The CMS Trigger Supervisor Project

Ildefons Magrans de Abril, Member, IEEE, Marc Magrans de Abril

***Abstract*–The Trigger Supervisor (TS) is an online software system. Its purpose is to set up, test, operate and monitor the Level-1 trigger components on one hand and to manage their interplay and the information exchange with the run control and monitoring part of the data acquisition system on the other. Due to the large number of subsystem interfaces, the fact that a malfunctioning can result in significant economic losses and hardware physical damage, the Trigger Supervisor must be considered a critical system in the context of the CMS experiment. This paper presents the main components of the TS project that helps to cope with the complexity of this software system.**

I. Introduction

THE experiment CMS (Compact Muon Solenoid) at the Large Hadron Collider (LHC) of CERN, the European Organization for Nuclear Research in Geneva, is a detector designed to find answers to some of the fundamental open questions in physics today [1]. The trigger and the data acquisition are vital components of the experiment. The two systems are responsible for the selection and recording of collision events. Since millions of collisions occur each second and only a small fraction of these can provide insight into new physics, events have to be selected online according to their properties. The trigger system is composed of subsystems and is organized in two basic levels. The Level-1 Trigger (L1T) [2] consists of custom developed and largely programmable electronics, the High Level Trigger (HLT) is a large computer farm [3]. Apart from recording events, the data acquisition system (DAQ) [3] also provides the Run Control and Monitoring System (RCMS) framework [4] for the experiment.

The Trigger Supervisor

The Trigger Supervisor (TS) is an online software system. Its purpose is to set up, test, operate and monitor the Level-1 trigger components on one hand and to manage their interplay and the information exchange with the run control and monitoring part of the data acquisition system on the other.

Other collaboration parties are also involved in this project: sub detectors, the luminosity monitoring system, the HLT, the Online Software Working Group (OSWG), the DataBase Working Group (DBWG). A consistent configuration of the trigger primitive generator modules of each sub detector, the automatic update of the trigger pre scales as a function of information obtained from the luminosity monitoring system, the adequate configuration of the HLT and the agreement in the usage of software tools and database technologies enlarge the number of involved parties in the design of the TS.

In addition, we have to cope with the complexity arising from the large periods of preparation and operation of high-energy physics experiments. During this time the hardware and software environments evolve.

The TS Project must cope with these complexity parameters. The elements of this project are: a team of system engineers, a conceptual document, a development process, a software framework, a software system based on this framework, a specifications document consistent with the current state of the TS and a discussion working group that involve all concerned parties. This paper describes the different elements of the TS project.

II. The Trigger Supervisor Development Process

Due to the large number of subsystem interfaces, the fact that a malfunctioning can result in significant economic losses and hardware physical damage, the Trigger Supervisor must be considered a critical system in the context of the CMS experiment. Generally speaking, a critical system in high energy physics experiments with particle accelerators [5] requires a well defined development process. In the case of the Trigger Supervisor, a suitable methodology should:

- Help to understand the requirements coming from a large number of groups of people working in different subsystems.
- Include a development plan. This would help to properly define the human resources requirements in every stage of the project and would be used as a reference for other subsystems in order to prepare their own plans.
- Help to avoid mistakes and to detect and remove errors by applying a detailed software design.
- Limit the damage caused by operational failures when the system is in use.

We believe that a development process based on the spiral model created originally by Boehn [6] and adequately adapted to the TS project specificities would suit our demands. Fig. 1 shows the TS development process.

In the initial state of the development process, the TS team, defined the problem and identified the major guidelines of a solution. This first concept evolved in a second step to become a document which specified the functional and non-functional requirements of the Trigger Supervisor, and its design and operational details. This document was a starting point for discussion with all involved parties. A final version of this document [7] was endorsed by the CMS Trigger and DAQ groups. This document was used as a starting point for the software development cycle and serves as a reference for future discussions.

Manuscript received November 10, 2005.

Ildefons Magrans de Abril (E-mail: ildefons.magrans@cern.ch) is with the Institute for High Energy Physics of the Austrian Academy of Sciences, Vienna, Austria, and also with the Electronic Engineering Department, Universidad Autónoma de Barcelona, Barcelona, Spain.

Marc Magrans de Abril (E-mail: marc.magrans.de.abril@cern.ch) is with the University of Wisconsin.

The TS software development cycle is an iterative process with three main steps: 1) design and development of a software framework including the database infrastructure, 2) trigger subsystems and sub detectors integration into the TS software framework, and finally, 3) the system deployment, release creation and test. During this third step the framework and the TS system are evaluated and evolution guidelines are proposed. The second step also includes maintaining a document that describes the current state of the TS. This document is used by the TS team as a communication channel with all involved parties to communicate and discuss the evolution of the system.

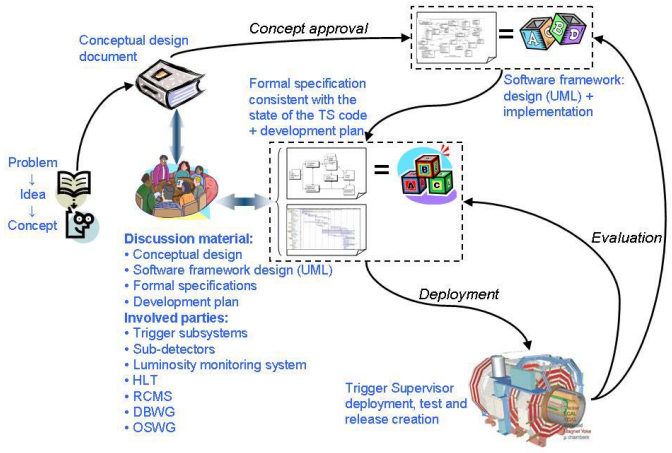

Fig. 1. The Trigger Supervisor development process.

III. Conceptual Design

The TS architecture [7] is composed of a central node in charge of coordinating the access to the different subsystems and a customizable TS leaf for each of them that will offer the central node a well defined interface to operate the online software infrastructure of each subsystem. Fig. 2 shows the architecture of the TS.

Each node of the TS can be accessed independently. The available interfaces and location for each of those nodes are defined in a Web Service Description Language document (WSDL, [8]). Both the central node and the TS leaves are based on a single common building block, the Control Cell. Each subsystem group will be responsible for customizing a Control Cell, and to keep the consistency of the available interface with the interface described in the corresponding WSDL file. The presented design is not driven by the available interface of the online software infrastructure (OSWI) of a concrete subsystem, and is logically and technologically decoupled from any controller. Therefore, this will improve the evolution potential of the low-level infrastructure and the TS.

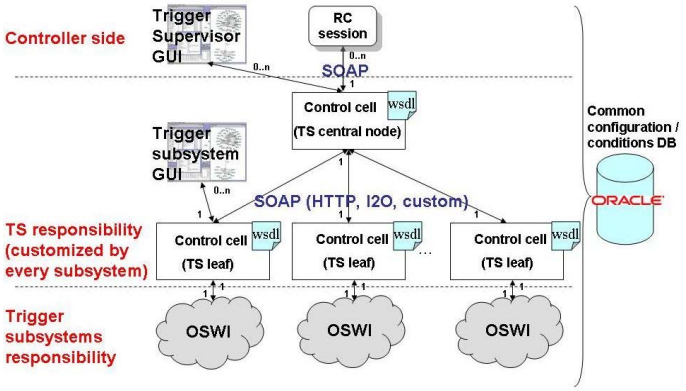

Fig. 2. Architecture of the Trigger Supervisor. A central node and customizable leaves for each subsystem are the core elements.

A. The Control Cell

The architecture of the TS is characterized by its tree topology, where all tree nodes are based on a common building block, the Control Cell. Each cell can work independently of the rest, or inside a more complex topology. The Control Cell is a program that offers the necessary functionalities to coordinate the control operations over other software systems, for instance the OSWI of a concrete trigger subsystem, an information server, or even another control cell. The basic idea is that a set of available operations exist that can be accessed by a given controller. Each operation corresponds to a finite state machine (FSM). The default set of operations is customizable and extensible. The following points describe the components of the control cell:

Control Cell Interface (CCI): This is the external interface of the control cell. Different protocols will be available. An HTTP [9] interface will facilitate a first entry point from any web browser. A second interface based on SOAP [10] will also be provided in order to ease the integration of the TS with the run control or any other controller that requires a web service interface.

Access Control Module (ACM): Each module is responsible for identifying and authenticating every user or entity (controller) attempting to access, and for providing an authorization protocol. The access control module will have access to a user list, which will provide the necessary information to identify and authenticate, and the privileges assigned to each controller. Those privileges are used to check whether or not an authenticated controller is allowed to execute a given operation.

Task Scheduler Module (TSM): This module is in charge of managing the command requests and forwarding the answer messages. The TSM is also responsible for preventing the launching of operations that could enter into conflict with other running operations (e.g., simultaneous self test or configuration operations within the same trigger subsystem, interconnection test operations that cannot be parallelized).

Shared Resources Manager (SRM): This module is in charge of coordinating access to shared resources (e.g., the configuration data base, other control cells, or a trigger subsystem online software infrastructure).

Error Manager (ERM): This module will provide the management of all errors not solved locally, which have been generated in the context of the control cell, and also the management of those errors that could not be resolved in a control cell immediately controlled by this one. Both the error format and the remote error notification mechanism will be based on the global CMS distributed error handling scheme.

B. *The Trigger Supervisor Services*

The Trigger Supervisor Services are the final functionalities offered by the TS. These services emerge from the collaboration of several nodes of the TS tree. In general, the central node will always be involved in all services coordinating the operation of the necessary TS leaves. The following services have been required:

TS start-up: This service facilitates the starting up of the whole L1 Trigger software infrastructure including: the TS and the corresponding online software infrastructure of each subsystem.

Configuration: This service is intended to facilitate the hardware configuration of the trigger system, which includes the setting of registers or look-up tables and downloading the trigger logic into the FPGA's of the electronics boards. The configuration service requires the collaboration of the central node of the TS and all the TS leafs.

Self test: This service is intended to check that each individual subsystem is able to operate as foreseen. If anything fails during the test of a given subsystem, an error report is returned, which can be used to define the necessary corrective actions. The self test service can involve one or more subsystems.

Interconnection test: This service is intended to check the connections among subsystems. In each test, several trigger subsystems and subdetectors can participate as Sender/s or Receiver/s. The interconnection test service requires the collaboration of the central node of the TS and some of the TS leaves.

Monitoring: The monitoring service will be implemented by an operation running in a concrete TS leaf or as a collaborative service where an operation, running in the central node of the TS, is monitoring the monitoring operations running in a number of TS leafs.

IV. Subsystem Package Deliverable

The development of the TS requires the distribution of its implementation among the trigger subsystems and the TS team. In order to facilitate the distributed development a package is delivered to every subsystem.

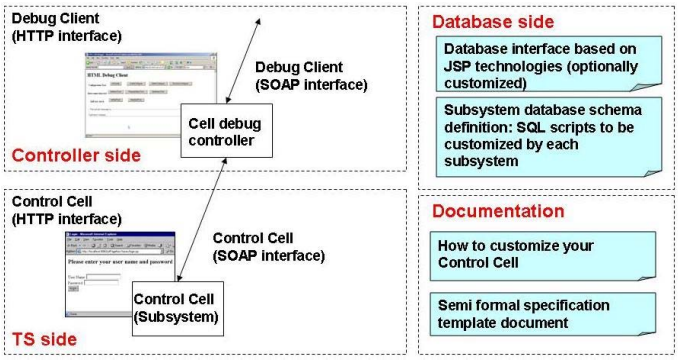

Fig. 4. Subsystem package contains: The Control Cell skeleton, a control cell controller to debug the remote interface, a database interface to access the corresponding configuration database, SQL scripts and necessary documentation to create the configuration data base, a 'howto' guide describing the Control Cell customization process and a template document that tells how to properly document the customization decisions.

The subsystem development package, shown in Fig. 4, includes the following items:

Control Cell: This application allows to multiple users to monitor, and to initialize and control operations remotely using a SOAP or an HTTP interface. The susbsystem responsible will be in charge to adapt the FSM for each operation (i.e. Configuration, Selftest. Interconnection test, Start up), and to add the necessary functionality to the state transition functions. The Control Cell interface can also be enlarged with additional operation types and monolithic commands.

Debug Client: This application is provided to facilitate the development and debug process of the Control Cell, and to certify the SOAP interface compatibility.

Database Schema: A script that creates the database schema to store the hardware configuration information for a generic trigger subsystem is provided. The Level-1 Trigger Configuration database is the final result of creating the different subsystem configuration databases.

Graphical User Interface: The Control Cell will provide an HTTP interface that will allow to fully operate the Control Cell and to visualize the state of any running operation. This can also be customized to adopt a specific look'n'feel and/or to add new functionalities.

Documentation: the necessary documentation is also provided with the software framework that explains how to modify and execute the SQL scripts, a 'howto' guide describing the Control Cell customization process and a template document that tells how to properly document the customization decisions (see section below).

V. Communication Channels

To set up adequate communication channels between the TS team and the rest of involved parties is a key stone in the project development process. One of the problems in defining this communication channels is that they may concern different classes of consumers having fairly different background and language – electronic engineers, physicists and programmers. Consumers can be roughly divided between the TS team and the rest. For internal use, the TS members use the Unified

Modeling Language (UML) [11] descriptions to model and document the status of the TS software framework: concurrency, communication mechanism, access control, task scheduling and error management. This model is kept consistent with the status of the TS software framework. This additional effort is worth because accelerates the learning curve of new team members that are able contribute effectively to the project in a shorter period of time, helps to detect and remove errors and can be used as discussion material with other software experts, for instance to discuss the data base interface with the DBWG or to justify to the OSWG an upgrade in a core library. But this approach is no longer valid when the consumer is not a software expert. Project managers, electronic engineer or physicist must also contribute. A semi-formal description approach based on Finite State machines and flow diagrams has been chosen. Each node of the TS is modeled as a black box that offers a set of services that can be either monolithic commands or finite state machines. In both cases the functionality is defined with flow diagrams.

This simple approach has facilitated the understanding of the TS to a wide range of experts and has helped in the continuous process of requirements understanding. A practical way to communicate the status of the project has also facilitated the maintenance of a realistic development plan and man power prevision calendar.

VI. Demonstrator

In this section we outline two demonstrators of the TS. The first is demonstrator is the integration of three different trigger subsystems: the Global Trigger (GT [12]), the Global Muon Trigger (GMT [2]) and the Regional Calorimeter Trigger (RCT [2]). The second one is an example of interconnection test service involving the Hadron Calorimeter (HCAL [13]), the Electromagnetic Calorimeter (ECAL [14]) and the RCT.

A. Configuration Service Demonstrator

This first demonstrator is a distributed system with four nodes or Control Cells: the central node that serves as entry point, and one additional node per trigger subsystem. It is interesting to note that the TS nodes are running in three different countries: Switzerland, France and Austria. A high level diagram of the system is shown in Fig. 5.

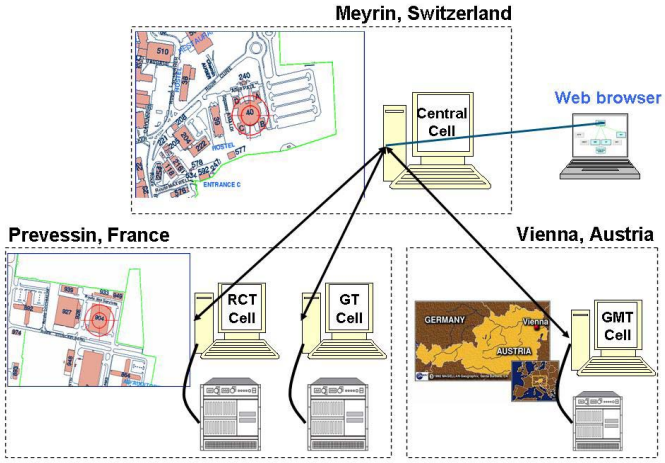

Fig. 5. Overview of the Trigger Supervisor demonstrator.

This demonstrator features a single access point reachable using a web browser which offers three levels of control: The lowest level of access control allows controlling a concrete board; the second level offers a high level interface per each trigger subsystem which simplifies the configuration and test procedures; finally, the highest level of access facilitates the coordination of the configuration and test over all trigger subsystems. The implementation of the system took 2 weeks to 4 persons working full time.

B. Interconnection Test Demonstrator

Due to the large number of communication channels between the trigger primitive generator (TPG) modules of the subdetectors and the trigger system, and between the different trigger subsystems, it is necessary to provide an automatic testing mechanism. The interconnection test service of the Trigger Supervisor is intended to automatically check the connections between these subsystems. Fig. 6 shows an interconnection test scenario intended to check the communication channels between the TPG modules of the HCAL and ECAL in one side and the RCT on the other side.

The interconnection test service requires the collaboration of the central node and the cells of every involved subsystem. Each subsystem cell offers to the central node cell two remote methods: *prepare_test(test_id)* and *start_test*:

Prepare_test(test_id): This method configures the hardware as a function of the *test_id* parameter and waits for the *start_test*. As a function of the *test_id*, each subsystem knows whether they have to behave as a sender or as a receiver. The hardware access in the case of the subdetectors goes through the subdetector control module, which is another web servive based distributed application. On the other hand, the hardware access for the RCT is performed locally using a C++ API. The central node of the TS is in charge to coordinate configuration and start up of every involved susbsystem.

Start_test: This method has a different behavior depending whether the subsystem is a sender, receiver or the central node. The *start_test* method in the central node is in charge to configure the synchronization modules of the HCAL and ECAL to assure that the RCT input modules will receive the

data at the same time. The *start_test* method in the RCT cell waits that the input buffer reaches 128, compare the received data against the expected one, and sends a summarized report to the central node.

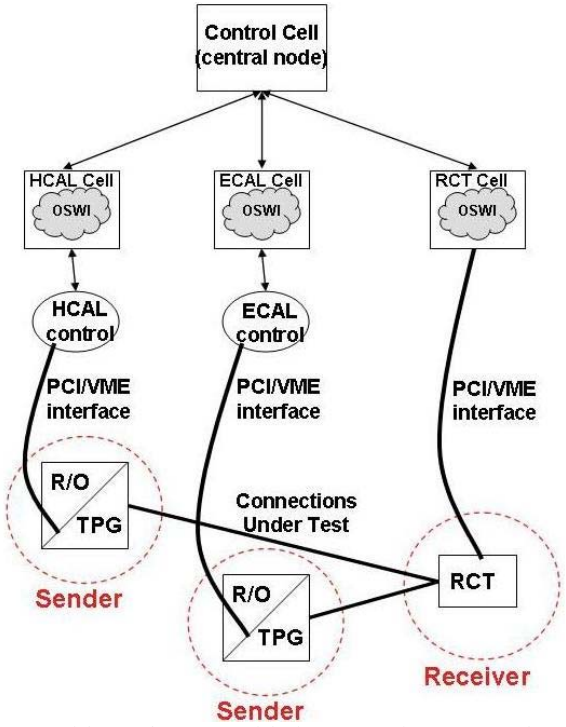

Fig. 6. Diagram of the HW/SW setup of the interconnection test example.

VII. Software Configuration Management

The TS and the online software infrastructure of the trigger subsystems will evolve during the development phase, but also during the installation, the commissioning and the operational phases. An accurate and centralized control over the evolution of the system configuration has the following advantages:

- To facilitate the exact replication of the trigger system.
- To improve the communication of the software evolution.
- To facilitate the coordination among trigger subsystem groups.
- To ease the resource sharing among trigger subsystem groups.

As a part of the strategy for a proper management of the level-1 trigger online software infrastructure, we have adopted the following actions:

- Common CVS repository for all trigger subsystems.
- Generic Makefile for all related software projects.
- Web page of the TS project [15] from were it can be downloaded the last releases of the subsystem package and the TS demonstrator. This web page also contains all the documentation: internal collaboration presentations, conference reports, conceptual design, UML diagrams and semi formal specifications.

VIII. Summary

The Trigger Supervisor is an online software system designed for the CMS experiment at CERN. Its purpose is to provide a framework to set up, test, operate and monitor the trigger components on one hand and to manage their interplay and the information exchange with the run control part of the data acquisition system on the other. Due to the large number of subsystem interfaces, the fact that a malfunctioning can result in significant economic losses and hardware physical damage, the Trigger Supervisor must be considered a critical system in the context of the CMS experiment. Generally speaking, a critical system in high energy physics experiments with particle accelerators requires a well defined development process. Important points of this development process are: the agreement of a conceptual design document, the design and development of the TS framework, the adoption of adequate communication channels between the TS team and the rest of involved parties and the proposal of a minimum set of configuration management actions to be followed by the TS team and all trigger subsystems. Finally, a demonstrator of the Trigger Supervisor has been presented.

References

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

[2] The CMS Collaboration, The Trigger and Data Acquisition Project, Vol. I: The Level-1 Trigger, CERN LHCC 2000-038 (2000).

[3] The CMS Collaboration, The Trigger and Data Acquisition Project, Vol. II: Data Acquisition and High-Level Trigger, CERN LHCC 2002-26 (2002).

[4] V. Brigljevic et al., "Run Control and Monitor System for the CMS Experiment", Computing in High Energy and Nuclear Physics, 24-28 March 2003, La Jolla, California.

[5] E. Gaytan, T. Rivero, J. Palacios, J. Ortiz, J. Segovia, "Is the software engineering essential in the nuclear sciences?", Nuclear Science Symposium Conference Record, 2004 IEEE Volume 3, Date: 16-22 Oct. 2004, Pages: 1877 – 1880.

[6] B. Boehm, "Spiral development: experience, principles and refinements", Software Engineering Institute SEI-CSE Workshop in Spiral Development. Special Report CMU/SEI-00-SR-08, June, 2000.

[7] CMS Note 2005/011, I. Magrans de Abril, C.-E. Wulz, J. Varela, "Concept of the CMS Trigger Supervisor".

[8] Web Services Description Language [online]: http://www.w3.org/TR/wsdl

[9] Hypertext Transfer Protocol, [Online]: http://www.w3.org/Protocols/HTTP/

[10] Simple Object Access Protocol (SOAP) 1.1, [online]: http://www.w3.org/TR/soap

[11] Unified Modeling Language [online]: http://www.rational.com/uml/

[12] C.-E. Wulz, "Concept of the First Level Global Trigger for the CMS experiment at LHC", Nucl. Instr. and Meth. A473 (2001) 231 – 242.

[13] CMS Collaboration, "The Hadronic Calorimeter Project, Technical Design Report", CERN/LHCC 97-31, CMS TDR 2, 20 June 1997.

[14] CMS Collaboration, "The Electromagnetic Calorimeter Project, Technical Design Report", CERN/LHCC 97-33, CMS TDR 4, 15 December 1997.

[15] Trigger Supervisor Project Web Page [Online]: http://triggersupervisor.cern.ch