Applications of advanced data analysis and expert system technologies in the ATLAS Trigger-DAQ Controls framework

G Avolio¹, A Corso Radu¹, A Kazarov², G Lehmann Miotto³, L Magnoni³

¹ Department of Physics and Astronomy, University of California, Irvine, USA
² Petersburg Nuclear Physics Institute, NRC Kurchatov Institute, Gatchina, Russia
³ CERN, Geneva, Switzerland

E-mail: Giuseppe.Avolio@cern.ch

Abstract. The Trigger and Data Acquisition (TDAQ) system of the ATLAS experiment is a very complex distributed computing system, composed of more than 20000 applications running on more than 2000 computers. The TDAQ Controls system has to guarantee the smooth and synchronous operations of all the TDAQ components and has to provide the means to minimize the downtime of the system caused by runtime failures. During data taking runs, streams of information messages sent or published by running applications are the main sources of knowledge about correctness of running operations. The huge flow of operational monitoring data produced is constantly monitored by experts in order to detect problems or misbehaviours. Given the scale of the system and the rates of data to be analyzed, the automation of the system functionality in the areas of operational monitoring, system verification, error detection and recovery is a strong requirement. To accomplish its objective, the Controls system includes some high-level components which are based on advanced software technologies, namely the rule-based Expert System and the Complex Event Processing engines. The chosen techniques allow to formalize, store and reuse the knowledge of experts and thus to assist the shifters in the ATLAS control room during the data-taking activities.

1. Introduction
The ATLAS [1] Trigger and Data Acquisition (TDAQ) [2] system is operated by a non-expert shift crew, assisted by a set of experts providing knowledge for specific components. The daily work of operators is made of procedures to run the system, periodic checks and controls on system status, well defined reactions in case of known problems and interaction with experts in case of non-standard issues. The evaluation of the status of the TDAQ system requires strong competencies and experience in understanding log messages and monitoring information, and often the meaningful information is not in the single event but in the correlation of several events in a certain time-line. Due to the very critical operational task, both economically (i.e., beam time is rather expensive) and in terms of manpower, dealing fast and effectively with problems and failures is fundamental in order to minimize operational inefficiency.

This paper describes the verification, diagnostics, error-detection and error-recovery components used in the frame of the TDAQ Controls framework [3].
1.1. The monitoring infrastructure

The evaluation of correctness of running operations requires shifters and experts to gather and correlate information from multiple data sources, often to be aggregated in a certain time-line. Information sources are spread among the different levels and provide views on multiple aspects of the data acquisition system. The monitoring infrastructure is composed of data providers at different levels that can be grouped in three main categories:

- **TDAQ core-services**: which provide access to low-level, unfiltered data about the basic activities in the system (e.g., application messages, process communication and system configuration);
- **Data Monitoring tools**: a set of high-level monitoring tools that provide views at different levels of the data-flow chain. They may collect and aggregate information from other providers to compute new information, such as the overall data filtering and archiving rates during runs, or the occupancy of different network segments. Monitoring tools provide also operational information about the status of the Large Hadron Collider (LHC) and of the various ATLAS sub-detectors;
- **Farm tools**: a set of specific tools managed by system and network administrators to get information about the status of the farm and of the networking infrastructure.

1.2. Error management in the TDAQ system

Given the size and complexity of the TDAQ system, 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. In the TDAQ an Error Management System (EMS) has been implemented and is currently in use. This EMS proved to be very effective for automated recovery for well-defined situations.

Nevertheless, only a fraction of the overall operational procedures can be automated, and a non-negligible fraction of the TDAQ operational inefficiency is coming from situations where the human intervention is still required. In this respect, a high level tool helping operators with diagnosis of problems and suggesting appropriate reaction has been developed: the DAQ Assistant [4].

2. The Error Management System

The Error Management System (EMS) aims at detecting failures and performing recovery procedures during data taking operations without the need for human intervention. Its main functionalities are:

- Gather the knowledge on system condition and errors, by connecting to the core services;
- Detect problems and react appropriately.

A rule based expert system is used at the core of the EMS and is described in the next section.

2.1. The CLIPS framework

The EMS is implemented on top of CLIPS\(^4\) (C Language Integrated Production System), an open-source expert system framework. Some of the main features of the CLIPS framework are:

- An inference engine supporting forward chaining;
- Support for both procedural and object-oriented programming (“COOL” language) in addition to the declarative rule programming;
- Represents expert knowledge in an IF-THEN form which is human readable;
- It is easy to extend using the C++ programming language.

\(^4\) [http://clipsrules.sourceforge.net/](http://clipsrules.sourceforge.net/)
CLIPS uses the Rete [5] algorithm for driving the inference engine. The Rete algorithm is best used in situations with many rules/many objects and is therefore well suited for representing the complexity of the TDAQ system.

2.2. The Knowledge Base
At the core of the expert system is naturally the Knowledge Base (KB) containing the necessary rules to affect the EMS. Information about the different applications, computers and other hardware is represented in the expert system using proxy objects. Whenever the ES is started it is populated with relevant information such as class instances representing applications running in the system. This information can then trigger rules in the expert system.

2.3. Architecture
The EMS system includes two high-level components which are responsible for the automation of system verification, diagnostics of failures and recovery procedures (figure 1): DVS (Diagnostics and Verification System) [6] and Online Recovery [7].

The Online Recovery is composed of a global server that handles errors having a system-wide impact, and by local units to handle errors that can be dealt with at a subsystem level, that is, errors that do not have an immediate effect on the rest of the system. For execution of atomic tests from a Test Repository, the Test Manager service is used which, in turn, utilizes other Controls services that are not covered in this paper. In figure 2 the Online Recovery working model is shown: the system is described and configured via a Configuration Service; the sets of objects for the actual configuration is loaded by the ES (using the COOL object-oriented language provided by CLIPS); the ES engine uses information coming from errors, messages and tests to match the loaded rules.

The DVS is a framework used to assess the correct functionality of the system, to detect and diagnose eventual problems. DVS allows the configuration of one or several tests for any component in the system by means of a configuration database. The system and the testing results can be viewed in a tree like structure using a user friendly graphical user interface. The DVS is able to presents the
TDAQ system in a form of testable tree-like structure, allowing automating testing and diagnostics of problems at different levels of the configuration.

2.4. Limitations
Although the rule-based expert system approach suits well for the error recovery functionalities provided by the EMS, there are limitations preventing its adoption as a more general intelligent engine:

- Forward chaining is not appropriate for root cause analysis of problem. It adopts a data-driven approach, i.e. the engine starts rules evaluation with the available data and uses inference rules to extract more data until a goal is reached. It can be used to detect error conditions but it is not meant to deduce how a particular goal was achieved;
- The ability of the system to perform reasoning about time is very limited. While it can react to and deal with a large number of facts, the IF-THEN approach is not meant to detect complex patterns over time;
- System complexity: an expert system approach with broader requirements set was firstly implemented more than 8 years ago for controlling the TDAQ system. The size and the complexity of the knowledge base became very hard to maintain and it eventually required a redesign to deal only with error-management functionalities.

3. The DAQ Assistant
As previously mentioned, a non-negligible fraction of the TDAQ data taking inefficiency is coming from situations where human intervention is involved. A high-level monitoring tool helping operators with automated diagnosis of problems and suggesting the appropriate reaction could reduce the time for error management and minimize the loss of experimental data. This is the objective of the DAQ Assistant: to be an automated and intelligent assistant for the TDAQ operators.

3.1. Aims
Assisting the TDAQ operators means increasing the situational awareness they have on the data taking operations. The targets for the assistant are both shifters and experts. It aims at providing a clear and effective support for shifters as well at presenting detailed and complex system analysis for experts in case of problem troubleshooting.

3.2. Requirements
The assistant aims to be intelligent in the way it processes the TDAQ working conditions and automated in how it detects problems and notifies operators. Its main requirements are:

- To automate checks and controls in real-time;
- To detect complex error situations, performing time-based analysis on multiple system conditions;
- To receive instructions from the TDAQ experts on what to detect and how to react, building a knowledge-base of instructions;
- To effectively notify the TDAQ operators with the problem diagnosis and appropriate reaction.

3.2.1. Complex Event Processing
The need to process streams of information from distributed sources at high rate with low latency is of interest from the most disparate fields: from wireless sensor networks to financial analysis, from business process management to fault diagnosis. All these applications rely on an information processing engine capable of timely processing and digesting the flow of data, to extract new knowledge to answer complex queries and to promptly present results. In recent years CEP technologies have emerged as effective solutions for information processing and
event stream analysis. CEP technologies provide the means to reason upon events and relationships among them. Esper\textsuperscript{5} is the leading open source engine for complex event processing and it has been adopted as CEP engine in the DAQ Assistant. It is designed for high volume event correlation over millions of events with low latency. Esper focuses on providing powerful processing capabilities via a high-performance engine with a rich and flexible API. Event patterns are expressed via the rich Event Processing Language (EPL) that supports filtering, aggregation, and joins, possibly over sliding windows of multiple event streams.

3.3. Architecture

The DAQ Assistant performs a real-time\textsuperscript{6} analysis of the whole TDAQ system, detecting problematic situations and misbehaviors and producing notifications to operators. It is able to react on single problems (e.g., a log message reporting a network connectivity issue from a data acquisition application), but it offers more advanced correlation and analysis capabilities (e.g., if a burst of similar log message is received in a short time period from multiple applications belonging to the same farm rack, then the problem should be recognized as a network switch failure). Three main operational stages are identified: information gathering, information processing and result distribution.

The Assistant combines technologies coming from different disciplines. In particular it leverages on an event driven architecture to unify the flow of data to be monitored, on a Complex Event Processing (CEP) engine for real time correlation of events and pattern recognition and on a Message Queuing system for components integration and communication. The picture in figure 3 presents an overview of the architecture together with the three operational stages.

![Figure 3. High-level view of the DAQ Assistant architecture and operational stages.](image)

\textsuperscript{5} [http://esper.codehaus.org/](http://esper.codehaus.org/)

\textsuperscript{6} In this context the adjective real-time refers to a level of responsiveness that a user senses as immediate or nearly immediate, as the delay from problem detection to notification is expected to be in the order of few seconds.
3.3.1. **Information gathering**  The information about correctness of data acquisition operations in the TDAQ infrastructure is spread among several data sources, different for data formats, technologies and publication mechanisms. The DAQ Assistant is able to gather and process all log messages from data acquisition applications, the operational data published in the information system, the network and farm metrics, as well as data retrieved from the configuration databases. The high-rate of information events, that can reach spikes in the order of hundreds of kHz, together with the diversity of technologies and of data formats are the main challenges concerning information gathering.

3.3.2. **Information processing**  The continuous processing of monitoring data in order to detect problems and failures is the key objective of the DAQ Assistant. The Assistant is fed with instructions about what situations to detect by the TDAQ experts, leveraging their know-how and expertise on the TDAQ system and operational procedures. The main aspects of information processing are real-time complex data processing (continuous evaluation of monitoring data streams to detect complex pattern) and knowledge engineering (formalize expert knowledge in patterns of monitoring events, together with instructions on what type of result the pattern detection should produce).

The DAQ Assistant relies on a CEP engine to provide the real-time processing functionalities and, for what concerns knowledge engineering, it implements a flexible approach based on generic directives structured as XML documents. A directive is composed by two main elements: the *pattern*, which defines the sequence of events to react on, and by one or more *listeners*, that define the actions to be performed when the pattern is matched. A directive is derived from the more generic concept of CEP rule but it introduces a precise structure for the listener part, specific to the DAQ Assistant project.

3.3.3. **Result distribution**  The DAQ Assistant has been designed to support different types of reactions in case of pattern detection; the generation of alerts is the most common reaction to pattern detection. Alerts are generated to notify the TDAQ operators of problems and failures in the system. Alerts can be customized per TDAQ sub-system (e.g., an alert can be addressed to specific TDAQ shifters), offering customized views on the system conditions. Alerts provide operators with the complete set of information they need to react promptly and effectively to the problem. An alert is composed by several fields: *problem description* (brief description of the problem detected), *reaction* (expected reaction to be taken by the operator), *severity* (the severity of the issue), *domain* (the domain of the notification), and *pattern details* (all the information, as collected by the patterns, about the events that triggered the alert).

3.4. **Integration**

The DAQ Assistant project has a loosely-coupled architecture where two main modules interact via a message broker, also known as event/message bus. As presented in figure 4 the DAQ Assistant components are:

- *The engine*: responsible for the collection and correlation of monitoring data as specified in directives. It is a Java-coded service that manages data gathering, events processing and results generation;
- *The web application*: responsible for providing a dynamic and interactive visualization of alerts for operators;
- *A message broker* (Apache ActiveMQ\(^7\)) that centralizes the communication between modules.

A message queuing system, or message broker, provides a generic communication facility for heterogeneous components via a publish/subscribe interface for sending and receiving messages. In this case the engine acts as a message producer while the web application acts as receiver.

\(^7\) [http://activemq.apache.org/](http://activemq.apache.org/)
The web application allows both shifters and experts to monitor the TDAQ system conditions independently from the platforms and the device used, improving the overall effectiveness of the whole project. It collects and archives alerts produced by the engine and builds rich web pages for alert visualization. The rich functionalities are a set of actions operators can perform:

- Mark alerts as read (e.g., when a problem has been handled);
- Mask read alerts (e.g., to visualize only new alerts as they arrive);
- Filter on alerts parameters;
- Browse alert history;
- Customize pages layout (e.g., create new pages containing a desired category of alerts, identified by a domain expression).

![Diagram](image)

**Figure 4.** The DAQ Assistant components: the engine, the web application and the message broker.

The web pages are automatically updated when new alerts arrive via an asynchronous communication over HTTP (AJAX\(^8\)) between the user's browser and the ActiveMQ message broker. This guarantees a prompt notification of new problems as they happen.

### 4. Conclusions

In this paper two services provided by the ATLAS TDAQ Controls framework have been presented: the Error Management System and the DAQ Assistant. The former is built on the top of a rule-based expert system, while the latter unleashes the power of a Complex Event Processing Engine. Both services are currently widely used during the ATLAS data taking operations and successfully

\(^8\) [http://en.wikipedia.org/wiki/Ajax_%28programming%29](http://en.wikipedia.org/wiki/Ajax_%28programming%29)
accomplish their main tasks: effective monitoring and prompt error handling, fundamental in order to maximize the data taking efficiency of the whole experiment.

References
[1] The ATLAS Collaboration 2008 The ATLAS experiment at the CERN Large Hadron Collider *J.Instrum.* 3 S08003
[2] Jenni P, Nessi M, Nordberg M and Smith K 2003 *ATLAS high-level trigger, data-acquisition and controls : Technical Design Report* (Geneva : CERN)
[3] Lehmann Miotto G et al. 2010 Configuration and control of the ATLAS trigger and data acquisition *NIMA* 623 549-551
[4] Magnoni G, Kazarov A, Lehmann Miotto G 2011 *Proc. 14th Int. Workshop On Advanced Computing And Analysis Techniques In Physics Research*
[5] Forgy C 1982 Rete: A fast algorithm for the many pattern/many object pattern match problem *Artificial Intelligence* 19 17–37
[6] Kazarov A et al. 2007 A rule based verification and control framework in ATLAS Trigger-DAQ *IEEE Transaction on Nuclear Science* 54 604-608
[7] Sloper JE, Lehmann Miotto G, Hines E 2008 Dynamic Error Recovery in the ATLAS TDAQ System *IEEE Transaction on Nuclear Science* 55 405-410.