Use of expert system and data analysis technologies in automation of error detection, diagnosis and recovery for ATLAS Trigger-DAQ Control framework

Andrei Kazarov, Alina Corso Radu, Luca Magnoni and Giovanna Lehmann Miotto

Abstract—**Trigger and Data Acquisition (TDAQ) System of the ATLAS experiment on LHC at CERN is a very complex distributed computing system, composed of O(10000) applications running on a farm of commodity CPUs. The system is being designed and developed by dozens of software engineers and physicists since end of 1990's and it will be maintained in operational mode during the lifetime of the experiment. The TDAQ system is controlled by the Control framework, which includes a set of software components and tools used for system configuration, distributed processes handling, synchronization of Run Control state transitions etc. The huge flow of operational monitoring data produced is constantly monitored by operators and experts in order to detect problems or misbehavior. Given the scale of the system and the rates of data to be analyzed, the automation of the Control framework functionality in the areas of operational monitoring, system verification, error detection and recovery is a strong requirement.**
The paper describes requirements, technologies choice, high-level design and some implementation aspects of advanced Control tools based on knowledge-base technologies. The main aim of these tools is to store and to reuse developers expertise and operational knowledge in order to help TDAQ operators to control the system with maximum efficiency during life time of the experiment.

I. Introduction

THE paper describes advanced Control tools based on knowledge-base technologies. First, we introduce the Control framework for the TDAQ system [1] of the ATLAS experiment [2] on LHC at CERN, then we describe motivations and requirements for the use of knowledge-based intelligent monitoring and diagnostics tools in the framework. In Section III we present the high-level design and some implementation aspects of the Control components carrying out the functionality of error detection and analysis (AAL), error diagnosis and verification (DVS) and error recovery (OR). More attention is payed to AAL and OR components, which were actively developed in last years. The section includes discussion on the choice of knowledge-based technologies used for implementation of the described components.

Manuscript received May 31, 2012.

A. Kazarov is with CERN, on leave from Petersburg Nuclear Physics Institute, Kurchatov NPI, Gatchina, Russian Federation (e-mail: Andrei.Kazarov@cern.ch).
A. Corso Radu is with UCI California Irvine(e-mail: Alina.Radu@cern.ch).
L. Magnoni is with CERN, Geneva, Switzerland (e-mail: luca.magnoni@cern.ch).
G. Lehmann Miotto is with CERN, Geneva, Switzerland (e-mail: giovanna.lehmann@cern.ch).

II. Motivations and requirements

Control framework in TDAQ System

The TDAQ Control framework [3] was developed in the scope of the TDAQ system for the ATLAS experiment at LHC, CERN. Its main objectives are to process control commands of TDAQ operators and to steer the whole TDAQ system through run control transitions, guaranteeing synchronous execution of commands and smooth error recovery. The TDAQ Control framework is composed of a number of services, starting from the basic ones as resource, access and process management and ending with integrated GUI tools for TDAQ operators. Control components widely use other core TDAQ services, namely the Configuration Service and the Monitoring Infrastructure.
The following factors had an important impact on the design of the TDAQ Control system:

- system complexity and heterogeneity
- use of commodity hardware which fails quite often
- long development and maintenance time-line
- involvement of a big number of developers, high rate of staff rotation

Operational aspects

The TDAQ system is operated by a non-expert shift crew of operators assisted by a set of experts providing knowledge for specific subsystems and components. The daily work of operators is made of procedures to run the system, periodic checks and controls on system status, well defined reaction in case of known problems and interaction with experts in case of non standard issues.
During a typical physics run, tens of thousands of messages can be produced by different applications (especially in case of some operational problems), and the rate of information objects updates can be as much as tens of kHz. In this conditions, prompt reaction to the errors can hardly be done by a human operator, and Control tools must include tools for automation of most critical operational tasks.

Requirements

In the described conditions, in order to guarantee the high run efficiency of the experiment and to minimize downtime resulting from failures, the TDAQ Control systems must include advanced intelligent tools to help TDAQ Operators and to automatize the important and complex control operations, typically manned by humans, in particular in the following important areas: system verification, error detection and analysis, error diagnosis and error recovery. From this perspective, the principal requirements to the TDAQ Control system are:

- to analyze information flow and error messages in the system and to perform diagnostics of the operational errors
- to recover from possible system failures with minimal disturbance to the running system
- to perform verification of the state of system components
- to keep and reuse the expertise of system developers experts and to allow storing of the operational experience acquired during the years of system maintenance
- to allow flexibility in defining Control system behavior in order to adapt promptly to changing experiment conditions and requirements

III. Design and Implementation aspects

A. Choice of technologies

Given the requirements it was decided that Control components responsible for error detection, analysis, diagnosis and recovery shall be based on knowledge-base software technologies, namely on a rule-based expert system (ES) and a complex event processing (CEP) engines. The use of such techniques allows to formalize, to acquire, to store and to reuse the system developer's knowledge and expertise in automation of complex controls tasks. Also, the non-procedural programming paradigm allows easy modification and extension of the knowledge base (KB) (which is maintained as a set of human-readable text files) without need to reprogram system components.

Expert system: CLIPS: CLIPS [4] is a forward-chaining expert system framework originally developed by NASA. It is open-source and the ES engine can be embedded into other applications that allows to use it as part of TDAQ software. The CLIPS engine implements the Rete algorithm to match facts (e.g. the run-time messages and information updates coming from the Monitoring system) to the set of rules loaded into the engine. The rules are kept in readable text files, so the KB can be easily maintained and extended independently by the experts. CLIPS features object-oriented (OO) language called COOL allowing you to define classes and instantiate objects. This maps very well to the OO configuration approach used in TDAQ software and thus allows experts to speak in ”TDAQ language” when developing the rules.
CLIPS is used as an expert system engine for the OR and DVS components.

Complex Event Processing: ESPER: Complex Event Processing [5] is one of the fundamental techniques in the so called Operational Intelligent procedures, where the goal is to provide insight into dynamic operations by running query analysis against live feeds and event data. In the last decade it has been adopted in very different context, from financial analysis to intelligent system monitoring. CEP systems are meant to analyze and correlate streams of events, but the great advantages they offer is to abstract events and to provide a standard language to express pattern and correlation as queries. ESPER [6] is a CEP engine written in Java that offers a very powerful Event Processing Language with built-in capabilities for analysis over time and event correlation. ESPER is able to handle very high flow of data, thanks to a non-persistent approach where events pass through memory and the engine continuously check for the one matching criteria expressed by queries. This characteristic perfectly fit with Control software needs, so we adopted it for the event analysis.
CEP event processing model, where streams of events are queried by the rules, can be seen as extension of traditional expert system ”rules-to-facts” matching model: time dimension is added to ”flat” field of facts, thus adding the power to make more complex queries over the time dimension, retaining the possibility to correlate and to join different facts in the forward-chaining manner of traditional expert system. Given the fact that the TDAQ system is trigger-driven where events occur at predictable rates, the possibility to analyze events over time axis is an important addition to the analysis and diagnostics capabilities of the Control tools.

Knowledge gathering and maintaining: Knowledge is gathered from experts, formalized in CLIPS or ESPER rules and stored in the KB which is a set of text files. This task is carried out by a small group of experts (knowledge-engineers) which are communicating closely with the TDAQ and detectors experts providing their expertise. After a rule is implemented, it can be adjusted directly by an expert, willing to tune some parameters like constants, thresholds or message text. The development foresees storing the rules in a database and providing more advanced expert-oriented editing facilities, which may include possibility to create new rules, based on common templates.

B. High-level design

The TDAQ Control system includes the following intelligent components designed on these principals: Automated AnaLysis (AAL) project, Diagnostics and Verification System (DVS), Online Recovery System (OR). These components cover all aspects of the error handling in the Control system, as shown on the Figure 1. A typical cycle of error handling is started with error detection and analysis of the operational information from monitoring systems. The filtered and analyzed information is used at the next step of the error diagnosis which usually involve deep testing of a faulty subset of the system, in order to complement monitoring information with more precise knowledge about state of the system components. Finally the error handling is completed by recovery actions that shall bring the system back to its normal operational state. At every stage, passive monitoring information may be not enough in order to make the correct evaluation of the error conditions. A more complete picture about the status of the system is acquired by means of executing the tests which are developed by experts and made available to the Control components via the Test Repository and DVS framework, as described in Section III-D. Experts contribute the knowledge which is stored in the KB. The tools are making evaluations and decisions by applying

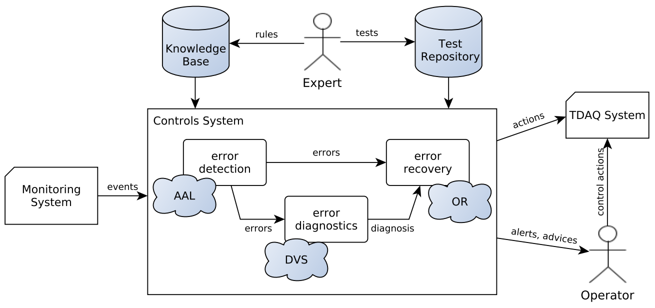


Fig. 1. Use of expert knowledge in automation of error handling in TDAQ Control system

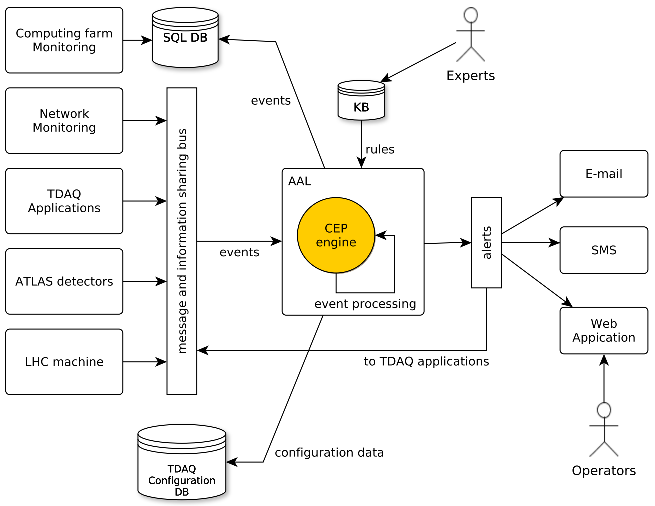


Fig. 2. High-level architecture of AAL component

the knowledge to the run-time operational monitoring data and to the results of tests execution.

C. Error detection and analysis: AAL component and Shifter Assistant

During data taking runs, streams of messages sent by applications together with data published via information services are the main sources of knowledge about the correctness of running operations. The huge flow of data produced (with an average rate of O(1-10KHz)) is constantly monitored by experts using variety of monitoring tools to detect problems or misbehavior. This requires strong competence and experience in understanding and discovering problems and root causes, and often the meaningful information is not in the single message, but in the aggregated behavior in a certain time-line. The AAL (Automated AnaLysis) component was developed in order to automatize routine system checks, to aggregate and correlate error messages and monitoring data coming from different information sources, to detect complex event patterns over time and finally to provide TDAQ operators with more pertinent and meaningful information. AAL guide the shift crew operators and experts in their daily work, for this reason it is also known as *The Shifter Assistant*.

Architecture: The high-level architecture of AAL component is shown in Figure 2. Information updates and application messages received from different sources via the monitoring infrastructure are injected into the AAL engine as streams of events with certain properties. Directives (rules) developed by experts are loaded into the engine at start-up and applied to the streams of events as they come in, in real-time manner. As result, engine produces *alerts* which are then routed to the users by different means or can be re-injected back to the TDAQ messaging system for benefit of other applications.

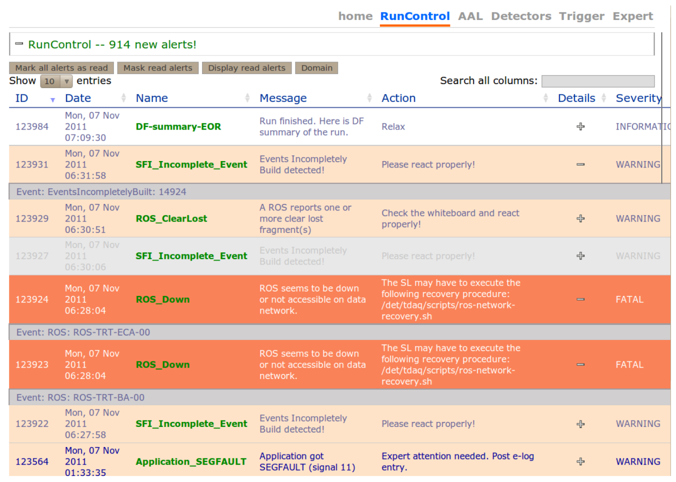
Knowledge Base: The rules implemented for the TDAQ AAL engine can be grouped in the following categories:

- selecting most important events (like error messages), correlating the messages with the experiment conditions and TDAQ system state, i.e. putting the messages into the context of ongoing activity
- reacting on absence of certain events in a time window, which indicates some system misbehavior or a service being down
- interpreting certain messages, making them more user-friendly
- aggregating over certain types of events, making statistical calculations (average, maximum, sums) over information properties
- correlating (joining) certain type of events and making diagnostics deductions
- detecting complex time-based patterns of events, like detecting repetitive bursts of events in a certain time period, or detecting sequences of events following (or not following) each other

An example of a simple rule is presented below. It checks if the value of information object `DFM-1.deadtime` averaged in last minute is exceeding a given threshold. When the condition occurs, an alert is sent to `TDAQ.RunControl` domain and the corresponding shifter is notified that an action is to be taken.

```
<directive name="DFM-high-occupancy">
<epl>select
avg(attributes('deadtime').double) as AverageOccupancy
from
ISEvent(partition='ATLAS', name='DFM-1').win:time(60 sec)
having avg(attributes('deadtime').double) > 0.08
and ATLAS_IS_RUNNING
output first every 2 minutes
</epl>
<listener type="alert">
<domain>TDAQ.RunControl</domain>
<severity> WARNING </severity>
<message>DFM average occupancy is high
($AverageOccupancy$) in last 60 seconds</message>
<action>Investigate.
Check DAQ summary for busy source.</action>
<details>true</details>
<writer type="jms"><format>XML</format>
</writer></listener>
</directive>
```

At present, AAL KB contains more then 100 rules or directives covering not only TDAQ areas, but also ATLAS subdetector areas. The deployment of ALL allowed to suppress one of the shifter desks in the ATLAS control room: all the checks normally done by shifters at this desk were implemented as SA directives and the output was directed to other desks or experts.



RunControl -- 914 new alerts!

Show 10 entries   Search all columns:

ID	Date	Name	Message	Action	Details	Severity
123984	Mon, 07 Nov 2011 07:09:30	DF-summary-EOR	Run finished. Here is DF summary of the run.	Relax	+	INFORMATI
123931	Mon, 07 Nov 2011 06:31:58	SFI_Incomplete_Event	Events Incompletely Build detected!	Please react properly!	−	WARNING
Event: EventsIncompletelyBuilt: 14924						
123929	Mon, 07 Nov 2011 06:30:51	ROS_ClearLost	A ROS reports one or more clear lost fragment(s)	Check the whiteboard and react properly!	+	WARNING
123927	Mon, 07 Nov 2011 06:30:06	SFI_Incomplete_Event	Events Incompletely Build detected!	Please react properly!	+	WARNING
123924	Mon, 07 Nov 2011 06:28:04	ROS_Down	ROS seems to be down or not accessible on data network.	The SL may have to execute the following recovery procedure: /det/tdaq/scripts/ros-network-recovery.sh	−	FATAL
Event: ROS: ROS-TRT-ECA-00						
123923	Mon, 07 Nov 2011 06:28:04	ROS_Down	ROS seems to be down or not accessible on data network.	The SL may have to execute the following recovery procedure: /det/tdaq/scripts/ros-network-recovery.sh	−	FATAL
Event: ROS: ROS-TRT-BA-00						
123922	Mon, 07 Nov 2011 06:27:58	SFI_Incomplete_Event	Events Incompletely Build detected!	Please react properly!	+	WARNING
123564	Mon, 07 Nov 2011 01:33:35	Application_SEGFAULT	Application got SEGFAULT (signal 11)	Expert attention needed. Post e-log entry.	+	WARNING

Fig. 3. Snapshot of Shifter Assistant web page for Run Control desk, filled with alerts for the ongoing run.

Interaction with users: The information is provided to shifters on an dynamic web page in form of *alerts*, which contain precise description of a detected issue and suggested action for the shifter to take. Alerts are routed to different domains, such that each domain is presented on a dedicated page on the subsystem shifter desk in the experiment control room. A snapshot of this page with a number of alerts is shown in Figure 3. The aim of the AAL web page is to provide the TDAQ operator with shifter-friendly and action-oriented view on TDAQ operational monitoring information.
In addition, selected alerts can be routed directly to particular responsible persons e.g. experts on-call in form of e-mail or SMS messages. This guarantees maximum responsiveness of the support team in critical situations. Alerts can also be injected back to the TDAQ messaging system where they are available to other TDAQ components, for instance to OR component.

The engine of the AAL is based on a Java implementation of CEP technology ESPER, primarily developed for business processes analysis applications. Rules in CEP are independent SQL-like directives which are applied to the streams of data injected in the engine.

D. Error diagnosis and system verification: Diagnostics and Verification Framework

Understanding the root causes of a problem is an important aspect of error handling in complex controls systems. Collecting additional precise and detailed information about the status of faulty components, and also performing verification testing of the system for preventive error detection is an essential part of the error-handling procedures. A knowledge-based diagnostics and verification framework DVS was developed in order to automatize these tasks in the scope of TDAQ Control system. The framework allows configuring specific tests with different levels of details for any TDAQ component and to define test sequences and diagnostics rules for detection and diagnosis of faults of the TDAQ components. Tests for components are developed by subsystem experts and added to the framework, such that they are used later by the operators and experts willing to automate verification of the system. The framework has embedded ES (CLIPS engine), so it can also load set of rules which typically describe a) tests sequences for particular components; b) what to do in case a test fails (for instance additional tests to be launched); c) error diagnosis, by correlating tests results for different components and deducing the diagnosis, if possible or simply the log of the completed testing sequences. The outcome of the verification procedures is presented to the users in a GUI, or it is made available to the OR component which uses it to perform further recovery decisions and actions.
More details on implementation of the DVS component is given in [7]. The present test repository for ATLAS contains about 40 tests for different elements of the system, which are launched regularly to ensure the healthiness of the overall system status.

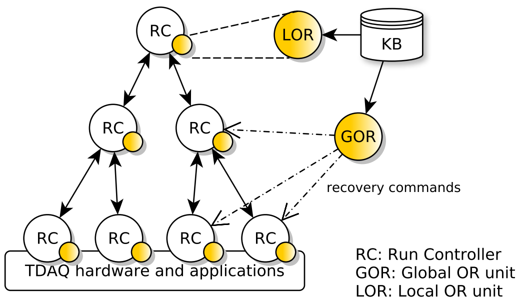


Fig. 4. Integration of OR into Run Control tree

E. Error recovery: Online Recovery in Run Control

Online Recovery system is a distributed rule-based expert system integrated into the TDAQ Run Control framework. TDAQ Run Control is organized as an hierarchical tree of Controllers and managed applications, as shown on Figure 4.

Each Controller in the tree controls a particular sub-domain (a Segment) of the whole system. It contains an embedded expert system engine which loads and executes a subset of the KB specific to that particular Segment or subsystem. The rules define local error-recovery scenarios, like reacting on error conditions, restarting or ignoring of faulty applications. In addition to local recovery units embedded in Controllers, a top-level OR component is implemented. Its task is to coordinate recovery scenarios which can not be handled on local level and involves actions across different subsystems. OR also automates different important routine actions otherwise performed by the Run Control shifter manually.
Examples of the implemented recovery scenarios are: restarting and disabling of failing applications, trigger clock changes, resources reallocation, stop-less disabling of faulty parts of the system read-out, stop-less re-inclusion of recovered components. OR uses testing capabilities of DVS in order to execute specific tests for re-assuring the error conditions or for confirming the status of the components after completion of the recovery actions.
An example of an OR recovery rule is listed below:

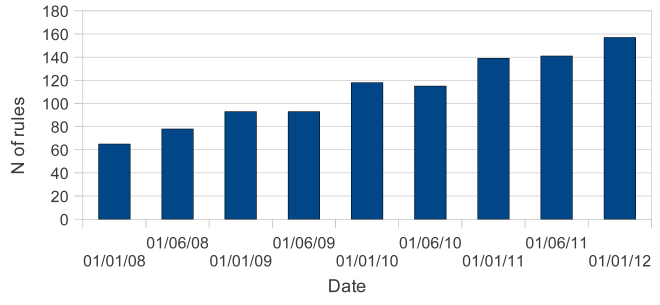


Fig. 5. Growth of OR knowldge base size during the last years of deployment and maintanance.

```
(defrule l2sv-died
  (declare (salience ?*salience_highest*))
  ?problem<-(object (is-a PROBLEM)
   (TYPE APPLICATION_DIED_UNEXPECTEDLY)
   (OBJECT ?name)(HANDLED FALSE))
  ?l2sv <-(object (is-a OR-APPLICATION)(UID ?name)
   (CLASS-NAME "L2SVApplication")
   (RUNS-ON ?host)(MEMBERSHIP IN))
  ?ctrl <-(object (is-a CONTROLLER)(STATE RUNNING)
   (TRANSITION NONE)
   (RC-COMMAND ~TERMINATE &~SHUTDOWN ))
  =>
(ers-log "[L2SV-Recovery::l2sv-died] L2SV '" ?name "'
 on host " ?host " is dead. Recovery action is being taken")
(l2sv-died ?l2sv)
(send ?problem put-HANDLED TRUE)
(send ?problem put-DECISION L2SV-RECOVERY))
```

The rule detects a fault of an L2SV application (a trigger supervisor, one type of TDAQ applications), and asserts other facts which make other rules to fire so the recovery actions gets completed.
At present, more then 150 rules were implemented for OR local and global components during the development and first years of maintenance of TDAQ Control system, and the size of KB was growing as shown on Figure 5. During a typical run (which lasts 10-20 hours) OR is very active, recovery actions are triggering regularly. Figure 6 shows the statistics of invocations of most frequently used rules, gathered during all runs taken in March-May of 2012. Most of the activity of OR is related to the stop-less recovery operations, which aim is to keep data-taking on-going while being recovering from unavoidable failures of different components in the complex TDAQ and detector read-out systems. Such operations are subsystem-specific, input conditions and subsystem behavior were changing frequently and the KB were modified on daily basis.

IV. Results and conclusions

Knowledge-based components are widely used in the Control software of the ATLAS TDAQ system. These components make use of rule-base Expert System and Complex Event Processing software technologies. Used all together, they assist TDAQ operators to steer the system without frequent manual interventions by automatizing control procedures, related to error detection, analysis, diagnosis and recovery.
Use of the intelligence in Control of TDAQ system substantially reduced the load on TDAQ control shift crew, and contributed a lot to the impressive TDAQ run efficiency of 95% during 2010-2011 years of system operations. About

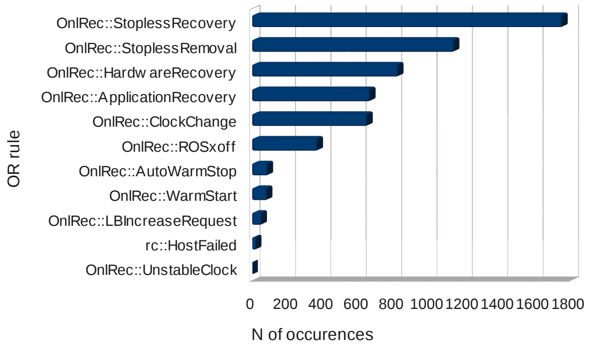


Fig. 6. Statistics of invocations of OR automation rules, during data-taking runs in March-May 2012.

half of the remaining 5% inefficiency is still coming from the manual and often mistaken interventions from the shifters, so there is still a room for improving the automation of the control operations.
Deployment of Shifter Assistant component allowed to suppress one of the shifter desks in the ATLAS control room. After the deployment of the component started, more ATLAS subdetectors expressed interest in using this approach to assist shifters on other control room desks.
The KBs used by CLIPS and ESPER-based components presently contain about 300 rules and directives, which where gathered by a small team of engineers taking the input from a dozen of experts. The chosen approach has shown to be a powerful and flexible way to automate complex error-handling operations, otherwise performed manually by TDAQ operators.

References

[1] The ATLAS Collaboration, *ATLAS high-level trigger, data-acquisition and controls : Technical Design Report* (Geneva : CERN, 2003
[2] The ATLAS Collaboration, *The ATLAS experiment at the CERN Large Hadron Collider* J.Instrum. 3 S08003, 2008
[3] Lehmann Miotto G et al. *Configuration and control of the ATLAS trigger and data acquisition* NIMA 623 549-551, 2010
[4] Gary Riley *CLIPS: A Tool for Building Expert Systems*. Available: http://clipsrules.sourceforge.net/.
[5] Luckham, D. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Boston, MA : Addison-Wesley, 2002. - ISBN 978-0-201-72789-0
[6] Bernhardt T. *Where Complex Event Processing meets Open Source*. Available: http://esper.codehaus.org/
[7] Kazarov A, CorsoRadu A, Lehmann Miotto G, Sloper J.E., Ryabov Y. *A rule-based verification and control framework in ATLAS Trigger-DAQ*. IEEE Transaction on Nuclear Science, 54 (2007) 604-608.