Introduction

The trigger and data acquisition (TDAQ) system of the ATLAS\(^2\) experiment at the LHC is controlled and configured via a software infrastructure that takes care of coherently orchestrating the data-taking, the Control, and Configuration software. Its overall architecture, established at the end of the 2001, has proven to be so flexible and has contributed to the experiment’s excellent data-taking efficiency and, therefore, its Physics research achievements. Nevertheless, many additional modifications were introduced in the software in the course of Run 1 (2008-2012) in order to fix bugs, enhance error detection and handling capabilities, automate procedures and accommodate other needs and user requests. In the course of Run 1, software changes driven by operations left some parts of the code unclear and hard to maintain, such that a complete re-implementation of individual software components was deemed necessary by the start of the first long shutdown (LS1) of the accelerator.

In addition, in the course of Run 1 new requirements were defined. Some of those could not be implemented in a running system and/or with the available manpower. Those requirements were therefore addressed in LS1.

Last but not least, several IT technologies matured in the past few years: after careful evaluation we chose to introduce some new technologies in the Control and Configuration software. The choice of this new technology led to advantages in terms of user experience, development and maintenance costs.

This poster gives an overview of the main changes carried out on the Control and Configuration software during LS1. We will describe the new technologies that were adopted and the way in which they are used, present some of the main results obtained, and indicate the areas in which other upgrades may be envisaged in the future.

Core Modifications

- **Configuration Database\(^{3}\)**
  Databases schemes and algorithms have been modified to describe similar components by using few template objects. These changes have increased the maintainability of the system by adding one object instead of tens and reduced the overall configuration size (a reduction factor of 10 can be noticed).

- **Test Manager\(^{4}\)**
  The Test Manager is a service devoted to the verification of the TDAQ system functioning by executing tests on request. The environment has been re-organized and re-configured from the configuration database in order to obtain that, new features have been introduced: test policies, new types of tests (CORBA\(^3\) interfaces), handling possible failures (extended diagnostic and recovery scenarios related to them).

- **Message Transport System\(^{5}\)**
  MTS undertakes 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 Recordings System\(^3\) messages). The redesigned system is reliable, scalable and its performance has been improved.

- **Information System Archiver\(^{6}\)**
  The Information System Archiver (ISA) is a tool for archiving operational monitoring information for analysis by experts. It provides CORBA and REST interfaces for data access. It has been developed from scratch and using protocols supported by Google protocol (data persistence), CORBA (internal protocol and user programming interface) and libwebsockets (Web server).

- **Resource Manager\(^{7}\)**
  After an initial review and simplifications of the requirements, the system underwent partial changes with the introduction of Boost multi-index container. As a result the code base has been reduced by 40% against the previous implementation thus leading to more maintainable system. The plot shows that there is no need by the inclusion of the resource manager.

Web application deployment improvements

The main idea behind the new design is to isolate web applications inside a DMZ, limiting the access to resources located inside the ATLAS Technical Control Network. This arrangement is considered as a separation between Web and core applications.

To increase security applications that are running on the same host should be isolated from each other. Moreover, by not sharing the environment, different applications are allowed to use conflicting technologies. This can be achieved using virtual machines or more lightweight solution like Linux containers.

Docker\(^{13}\) can be used to specify all requirements and build instructions for a Linux container. A tool like Puppet\(^{21}\) can be used to set up an environment inside a VSA.

Run Control & CHIP

The Run Control (RC) and the Central Hint and Information Processing (CHIP) are key components of the Control and Configuration Software. The RC system skews the data acquisition by starting and stopping processes by carrying all data-taking elements through well-defined states in a coherent way. During the Run 1 the RC has been completely re-designed with state of the art Complex Event Processing such as Esper\(^{16}\) and Threading Building Blocks\(^{17}\) (TBB).

Given the size and complexity of the TDAQ system (2000+ PCs, 3000+ applications, 9000+ network ports...), errors and failures are bound to happen and must be dealt with. The data acquisition system has to recover from these errors promptly and effectively, possibly without the need to stop data-taking operations. To achieve this, CHIP has been introduced.

CHIP is an intelligent system having a global view on the TDAQ. It is based on a third-party open source Java based Complex Event Processing (CEP) engine, ESPER.\(^{18}\) It aims to supervise ATLAS data-acquisition by taking operational decisions and handling abnormal conditions.

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 are also the ones that interact with CHIP by informing it about any changes (their own or their children), allowing CHIP to take the needed actions.

User Tools

- **P-BEAST Dashboard**
  This web application provides an interface to visualize the amount of resources in use by TDAQ system through dashboards and plots, configurable by the user according to own needs. The data are provided by P-BEAST and the application is based on the Grafana\(^{19}\) framework. The tool has been adapted to support a custom data source by AngularJS\(^{20}\).

- **Elia\(^{2}\)**
  The ATLAS electronic logging (Elia)\(^{2}\) 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 a MVG-driven architecture has allowed to focus code development on specific features of the project, while profiting from the reliability of established third-party technologies such as the Gring\(^{2}\) framework. The tool provides as well HTTP-based REST API for programmatic access to its features.

- **DBE**
  This database editor application allows the user to create, modify and delete Configuration Databases. The service provides to the user the possibility to have multiple views on the data stored in the configuration database and configure its views to one’s needs. It replaces the current MOTIF based implementation with a more modern alternative written in C++ with the Qt\(^{2}\) library.

- **SRapport** (Parallel session Track4, 4/14 17:00)
  It is a service to test new or modified Shifter Assistant\(^{2}\) (an open source system which simulates the ATLAS subsystems) directives by executing the system inside a sandbox, and using a configurable replace of past and archived data, as needed for the test. The variation of the run numbers is controlled and independent on the environment, keeping a history of past tests for further iterations and collaboration between developers. It is based on a software stack using a Oracle database, P-BEAST, and Java at the back-end, and Pythran and Django\(^{2}\) at the Web user interface front-end.

References

5. Configuration and Control of the ATLAS Trigger and Data Acquisition, 2008. Controls and Configuration/CC working group
9. 9 New Persistent Back-end for the ATLAS Online Information Service, 2014. I. Soloviev, A. Sindre
12. 12 http://oracle.org
13. 13 http://docker.org
14. 14 http://puppetlabs.com
15. 15 http://blueprints.openstack.org
16. 16 http://threadingbuildingblocks.org
17. 17 https://docker.org
18. 18 https://puppetlabs.com
19. 19 https://esper.codehaus.org/ (May 2014)
20. 20 https://nodejstoolbox.com/index.html
21. 21 https://nodejstoolbox.com
22. 22 https://www.gnu.org/software/libwebsockets/