Test Management Framework for the Data Acquisition of the ATLAS Experiment

IOP Conf. Series: Journal of Physics: Conf. Series **1085** (2018) 032054 doi:10.1088/1742-6596/1085/3/032054

Test Management Framework for the Data Acquisition of the ATLAS Experiment

A Kazarov[1], A Corso Radu[2], G Avolio[3], G Lehmann Miotto[3], I Soloviev[2] and G Unel[2]

[1]NRC "Kurchatov Institute" - PNPI, St. Petersburg, Russian Federation
[2]University of California, Irvine, USA
[3]CERN, Geneva, Switzerland

E-mail: Andrei.Kazarov@cern.ch

Abstract. Trigger and Data Acquisition (TDAQ) of the ATLAS experiment is a large distributed and heterogeneous system: it consists of thousands of interconnected computers and electronics devices that operate coherently to read out and select relevant physics data. Advanced testing and diagnostics capabilities of the TDAQ control system are a crucial feature which contributes significantly to smooth operation and fast recovery in case of problem and, finally, to the high efficiency of the whole experiment.

The base layer of the verification and diagnostic functionality is a test management framework. We have developed a flexible test management system that allows experts to define and configure tests for different components, indicate follow-up actions to test failures and describe inter-dependencies between TDAQ or detector elements. This development is based on the experience gained with the previous test system that was used during the first three years of data taking. We discovered that more emphasis needed to be put on the flexibility and configurability of the verification and diagnostics functionality by the many people that are, each, knowledgeable and expert on individual components of the experiment.

In this paper we describe the design and implementation of the test management system and also some aspects of its exploitation during the ATLAS data taking in the LHC Run 2.

1. Trigger and Data Acquisition of the ATLAS experiment at LHC

ATLAS Experiment

A Toroidal LHC ApparatuS (ATLAS) [1] is a particle physics experiment at the Large Hadron Collider (LHC) at CERN. The LHC is producing proton-proton head-on collisions with center-of-mass energy equal to 13 TeV at 40 MHz collision rate. The ATLAS detector comprises more than 140 million electronic channels which deliver raw event data at the rate of order of TB/s.

Trigger and Data Acquisition system

TDAQ is one of core ATLAS systems [2] which manages filtering and transfer of experiment data from the ATLAS detectors to large-scale mass-storage. The TDAQ system is composed of a large number of distributed hardware and software components (about 3000 machines and more than 40000 concurrent processes) which, in a coordinated manner, provide the data-taking functionality of the overall system. The system is required to handle data coming in parallel from the detector readout over some 1800 point-to-point readout links forming a flow of 1.5 MB

events at rate up to 100 kHz. During LHC runs, TDAQ is maintained in non-stop operation mode by a group of operators and experts on call.

Online Software
Given the complexity of the system and the probability of failures of hardware and software components, having advanced testing and diagnosing capabilities is an essential requirement for the TDAQ Online Software framework.

The Online Software [3] encompasses the software to configure, control and monitor the TDAQ system. It is based on a number of services which provide essentially the glue that holds the various sub-systems together. The Test Management is one of these services and is devoted to the verification of the functioning of the TDAQ system by executing tests on request.

2. Test Management Framework

Use Cases
The Test Management (TM) is a framework allowing from one side for experts: to develop individual tests for system components, to define dependencies between testable components, and to configure test dependencies and follow-up actions.

From another side, the testing expertise stored in TM is used (Figure 1): by the Run Control (RC) system that periodically verifies the functioning of the components it is in charge of; by the Central Hint and Information Processor (CHIP) [4] that analyses tests results to diagnose problems and applies follow-up actions; by the DAQ Operator or Expert who can have a view on testable hierarchy of TDAQ configuration components and perform testing of a subset of it via a dedicated graphical user interface. The Test Management utilizes other components of the Online Software framework. The most important ones are: Configuration Database (DB) [5] which provides the description of the TDAQ system (and tests) configuration, and Process Manager (PMG) [6] which offers a service to create, control and monitor the status of all the processes in the TDAQ system.

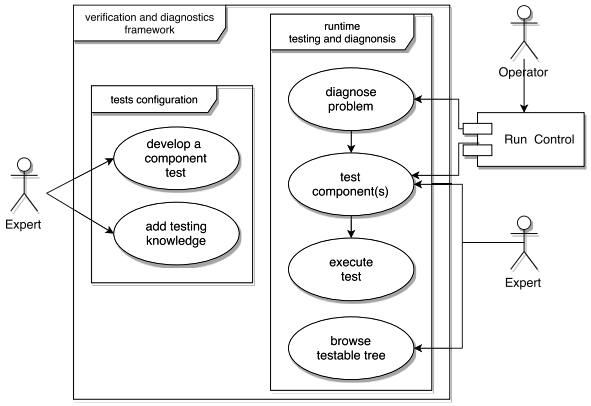

Figure 1: Test Management Framework use cases.

New requirements
The Test Management and Diagnostics service [7] was revised after the first data taking period of ATLAS. A set of new functional requirements were added:

- Experts shall be able to define the order in which tests should be executed for a component; the sequence may dynamically change based on the result of completed tests.
- Experts shall be able to define the order with which inter-related components shall be tested; the test sequence may change depending on the result obtained for the components.
- Experts shall be able to define what should be done upon failure of a test or a component to further diagnose the issue or recover.

All these requirements point towards an increased configurability of the system by ATLAS experts. The expert knowlege, like description of the testing behaviour listed above, must be

completely contained in the TDAQ configuration database, which can be populated and modified by the system experts using standard database editing tools.

An additional constraint was added, based on the evolution of other parts of the TDAQ system:

- The test management functionality should be provided in Java (in addition to C++), in order to be available for Java-based TDAQ applications like CHIP

Based on those extension of requirements a complete re-implementation was carried out.

Design and implementation
Typically tests are small programs or scripts which can be executed remotely on a host in the distributed TDAQ system, depending on configuration of a particular test. A single test can also be implemented as two or more programs started on different hosts, e.g. to verify the functionality of a link between source and destination elements in the system. A test returns a single result from a predefined enumeration `{Passed, Failed, Unresolved}`. Every component in the system may have a number of tests defined, where each individual test verifies a particular functionality of the component. In this case the test result for a component is a combination of individual tests resuls.

The test management framework is modular and is implemented as client libraries (Java, C++) and a Qt based application for the graphical user interface. In particular the package includes:

- a database schema to describe tests, components, dependencies and follow-up actions
- a client library (Java, C++) to perform tests on individual components
- a client library (C++) to handle the testing of components taking into account the other components they depend on (e.g. before testing an application, test that the computer it should run on works correctly)
- a GUI to allow the operator to browse a tree of testable TDAQ components and to request the execution of tests for a subset of it

The C++ and Java implementations address the same requirements and are based on the same configuration schema but they where developed independendly, as the need in Java implementation arised in course of the evolution of the TDAQ software. Both implementations achieve a good, comparable performance and are capable of launching many tests in parallel without imposing any significant overhead: for example a test for configuration of 1100 components (computer nodes) requires launching of 4500 individual tests on the tested nodes which is done in 32 parallel threads in total takes about 8 seconds. The Java implementation is, at the time being, capable of optimally parallelizing execution using less threads than the C++ implementation. Its threading model may thus be ported to the C++ implementation as well in the future.

The handling of dependencies between components is implemented at present only in the C++ library. A forward chaining inference engine is used to launch the testing of every component at the appropriate time, based on the tests policies and the test results obtained for other components.

Test Configuration schema
The Figure 2 represents an object schema for configuring all aspects of the test configuration. It can be divided into 3 main functional groups:
(1) Classes `Test`, `Test4Object`, `Test4Class` and `Test4CORBAServer` allow to describe a Test and to associate it to an object or class of objects in configuration. An `ExecutableTest` can be

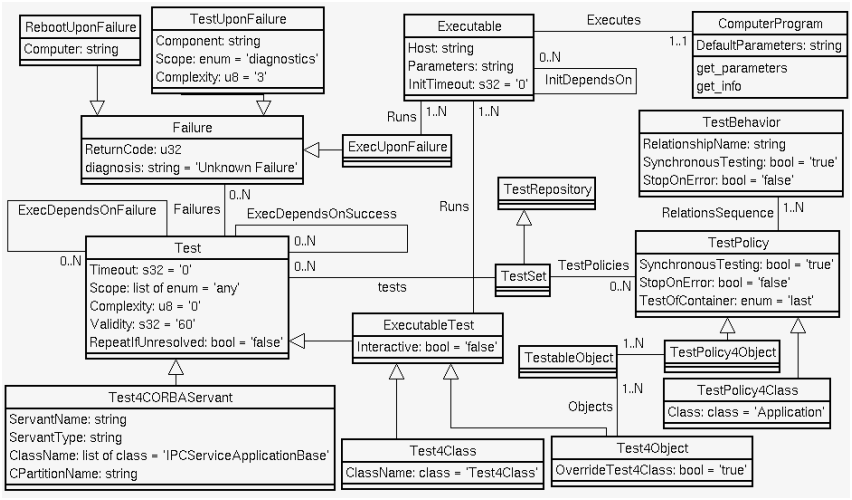

Figure 2: Test Configuration object schema.

launched on any host described in the system, whereas `Test4CORBAServer` is a special ping-like test which is executed locally as a library call in the caller thread, avoiding the overhead of launching of a process.
(2) Classes `TestPolicy`, `TestPolicy4Object`, `TestPolicy4Class` and `TestBehaviour` allows to build a hierarchical tree of testable components, following TDAQ configuration schema and to define policies (order, synchronicity) for testing the groups of components in the tree.
(3) Classes `Failure`, `RebootFailure`, `ExecUponFailure` and `TestUponFailure` define the possible actions which can be executed when test fails with a particular error code.

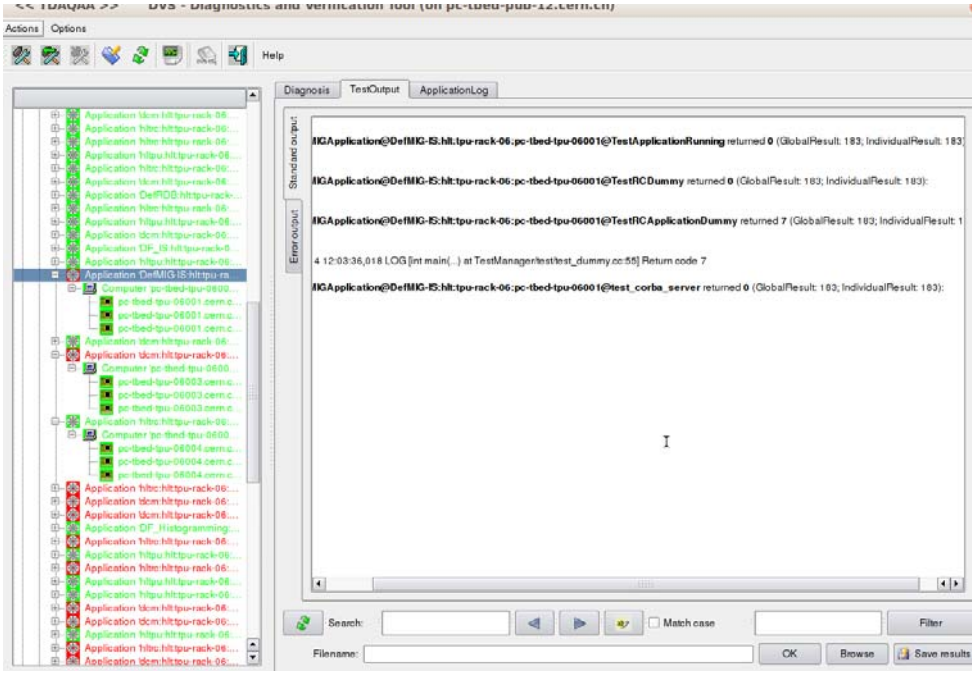

Figure 3: A schreenshot of the Testing GUI with a testable tree on the left and some tests results on the right panel.

GUI

The GUI (Figure 3) allows an operator to browse a testable TDAQ configuration, to select and to test one or more components, or to run individual tests for one component. The tree-like representation of a configuration is built according to the schema described before and indicates their interdependencies (e.g. an application relies on a working computer to run on). The operator can browse the detailed output of the tests, as well as the diagnosis that has been created based on the test results and the follow-up suggestions that have been configured by the experts in the configuration database.

3. Conclusions and outlook

A highly-configurable, modular and performant framework was developed for handling complex testing and diagnostics scenarios in Online Software for the ATLAS data acquisition system. It allows to store expert's knowledge about testing capabilities of the systems and to use it during TDAQ operations in order to maintain high data taking efficiency of the system in LHC Run 2. Some development is foreseen in C++ implementation in order to maximize utilization of available resources.

References

[1] ATLAS Collaboration 2008 *Journal of Instrumentation, vol.* **3** S08003
[2] M. Abolins et al. 2016 *JINST 11* **06** P06008
[3] Lehmann Miotto G. et al. 2010 *Nucl.Instrum.Meth.* **A623** 549-551
[4] Anders G. et al. 2015 *J.Phys.Conf.Ser.* **608** 1, 012007
[5] Lehmann Miotto G. et al. 2008 *J.Phys.Conf.Ser.* **119** 022004
[6] Avolio G. et al. 2008 *IEEE Trans.Nucl.Sci.* **55** 399-404
[7] Kazarov A. et al. 2007 *IEEE Trans.Nucl.Sci.* **54** 604-608