The architecture of the CMS Level-1 Trigger Control and Monitoring System using UML

Marc Magrans de Abril  
University of Wisconsin, Department of Physics, Madison, Wisconsin, US  
E-mail: marc@cern.ch

Jose L. Da Rocha Melo  
Laboratorio de Instrumentacao e Fisica Experimental de Particulas, Lisbon, Portugal

Carlos Ghabrous Larrea  
University of Wisconsin, Department of Physics, Madison, Wisconsin, US

Josef Hammer  
Physics Department, CERN, Geneva, Switzerland

Christian Hartl  
Institute of High Energy Physics, Austrian Academy of Sciences, Vienna, Austria

Christos Lazaridis  
University of Wisconsin, Department of Physics, Madison, Wisconsin, US

Abstract. The architecture of the Compact Muon Solenoid (CMS) Level-1 Trigger Control and Monitoring software system is presented. This system has been installed and commissioned on the trigger online computers and is currently used for data taking. It has been designed to handle the trigger configuration and monitoring during data taking as well as all communications with the main run control of CMS. Furthermore its design has foreseen the provision of the software infrastructure for detailed testing of the trigger system during beam down time. This is a medium-size distributed system that runs over 40 PCs and 200 processes that control about 4000 electronic boards. The architecture of this system is described using the industry-standard Universal Modeling Language (UML). This way the relationships between the different subcomponents of the system become clear and all software upgrades and modifications are simplified. The described architecture has allowed for frequent upgrades that were necessary during the commissioning phase of CMS when the trigger system evolved constantly. As a secondary objective, the paper provides a UML usage example and tries to encourage the standardization of the software documentation of large projects across the LHC and High Energy Physics community.
1. Introduction
The Compact Muon Solenoid (CMS) experiment is one of the two general-purpose particle physics detectors built for the Large Hadron Collider (LHC) at CERN, the European Organization for Nuclear Research [1]. Since millions of collisions occur each second and only a small fraction of these can be stored, events have to be selected online according to their properties. The trigger system is meant to reduce the data rate of $40\ TB/s$ down to $100\ MB/s$, which is the maximum rate that can be handled by the data acquisition system, and finally distributed and stored in the LHC Computing Grid (see figure 1) [2]. The trigger system has been implemented in two consecutive stages (Levels). The Level-1 Trigger (L1) consists of custom-developed and largely programmable electronics, and the High Level Trigger (HLT) is a large computer farm [3, 4].

![Figure 1. Diagram of the CMS event data flow. The boxes represent electronic, computer or storage subsystems. The black arrows represent the flow of event data, while the grey ones represent control signals. This is the L1 accept signals and the feedback from the Readout buffers status.](image)

The objective of the present paper is two-fold. The primary objective is to discuss and present the L1 Trigger Control and Monitoring System (L1CMS) architecture and its key aspects. The L1CMS is a medium-scale (i.e. 40 computers, 200 processes, 800 thousand lines of source code) distributed software system meant to configure, monitor and test the L1, and to provide appropriate human and machine interfaces for experts, shifters, and the Run Control and Monitoring System (RCMS) of the CMS experiment [5, 6]. The system has already been installed and commissioned, and it is being used for data taking. The description of its architecture has an interest per se as the description of a successful system, and as documentation for existing developers, operators, and managers to cope with the future maintenance and improvements.

The secondary objective is to demonstrate that a system architecture can be described without the help of metaphorical diagrams or pictures. This is, the authors advocate for the use of standardised diagram notations to describe software systems [7]. This paper will use a subset of UML (Universal Modelling Language) to describe the run-time structure and behavior of the L1CMS [8, 9]. The authors consider that amongst the different notations UML has the best trade-off between number of users, learning curve, and degree of formalisation. The compile-time structure of the system won’t be described because this is already achieved by the automatic documentation tools (i.e. Doxygen, javadoc, etc.), and because it is on the run-time behavior where there are the main comprehension difficulties.

The paper is organized into six sections with one or more diagrams each. Section 2 describes the L1CMS interfaces. Section 3 to 5 present the architecture of the configuration, monitoring and testing services. Finally, section 6 summarizes the work.
2. Actors and interfaces
This section is meant to provide the highest-level view of the L1CMS and the rationale behind its key characteristics.

Figure 2 shows the L1CMS context diagram. The diagram shows the different actors interacting with the L1CMS and the origin of the interaction. The head of the arrow points to the subsystem providing the interface, while the tail touches the subsystem requiring it. If there is no arrow, a bi-directional interface is implied. This diagram helps developers to establish borders (and therefore responsibilities) between the system and its environment.

The following key aspects should be noted:

- **Web services based architecture.** All the interfaces are web service based except the ones for Online-to-Offline (O2O), L1 Emulators, Web Based Monitoring (WBM), and of course, the L1 electronics (non-interrupt driven commands through PCI-VME bridge).
- **UNIX pipes over Secure Shell (SSH) to interface O2O and L1 emulators.** This was an adhoc solution to integrate the CMSSW in the L1CMS [10].
- **Oracle Database (DB) as the configuration DB backend.** The CERN Information Technologies department provides access to and administration of an Oracle DB. Therefore, this Oracle DB is the most cost-effective way to access persistency services.
- **Query Web Based Monitoring (WBM) DB to retrieve Luminosity data.** WBM provides all kinds of CMS configuration and monitoring data through an HTTP GUI, and also through public Oracle tables and views [11].

![Figure 2. Context Diagram of the L1CMS. The diagram shows the different actors interacting with the L1CMS.](image)

3. Configuration service
This section is meant to describe the configuration service using two different views. The first view is a component diagram with the runtime dependencies and associated interfaces. The second view is the hierarchical activity diagram that models the different steps to configure and start the L1.

Figure 3 is a component diagram of the subsystems involved in the configuration process. The arrows represent the required (beginning) and provided (ending) remote interfaces and their types. The diagram refines the information described already in figure 2. The configuration follows a hierarchical topology from the RCMS to the subsystems. The diagram can be used...
also to backtrack the experiment faults to the possible source of the error, or to foresee the impacts of a change while following the causal path of the configuration process. There are five components worth mentioning:

- **TRG FM** (Level-1 Trigger Function Manager). The TRG FM is meant to match the remote interfaces between RCMS and Trigger Supervisor (TS) frameworks.
- **Central Cell.** The Central Cell coordinates the interplay between L1 subsystems.
- **O2O.** The O2O process is meant to transfer the configuration data between the online and offline DBs.
- **WBM.** The Global Trigger (GT) retrieves the luminosity data from WBM DB in order to select the correct prescale set.
- **L1CE** (Level-1 Configuration Editor). The editor is a web page that simplifies the creation and maintenance of configuration data.

**Figure 3.** Component diagram of the L1CMS with its remote dependencies. The arrow begins where the interface is required and ends where it is provided. Five interface types are shown.

On the other hand, figure 4 shows a hierarchical **activity diagram** of the CMS configuration process. The diagram represents the three steps needed to start the CMS data taking (i.e. **preconfigure**, **configure**, and **start**), and the one to stop it. As the diagram shows, the L1 is configured in two steps instead of just one. This peculiarity allows the parallel configuration of CMS once the clock distribution is set up in the Trigger Timing and Control Machine Interface (TTC MI) and the Global Trigger (GT) (i.e. **preconfigure** activity). The diagram further refines the causal and time dependencies between subsystems during configuration. Therefore, it further simplifies the location of errors and the consequences of high-level changes.

**4. Monitoring service**
This section is meant to describe the monitoring service of the L1CMS system. The service is described in two parts. First, the remote interfaces are shown in a component diagram. Then, the rationale of the key architectural aspects is explained.

**Figure 5** shows the **component diagram** of the monitoring services and the remote interfaces involved. This diagram further refines the diagrams in figures 2 and 3. All the monitoring information can be retrieved from HTTP Graphical User Interfaces (GUI). The following components require additional details:
Figure 4. Activity diagram of the configuration process of the L1CMS and its subsystems. The activity labels identify the subsystem and the activity name itself (i.e. subsystem:activity). The following abbreviations apply to the different activities: P=Preconfigure, C=Configure, S=Stop or Start.

- **L1 Page.** The L1 Page is the entry point for the L1CMS monitoring services. For each subsystem, the web page provides access to the status of its process(es), its configuration state, the hardware state, and input/output rates.
- **WBM** provides several web pages with L1 specific information (e.g. configuration data, GT rates and prescales, etc.).
- **Subsystem Cell.** The internal state of the subsystem processes and hardware can be accessed from a web browser [14].
- **TStore.** This XDAQ component provides read and write access to the Oracle DB through SOAP.
- **XMAS** (XDAQ Monitoring and Alarming Service). This is a high throughput publish/subscribe service to distribute monitorables and alarms [12, 13].

5. **Interconnection test service**
The Interconnection Test (IT) service is a stateful web service with the objective to allow for the creation and execution of hardware tests between and within L1 subsystems. This service is meant to be used during the whole experiment lifecycle in order to assure the quality of fixes and upgrades.

The IT architecture is similar to the one depicted in figure 3 except for five aspects. First, the IT is launched from the Central Cell and, currently, it is not meant to be a CMS-wide service. Second, there is not just one IT service, there are many (one per inter- or intra-subsystem test). Third, the IT activity diagram, and therefore its SOAP interface, is different from the one of the configuration service (see figure 6). Basically, there is a loop to exercise different hardware connection paths (potentially using different bit patterns for transmission), sparing the need to set up the test again for each of the paths/patterns (i.e. repeat the configure activity, which takes O(100) seconds). Fourth, the orchestration of subsystem activities is defined as part of the IT configuration data (i.e. it is not fixed like in the configuration service case). This is because there is a large amount of use cases with different time dependencies for each. Finally, the L1
emulators could be accessed as part of the setup transition in order to create the input and output patterns through a system call (i.e. calling a remote script via SSH).

6. Summary
The present paper describes the architecture of the L1CMS and the rationales behind its key characteristics. Diagrams and explanations of the decomposition and external interfaces of the major subsystem components have been presented. The authors have also made a deliberate attempt to visualize the information using UML. Surprisingly enough, this effort has also helped the authors to reduce the size of the paper. A posteriori, this seems obvious because the syntactic and semantic content of the diagrams and of the notation used can be found in the corresponding standards and manuals. The authors expect that such an attempt will simplify the understanding of the L1CMS system for newcomers, and its evolution. Hopefully, the attempt will also encourage the LHC and HEP community to improve its software documentation by using standard notations for its software systems.
References
[1] The CMS Collaboration 2008 The CMS experiment at the CERN LHC JINST 3 S08004
[2] The LCG TDR Editorial Board 2005 LHC Computing Grid CERN-LHCC 24
[3] The CMS Collaboration 2000 The TriDAS Project: The Trigger Systems CERN-LHCC 38
[4] The CMS Collaboration 2002 The TriDAS Project: Data Acquisition and High-Level Trigger CERN-LHCC 26
[5] I Magrans et al 2006 IEEE Trans. Nucl. Sci. 53 474-483
[6] A Petrucci et al 2007 The Run Control and Monitoring System of the CMS Experiment International Workshop on Advanced Computing and Analysis Techniques in Physics Research 11 Amsterdam
[7] R Wieringa 1998 ACM Computing Surveys 30 4
[8] D Pilone and N Pitman 2005 UML 2.0 in a Nutshell (O'Reilly Media)
[9] J Garland and R Anthony 2002 Large-Scale Software Architecture: A Practical Guide using UML (Wiley)
[10] V Innocente, L Silvestris and D Stickland 2000 CMS Software Architecture CMS Note 47
[11] W Badgett et al 2010 CMS Web-Based Monitoring IEEE Real Time Conference 17 Lisbon
[12] J Gutleber et al 2005 Hyperdaq, Where Data Acquisition Meets the Web International Conference on Accelerator and Large Experimental Physics Control Systems Geneva
[13] Bauer G et al 2009 Monitoring the CMS Data Acquisition System Proc. International Conference on Computing in High Energy and Nuclear Physics
[14] M Magrans de Abril et al 2010 Homogeneous User Interface Infrastructure for Expert Control of the Level-1 Trigger CMS Note 05