The Control and Configuration Software of the ATLAS Data Acquisition System: Upgrades for LHC Run 2

G LEHMANN MIOTTO[a], I ALEKSANDROV[b], G ANDERS[a], G AVOLIO[a], M CAPRINI[c], A CORSO RADU[d], M D'ASCANIO[a], J DE CeSTRO VARGAS FERNANDES[e], A KAZAROV[f], B KOLOBARA[a,g], A LANKFORD[d], F LAURENT[a,h], L MAGNONI[a], L PAPAEVGENIOU[a], Y RYABOV[f], A SANTOS[i], J SEIXAS[e], I SOLOVIEV[d], G UNEL[d], Y YASU[j]

a: CERN (CH), b: Joint Inst. for Nuclear Research – JINR (RU), c: IFIN-HH Bucharest (RO), d: University of California Irvine (US), e: Univ. Federal do Rio de Janeiro (BR), f: B.P. Konstantinov Petersburg Nuclear Physics Institute – PNPI (RU), g: Ministere des Affaires Etrangeres et Europeennes (FR), h: École Polytechnique Fédérale de Lausanne (CH), i: Universidad Nacional de La Plata (AR), j: High Energy Accelerator Research Organization (JP)

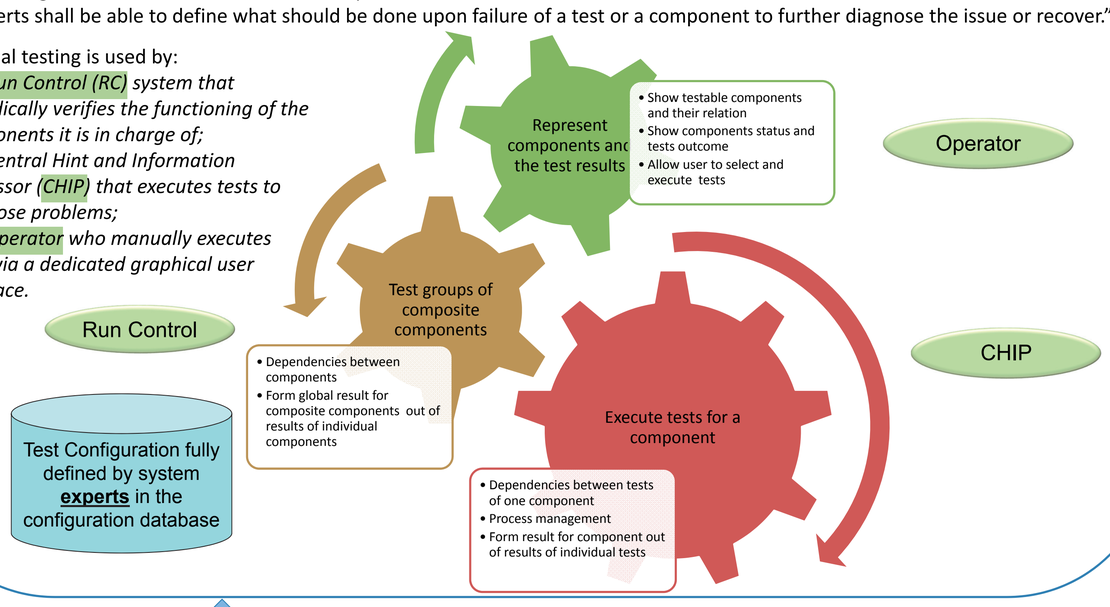
Requirements Extension

In a large, heterogeneous system such as the ATLAS TDAQ, it is essential to be able to verify the correct functioning of hardware and software components. Therefore, a **TDAQ functional testing framework** existed and was used already throughout Run 1. Additional requirements were identified with the experience gained during data taking:

- "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."

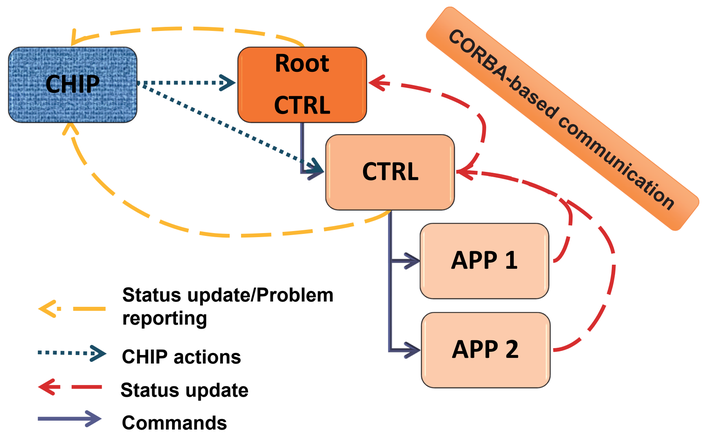
Functional testing is used by:

- *The Run Control (RC) system that periodically verifies the functioning of the components it is in charge of;*
- *The Central Hint and Information Processor (CHIP) that executes tests to diagnose problems;*
- *The Operator who manually executes tests via a dedicated graphical user interface.*



Software Refactoring

The **Run Control** (RC) system steers the data acquisition by starting and stopping processes and by carrying all data-taking elements through well-defined states in a coherent way. Given the size and complexity of the TDAQ system (2000+ PCs, 30000+ applications, 9000+ network ports,…), errors and failures are bound to happen and must be dealt with: in ATLAS this task is carried out by an **expert system.**
The RC and expert system components were tightly coupled in the first implementation of the software, with the consequence that their separation of duties became increasingly unclear with the additions of features during operations in Run 1.
Therefore these two components were completely redesigned, using a distributed run control tree and a separate central expert system application, the **Central Hint and Information Processor** (CHIP).



Applications in the ATLAS TDAQ system are organized in a tree-like hierarchical structure (the **run control tree**), where each application is managed by a parent **Controller**. The topmost node of the tree is the **Root Controller**. Controller applications are responsible to keep the system in a coherent state by starting and stopping their child applications and by sending them the proper commands needed to reach a state suitable for data-taking. Controller applications interact with **CHIP** by informing it about any changes (their own or their children). CHIP acts as an **intelligent system** having a global view on the TDAQ. It is capable of recovering from error conditions and guiding the TDAQ system through automated procedures in order to take data efficiently.
The RC has been re-implemented with state of the art C++ technologies such as boost and Intel Threading Building Blocks.
CHIP has been designed and implemented based on a third party open source java based **Complex Event Processing (CEP)** engine, ESPER.

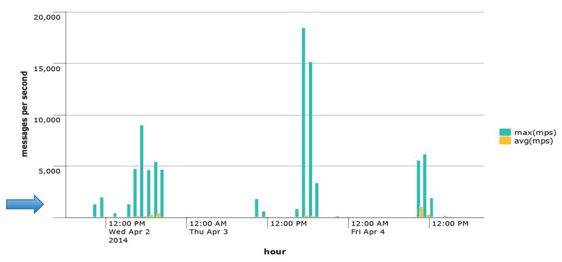
ATLAS in Run 1

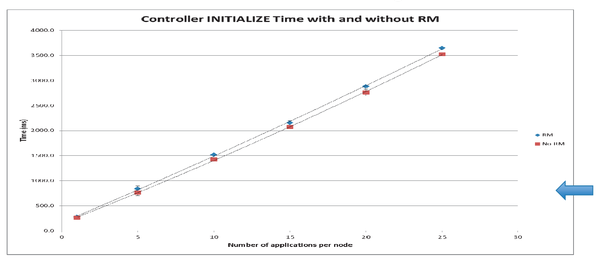
The ATLAS experiment at the Large Hadron Collider (LHC) operated very successfully in the years 2008 to 2013, identified as *Run 1*. It achieved an overall data taking efficiency of 94%, largely constrained by the irreducible dead-time introduced to accommodate the limitations of the detector read-out electronics. Out of the 6% dead-time only about 15% could be attributed to the central trigger and DAQ system, and out of these, a negligible fraction was due to the control and configuration sub-system. Despite these achievements, the first long LHC shutdown (2013-2014) was used to carry out a **complete revision of the control and configuration software**. The goals were three-fold: properly **accommodate additional requirements** that could not be seamlessly included during steady operation of the system; **re-factor software** that had been repeatedly modified to include new features, thus becoming less maintainable; seize the opportunity of **modernizing software** written at the beginning of the years 2000, thus profiting from the **rapid evolution in IT technologies**. This upgrade was carried out retaining the important constraint of minimally impacting the mode of operation of the system and public APIs, in order to maximize the acceptance of the changes by the large user community.
This poster illustrates, using a few selected examples, how the work was approached and which new technologies were introduced into the ATLAS DAQ system. Despite these being specific to this system, many solutions can be considered and adapted to different distributed DAQ systems.

Code Modernization

- **Message Transport System (MTS)**

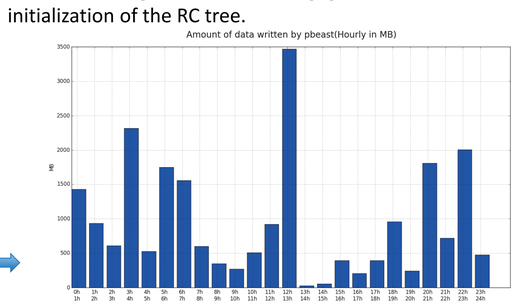
MTS underwent a review of the requirements that led to a complete redesign and new implementation to match its actual role (fast and reliable transport layer for TDAQ Error Reporting System messages). The redesigned system is **reliable, scalable and its performance has been improved**. The plot shows the rates of messages reported in MTS in technical runs in April 2014. In the condition of the plot (60000 applications running) MTS reached a maximum rate of 18kHz of delivered messages.





- **Resource Manager**

After an initial review and simplification of the requirements, the system underwent partial changes with the introduction of Boost multi-index containers. As a result **the code base has been reduced by 40%** against the previous implementation thus leading to a **more maintainable system**. The plot shows that the resource manager introduces a negligible overhead to the initialization of the RC tree.



- **Information System Archiver, P-BEAST (a Persistent Back-End for the ATLAS TDAQ)**

P-BEAST is a **new component** designed and implemented to archive operational monitoring information for analysis by experts. It provides CORBA and REST interfaces for data access. Its implementation is based on Google protobuf (data persistence), CORBA (internal protocol and user programming interface) and libmicrohttpd (Web server).

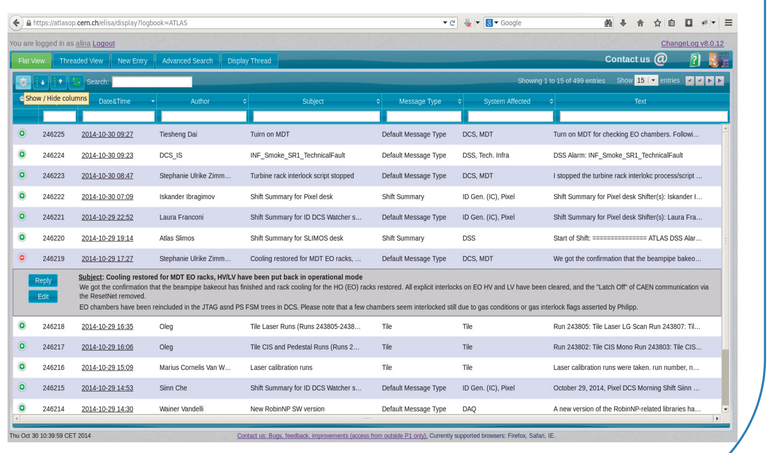
Web Based Tools

- **P-BEAST Dashboard**

This web application offers an interface to visualize any operational monitoring data published by the TDAQ system through configurable and customizable dashboards. The data is provided by P-BEAST and the application is based on the Grafana project, adapted to support a custom data source within the AngularJS framework.

- **ELisA**

The ATLAS electronic logbook (ELisA) is a web application used to record and share messages about ATLAS data taking activities by system operators, experts and automated services. The information is stored in an ORACLE database. The adoption of an MVC-driven architecture has allowed one to focus code development on specific features of the project, while profiting from the reliability of established third-party technologies such as the Spring framework. The tool also provides an HTTP-based REST API, such that other programs can access its features.



Conclusions & Outlook

The control and configuration software has contributed to the physics results obtained by the ATLAS experiment during Run 1 by ensuring smooth and efficient data taking. It was completely revised during 2013-2014 in order to accommodate additional requirements, improve maintainability and profit from advances in IT technologies: all this was done applying minimal changes to APIs, such that the large amount of client code would not need significant adaptations. The control and configuration software has already proved to be stable and well performing since the start of LHC Run 2 (2015 - 2018) and is prepared to face the new challenges that will arise during Run 2 operations. This experience has also demonstrated that **the overall architecture of the control and configuration system is flexible and supports partial upgrades, as well as step-wise modernization of its components**: this is fundamental for a system that is foreseen to run for the next 20 to 30 years and that will undergo several more upgrade iterations.