AUTOMATED EXECUTION AND TRACKING OF THE LHC COMMISSIONING TESTS

K. Fuchsberger, V. Baggiolini, M. Galetzka, R. Gorbonosov, M. Pojer,
M. Solfaroli Camillocci, M. Zerlauth, CERN, Geneva, Switzerland

Abstract

To ensure the correct operation and prevent system failures, which can lead to equipment damage in the worst case, all critical systems in the Large Hadron Collider (LHC), among them the superconducting circuits, have to be tested thoroughly during dedicated commissioning phases after each intervention. In view of the around 7,000 individual tests to be performed each year after a Christmas stop, a lot of effort was already put into the automation of these tests at the beginning of LHC hardware commissioning in 2005, to assure the dependable execution and analysis of these tests. To further increase the productivity during the commissioning campaigns and to enforce a more consistent workflow, the development of a dedicated testing framework was launched. This new framework is designed to schedule and track the automated tests for all systems of the LHC and will also be extendable, e.g., to beam commissioning tests. This is achieved by re-using different, already existing execution frameworks. In this paper, we outline the motivation for this new framework and the related improvements in the commissioning process. Further, we sketch its design and present first experience from the re-commissioning campaign in early 2012.

MOTIVATION

About 7,000 individual tests have to be performed each year after a long maintenance stop around Christmas, to ensure proper functionality of the superconducting circuits of the LHC. In view of this large amount of tests, a lot of effort was put in automation of tests, starting from the beginning of hardware commissioning in 2005 [1]. The status in 2011 was the following:

- The test procedures for all the superconducting circuits were formulated in so-called 'sequences' which can be executed by a dedicated tool, the Hardware Commissioning Sequencer (HWC Sequencer) [2].
- Dedicated Analysis tools (developed in LabView®) were provided to verify the data resulting from the tests. Some of them were automatized.
- The results of the tests could be displayed through a web interface (written in PHP).

The whole system had been grown over time and involved a lot of individual systems, which were loosely coupled together and based on different technologies and sometimes hard to maintain. This is why it was decided, in summer 2011, to review the whole testing system and streamline the developments in one direction.

ACCTESTING FRAMEWORK

The initial focus of the new framework was the execution and tracking of tests for LHC hardware commissioning. Nevertheless, it soon turned out that a more general approach was appropriate. The goal was then to create a general framework for the execution and tracking of tests for accelerator systems ('AccTesting' in the following). The framework must be able to deal with a high workload and enable its users to work in parallel. Furthermore, it must prevent execution conflicts and provide the current test status information to all of its users.

Overview

A general overview over the architecture of the framework can be seen in Fig. 1. The central point is the AccTesting server. The test execution and analysis results are stored in a database that only the server may access. The server itself is not aware of any specifics of the tests it handles. The test execution servers and the result analysis

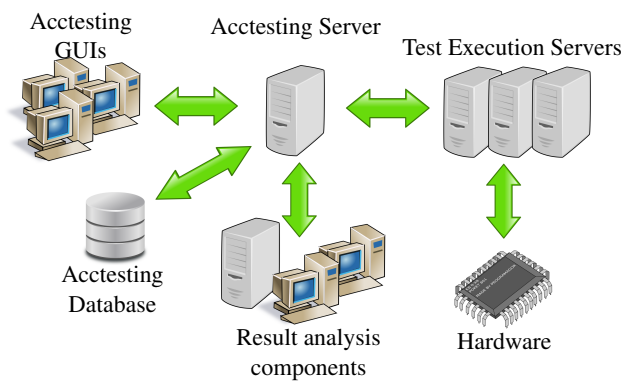

Figure 1: Components of the AccTesting framework.

components are connected to the server with a plug-in like system. Each of them can handle a specific type of tests. If the main server wants to start the execution or analysis of a test, it provides each of the plugged-in test handlers with the test information, which then in turn can decide if they are able to handle the test. Once a test handler has accepted a test and started the execution or analysis, the main server will regularly poll it to retrieve the test status and result.

The AccTesting server is controlled by several users through the use of a specific Graphical User Interface (GUI). The AccTesting GUI displays all the information about the currently executing tests and scheduled tests. In this sense it replaces the former test tracking web pages. Furthermore, it allows to enqueue a scheduling request to the AccTesting server directly from within the test plan view. An example screenshot of the GUI is shown in Fig. 2.

ISBN 978-3-95450-115-1

Figure 2: A screenshot of the graphical user interface (GUI) for the AccTesting framework.

The AccTesting main server must be very robust. It has to deal with unexpected behavior from its plugged-in test handlers, errors in the control GUI, incomplete test results and many other issues like a sudden crash of the virtual machine. Several mechanisms have been implemented to deal with these situations:

- The server itself constantly serializes all important data (which are not persisted in the database), as well as all of its test handlers, and writes the serialized information on the hard disk. In the case of a sudden server crash, it will read these serialized data and is able to start right where it stopped. Furthermore, this is a great help when the server has to be restarted after applying a patch, because it can be done in the middle of the work.
- Unexpected behavior from any test handler component results in the immediate abortion of the test. The aborted tests are collected in a special pool from where they can be easily restarted.
- Incomplete or inconsistent results are a problem if not treated correctly, because they are used to determine the status of a test and therefore its next steps in the AccTesting framework. In the worst case, a failed test with inconsistent results could be labeled successful and allow the execution of further tests. To overcome this problem, the status of each test is determined by a complex finite state machine that uses all available information about the test every time.
- Failures in the GUI, or even malicious hacking attempts, are not easy to detect, but the main server checks all the received data for consistency with its own database and prevents the execution of any commands if there is a mismatch.

Scheduling

The scheduling algorithm in the server is responsible for executing the enqueued tests in the most efficient way, while respecting all the constraints and preconditions. It has to be fast, reliable and easy to understand. The current implementation of the scheduling algorithm tries to find an arbitrary combination of tests that can be executed right away. Although the algorithm is rather fast, as it requires $\mathcal{O}(n)$ memory and runs in $\mathcal{O}(n^2)$ time, it is unable to plan ahead and chooses the tests to execute arbitrarily. Therefore, it might create suboptimal schedules. A better scheduling algorithm, without these shortcomings, is currently under development. A promising candidate is an algorithm based on heuristic repair. Other alternatives could be approaches based on tree search or ant colony optimization.

Advantages

The new framework is an improvement to the old architecture in the following ways:

- It provides a way for users to work in parallel on the tested systems. Before, this involved a lot of coordination where each user was only allowed to work on a specific working-set of systems.
- It is a central point for all the test related data. Before, all the systems directly accessed the database and communicated with each other, leading to a tight coupling of these systems.
- The system is easily extensible without any need to change the database or existing code.
- It features an automated scheduling which is able to automatically consider even complex test constraints. Before, each user had to have these constraints in mind when scheduling a test, what often resulted in failed tests.

SOME STATISTICS

Figure 3 shows the number of started hardware commissioning tests per day throughout the years 2008 to 2012.

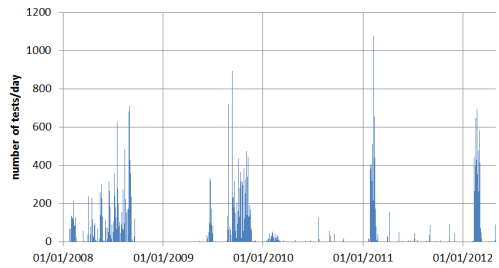

Figure 3: Executed tests per day between 01 Jan 2008 and 30 April 2012.

The initial tests were widely distributed over most of the year in 2008 and also a long commissioning campaign took place after the repairs in 2009. Two smaller campaigns were performed after the Christmas shutdowns in the beginning of 2011 and 2012 [3]. For the following analysis we consider the time ranges of the individual campaigns as defined in Table 1. The table also summarizes the number of shifts per campaign. HWC shifts were identified from the data as LHC-shift periods in which at least 5 tests were executed.

Table 2 shows some statistics about tests executed during the different hardware commissioning campaigns. According to the numbers in the table, surprisingly the percentage

Table 1: Summary of main LHC hardware commissioning campaigns between 2008 and 2012.

Name	Start Date	End Date	#Shifts
HWC 2008	2008-01-01	2008-09-20	225
HWC 2009	2009-06-01	2009-11-19	175
X-Mas 10/11	2011-01-01	2011-03-15	48
X-Mas 11/12	2012-01-01	2012-03-15	43

of failed tests stayed constant w.r.t. last year; the number of tests which failed because of a timeout during the initialization of the test sequence could be reduced by a factor of 10. This kind of failures are mostly provoked, if a test on a circuit is started, when the equipment is not ready (e.g. locked, or QPS not ready). Such situations were eliminated this year, because the new scheduling algorithm respects all these constraints and will never run a test on a locked circuit, for example. It was expected that also the average time between executions of different tests on the same system would be drastically reduced, because the scheduler would automatically start new tests as soon as the preconditions are fulfilled. This assumption could not be confirmed by the actual statistics data. One major cause for this inconclusive result is, that the time between tests on one circuit turns out to be mainly dominated by the analysis time.

Table 2: Test statistics for different HWC Campaigns ('#T/#S': Average number of tests per shift; '% FTO': percentage of failed tests that timed out).

Campaign	#Tests	#T/#S	% failed	% FTO
HWC 2008	17377	76.7	19.8	21.3
HWC 2009	15186	86.4	19.2	40.6
X-Mas 10/11	6977	144.8	16.7	47.8
X-Mas 11/12	6856	158.9	16.6	4.9

In the following, only data from the 2012 campaign is used, since this was the first time, when the end time of the analysis was also stored in the database. The average execution- and analysis-times for several test types are plotted in Fig. 4, together with the respective average execution times. Hereby, the execution time is considered as the time between start of the test sequence and termination of the test sequence. The analysis time is considered to be the time between the termination of the sequence and the reception of the analysis result.

For some of the test types, automatic analysis modules are already available (All PIC2 tests and PNO.a1) which shorten the analysis time drastically. Nevertheless, if the automatic analysis considers the test as failed, then the test has to be analyzed manually. This is one reason why for the PIC2 tests and PNO.a1 the analysis of successful tests is much faster than for failed tests. For other tests (e.g. PNO.d1/c2/a7), the analysis times for successful and failed tests are very similar. This already shows the necessity for more automatic analysis, which can be emphasized by looking at the total times spent in analysis for the different

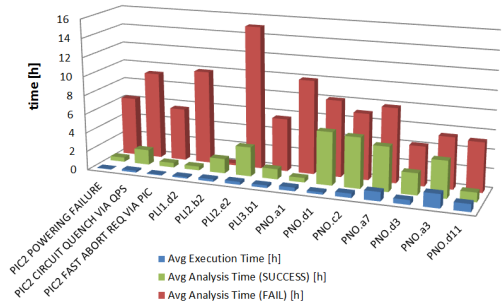

Figure 4: Average analysis times.

test types, as shown in Fig. 5. Again the PNO tests stick out here, except the automated PNO.a1, whose analysis time is already similar to the execution time.

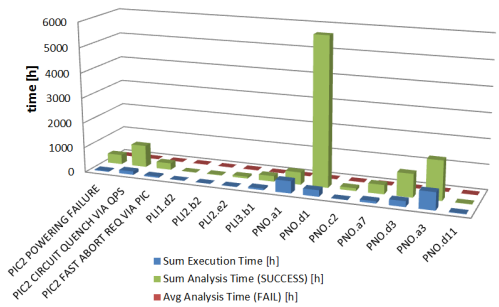

Figure 5: Total time spent in analysis per test type.

SUMMARY AND OUTLOOK

The execution for LHC hardware commissioning tests was reviewed in 2011 and the implementation of a general framework for the execution and tracking of tests for accelerator systems was started. The system consists of a central server, which orchestrates the testing process and is easily extensible by plugin-in like modules. This framework takes care of respecting all preconditions and constraints, which had to be kept in mind by the users before.

Next steps for this framework will include a streamlining of the database design, further improvement of the scheduling algorithm, the implementation of more automatic test analysis modules and the possibility to easily edit test plans.

REFERENCES

[1] B. Bellesia et al., "Information Management within the LHC Hardware Commissioning Project", proc. of PAC09, Vancouver, BC, Canada.

[2] V. Baggiolini et al., "A Sequencer for the LHC Era", proc. of ICALEPCS2009, Kobe, Japan.

[3] M. Pojer et al., "Studies on the LHC Superconducting Circuits and Routine Qualification of Their Functionalities", these proceedings.

ISBN 978-3-95450-115-1