ of CMS, and in the MTCC.

Acknowledgment

We acknowledge support from the DOE and NSF (USA), KRF (Korea) and Marie Curie Program.

References

[1] The CMS Collaboration, *The Compact Muon Solenoid Technical Proposal*, CERN/LHCC 94-38 (1994).

[2] The LHC Study Group, *The Large Hadron Collider Conceptual Design Report*, CERN/AC 95-05 (1995).

[3] The CMS Collaboration, *The Trigger and Data Aquisition project*, CERN/LHCC 2002-26, 15 December 2002.

[4] G. Glass, *Web Services: Building Blocks for Distributed Systems*, Prentice Hall, 2002.

[5] J. Boyer, *Canonical XML version 1.0*, W3C Recommendation, 15 March 2001 (see also http://www.w3c.org/XML).

[6] D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*, W3C Note, 08 May 2000 (see also http://www.w3c.org/TR/SOAP).

[7] *The GRIDCC Project* (EC Framework Programme 6 IST-511382), http://www.gridcc.org.

[8] J. Gutleber and L. Orsini, *Software Architecture for Processing Clusters based on I2O*, Cluster Computing, the Journal of Networks, Software and Applications, Kluwer Academic Publishers, 5(1):55-65, 2002.

[9] J. Gutleber et al., *Clustered Data Acquisition for the CMS experiment'*, Computing in High Energy and Nuclear Physics, Beijing, China, September 3-7 2001, Science press, pp.601-605.

[10] E. Christensen, *Web Services Description Language (WSDL) 1.1*, W3C Note, 15 March 2001 (see also http://www.w3c.org/TR/wsdl).

[11] *the Apache Axis Project*, http://ws.apache.org/axis/.

[12] I. Foster, C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, November 1998.

[13] *The Java Technologies*, http://java.sun.com/.

[14] *The Apache Jakarta Project*, http://tomcat.apache.org/.

[15] The CMS collaboration, *The Muon Project*, CERN/LHCC 97-32, CMS TDR 3, December 15, 1997.

[16] The CMS collaboration, *The Tracker System Project*, CERN/LHCC 98-6, CMS TDR 5, April 15, 1998.

[17] P. Musella et al., *The CMS Electromagnetic Calorimeter Data Acquisition System at the 2006 Test Beam* - 15th IEEE NPSS Real Time Conference 2007 (RT07) - Fermilab, Batavia (IL)

[18] *MySQL: the World's Most Popular Open Source Database*, http://www.mysql.com/.

[19] *The Oracle Database*, http://www.orcale.com/.

[20] *Java Database Connectivity (JDBC)*, http://java.sun.com/products/jdbc.

[21] *Java Message Service Specification, version 1.1*, http://java.sun.com/products/jms/.

[22] *Log4c : Logging for C Library*, http://log4c.sourceforge.net/.

[23] *Log4J: The complete manual*, http://logging.apache.org/log4j.

[24] *JavaServer Pages Technology, version 2.0*, http://java.sun.com/products/jsp/.

[25] J. Garrett, *Ajax: A New Approach to Web Applications*, Adaptive Path (2005-02-18), Retrieved on 2006-08-01.

[26] *Apache Tomcat Realm*, http://tomcat.apache.org/tomcat-5.0-doc/realm-howto.html.

[27] *Chainsaw*, http://logging.apache.org/log4j/docs/chainsaw.html.