THE AccTesting FRAMEWORK: AN EXTENSIBLE FRAMEWORK FOR ACCELERATOR COMMISSIONING AND SYSTEMATIC TESTING

D. Anderson, M. Audrain, K. Fuchsberger, J.C. Garnier, R. Gorbonosov, A.A. Gorzawski, A. Jalal, A. Moscatelli, P.C. Turcu, K. Stamos, M. Zerlauth,
CERN, Geneva, Switzerland

Abstract

The Large Hadron Collider (LHC) at CERN requires many systems to work closely together to allow reliable operation and at the same time ensure the correct functioning of the protection systems required when operating with large energies stored in magnet systems and particle beams. The systems for magnet powering and beam operation are qualified during dedicated commissioning periods and retested after corrective or regular maintenance. Based on the experience acquired with the initial commissioning campaigns of the LHC magnet powering system, a framework was developed to orchestrate the thousands of tests for electrical circuits and other systems of the LHC. The framework was carefully designed to be extendable. Currently, work is on-going to prepare and extend the framework for the re-commissioning of the machine protection systems at the end of 2014 after the LHC Long Shutdown. This paper describes the concept, current functionality and vision of this framework to cope with the required dependability of test execution and analysis.

MOTIVATION

The number of tests executed at CERN to ensure the proper functionality of the superconducting circuits in the LHC is increasing year by year. Just last year about 7000 individual tests were performed with the help of the *AccTesting framework* ("AccTesting" in the following) which is the central tool to manage this important process. Even if the initial goal of this framework was to take care of the LHC hardware commissioning, it was designed from the beginning to be able to execute and track tests for any kind of accelerator system. After a short Overview of the design of the framework, this paper will focus on the work which was done since the framework was introduced for the first time [1] and the ongoing work, which has to cover the upcoming challenges. The most important challenges are:

- Full Automation of the Test Analysis.
- Integration of tests for the commissioning of the LHC machine protection systems.

All this is necessary to prepare the framework for a smooth workflow during the restart of the LHC in 2014.

DESIGN OVERVIEW

Managing a large number of tests must take into account the scheduling of parallel sessions between different users. For this reason AccTesting has been structured to centralize the administration of the tests in the *AccTesting server* which is the only one able to access the database where all the test execution and the analysis results are stored. The AccTesting server orchestrates the whole testing workflow by providing the schedule of the different tests requested from the users and avoiding execution conflicts between them. An overview of the architecture is shown in Fig. 1.

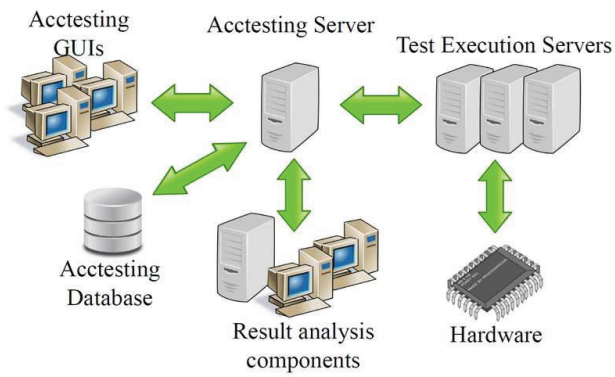

Figure 1: The components of the AccTesting framework.

This centralization allows the users to have a global view of the executed and scheduled tests at any time. An example of a test plan overview is shown in Fig. 2. A screenshot of an example test schedule which is produced by the centralized scheduler [2] is shown in Fig. 3.

Figure 2: The GUI to start tests and monitor their status.

The main building blocks of a test plan within AccTesting are illustrated in Fig. 4. In short, they are:

ISBN 978-3-95450-139-7

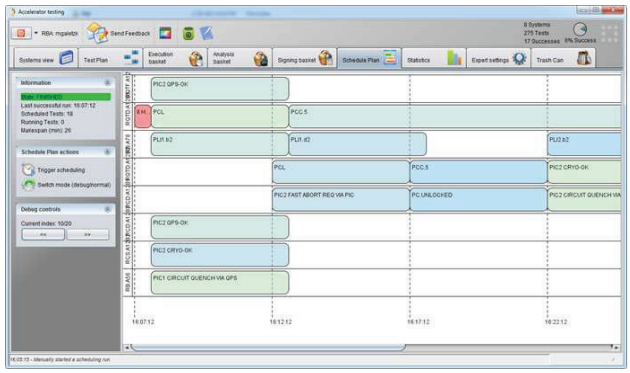

Figure 3: The schedule of the tests provided by the server.

- **Tests:** The core block is a test which represents certain actions taken by hardware or software. The same test can be executed on different systems and system types.
- **Test phases:** Each test belongs to exactly one test phase. The test phases group the tests together and depend on each other. This also defines the execution order of blocks of tests: It is not possible to start any test of a phase which depends on a phase where not all tests are completed successfully.
- **Test steps:** Each test has three test steps: Execution, Analysis and Signing. The Execution step represents the moment where something is actually done on the device in order to test it. If everything goes well in execution and the system does not receive any kind of blockers (crashes, errors, etc. . .) the Analysis step can start analyzing the results (the signals of the system monitored during the execution step). Finally there is the Signing step where, depending on the kind of test executed, one or more experts have to validate the outcome of the previous steps by signing with their name and role.

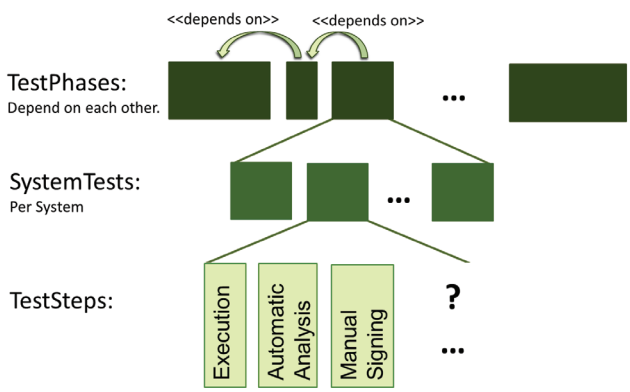

Figure 4: The designed structure of the test phases, tests, test steps.

Extension Points

The core design of the AccTesting framework is open for extension without the necessity to change critical components. The most relevant extension points (mostly represented by Java interfaces, that can be simply wired into the system) are the following:

- `TestStepHandler`: They determine what exactly will be the action to execute for a certain type of test during a test step. All these handlers have a simple `canHandle(..)` interface, which allows the server to find an appropriate handler for a certain test. As soon as a matching handler is found, the test is run on this handler. In analogy to the previously described possible test steps (execution, analysis, signing), there are three subtypes of test step handlers:
  - `TestExecutionHandler`: Implementations of this type specify the behavior when a certain type of test is executed. For example, the handler for testing LHC circuits uses another server (the so-called sequencer) to do the real testing work (driving currents, switching on/off equipment etc.).
  - `TestAnalysisHandler`: This type of handler takes care of analyzing data logged from the execution of tests. In previous years, there was only one handler, that was communicating with the old LabView programs to analyze the test data.
  - `TestSigningHandler`: This has only a single instance and will stay like this for the near future, which allows signing from clients of the server (GUIs).
- `Constraints`: These classes have a slim interface which basically allows the framework to query, if a given pair of tests is allowed to be executed in a given configuration. All enabled constraints are respected by the scheduler in the server while generating the test schedule. They can be enabled and disabled through the AccTesting GUI and can even be dynamically reloaded after some behavioral change, without the need to restart the server.
- `LockProvider`: A system can have different active locks. For example, LHC circuits can be locked at the hardware level, or only for test-execution with a simple flag in the database. This extension point is foreseen to plug further types for other systems into the framework.
- `SystemProvider`: These plug-ins allow us to feed different types of systems into the framework. Systems are only identified by a unique key, which is kept in the AccTesting database for reference. All other information about the systems is read at runtime from the system providers. In the meantime, this mechanism was extracted to a separate project [3] as will be described later.

ISBN 978-3-95450-139-7

- `SystemInformationProvider`: Allows us to transport other information of a system (e.g. Issues, Status) to the framework, mainly to display it in the GUI.

- `TestProvider`: These extension points provide tests to the system. It also has to provide the information on which systems a certain test could be executed. Again, the framework only keeps entries of the unique keys in a table, but loads all the remaining information from the test providers.

- `TestResultsViewer`: This is the only extension point for the AccTesting GUI so far. The framework queries its interface at the time the result of an executed test shall be displayed and collects all the viewers which are returned by all the providers and displays them in different tabs in the respective dialog. An example for this is the post mortem viewer, which is plugged into AccTesting to show raw data produced during the tests.

AUTOMATED ANALYSIS

After putting AccTesting in place in 2011, the duration of the Execution steps was brought to a reasonable minimum by eliminating human errors on triggering wrong sequences or triggering correct tests while not all conditions were fulfilled. The next action was targeted at the Analysis steps, to reduce (where possible) the time spent on analysis. As mentioned in the previous section, only one analysis handler was in place before. This handler was communicating with dedicated LabView applications to perform the analysis of signals produced by LHC circuit tests.

To find a more consistent approach, a new project was launched with the aim to provide a better way to analyze the data. Concerning the extension of the AccTesting framework, it is simply a new instance of an analysis handler. The concepts behind are the following:

- A dedicated analysis module can be scripted for each test type within the AccTesting framework, which describes which checks (assertions) have to be performed on the data. To simplify this task and make it possible for system experts (non-programmers) to write such modules, a dedicated assertion language was created, a so-called Java embedded Domain Specific Language (eDSL) [4].

- Using the checks formulated in these modules, the new analysis framework [5] can perform the necessary calculations and checks.

- Additionally, these analysis modules can also be seen as a machine-readable version of a test specification. Up to now, these specifications (test procedures) were always formulated in text documents. Nevertheless, it would be preferable in the future to formulate the tests in this executable format and generate documentation out of it. This ensures the perfect coherence between documentation and test execution (which can not be guaranteed up to now).

- On the GUI side an additional test results viewer was added, which displays the results of the assertions and the related signals.

One particular test was selected as a prototype for the new analysis concept, because in total it required the longest time to perform and had no automatic analysis module in LabView yet. Therefore the biggest margin of improvement could be gained by speeding up this particular test. In the longer term, the new mechanism shall replace the current LabView modules one after the other and will be capable of performing more sophisticated and general purpose analysis tasks.

MACHINE PROTECTION SYSTEMS COMMISSIONING

The biggest extension for the AccTesting framework which is currently ongoing, is to prepare it for use during the commissioning of the Machine Protection System (aka MPS) during the LHC startup in 2014 [6]. To meet the respective requirements, along with simple extensions, some fundamental changes and additions will have to be implemented.

Barriers

Currently the execution order of the tests within AccTesting is based only on dependencies between tests. Nevertheless a more general approach will be necessary during the MPS commissioning where the workflow will involve tests on different systems. For example, it is possible that a test on one system has to be done *some time before* another one on a different system. The concept of *barriers* will be introduced to solve this necessity: They will be put between test phases and they will ensure that all the tests affected by a certain barrier will take place before it and never beyond it. This will allow the execution and completion of the test plans to be performed in a more flexible way without losing any defined constraint. An example with two barriers is shown in Fig. 5.

Composite Tests

Currently one test in AccTesting is assigned exactly to one system but this approach might not always be that simple: One system might consist of several subsystems and a test might be formulated in a way, such that a certain set of tests on each subsystem has to be completed in order to contribute to the outcome of the test of the composite system. An example of this situation could be a test for a Beam Loss Monitor (BLM) crate which consists of one test for each BLM connected to that crate. To model this behavior an additional feature set will have to be implemented in AccTesting allowing the definition and the tracking of so-called composite tests, as illustrated in Fig. 6.

ISBN 978-3-95450-139-7

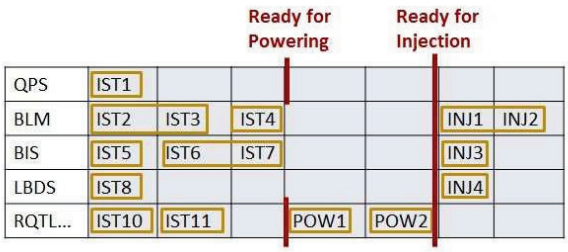

Figure 5: The Concept of test barriers: All the tests (brown rectangles) left of a barrier have to be successfully performed, before any of the tests right of the barrier are allowed to be started.

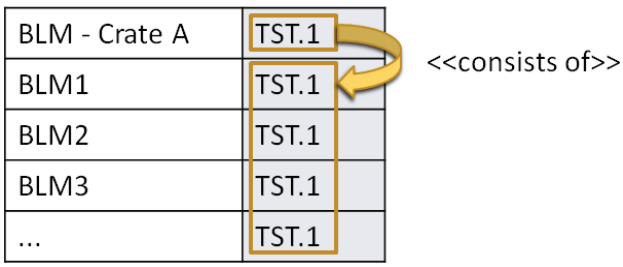

Figure 6: An example of a Composite Test: A test for a BLM crate could consist of a test for each BLM connected to the crate.

Systems and System Relations

While the initial version of AccTesting only supported electrical circuits, all the new features require the use of different kind of systems. Further, many of them require the knowledge of relations between systems. To manage this information in a consistent way, another sub-project was spawned [3]. This Systems Management project provides a slim framework, which collects the information about different systems (e.g. Circuits, Magnets) and relations between them and provides them, via a central Java API, to applications. While this framework is currently embedded in AccTesting, the final goal is to run it as a standalone server. Currently, it manages the information of about 17000 systems with 28000 relations between them.

Test Plan Editor

Up to now, it was only possible to edit the test plan by directly editing the database. Since this is problematic because of several reasons (Security, Consistency, Required Expert Knowledge), GUI support for performing this kind of task is in preparation. This will be especially needed as soon as AccTesting is used in a broader field, like MPS commissioning, where the test plan might have to be adapted more frequently (at least during the first campaign). The plan is to provide at least basic functionality in the beginning of 2014 and it is already possible to create and delete test plans. Extended functionality (Editing of Phases, Barriers and Composite Tests - see following sections) might have to be postponed until later in 2014.

CONCLUSIONS

Complex accelerator machines, such as the LHC at CERN, need robust and safe applications to lead their quality and safety tests. Working with thousands of systems, where each of them has tens of test to perform, was an additional challenge. The AccTesting framework has handled this task since 2011, providing a clear and complete overview of the situation helping the users during this very delicate process.

In this paper, we described the most recent extensions: The development of an analysis framework for test results and the AccTesting related preparations for MPS Commissioning. We outlined, how these extensions fit into the framework and summarized the changes in the core.

The framework served as a reliable tool for two hardware commissioning campaigns already and will, due to the ongoing work, be able to cover more and more needs of the overall commissioning process of CERN accelerator systems and beyond.

REFERENCES

[1] K. Fuchsberger, “Acctesting Framework - Motivation, Overview and First Experience”, CERN, Geneva, Switzerland 2012.

[2] M. Galetzka, M. Zerlauth, “Development and evaluation of a scheduling algorithm for parallel hardware tests at CERN”, Bachelor Thesis, University of Karlsruhe, 2012.

[3] M.Audrain et al., “System Dependency Management and Status Tracking for CERN Accelerator Systems”, ICALEPCS'13

[4] D.Anderson et al., “Using a Java Embedded DSL for LHC Test Analysis”, ICALEPCS'13.

[5] K.Fuchsberger et al., “Concept and Prototype for a Distributed Analysis Framework for the LHC Machine Data”, ICALEPCS'13.

[6] K. Fuchsberger, “Software tools for MPS”, MPP Workshop March 2013, Annecy, France.

ISBN 978-3-95450-139-7