CONTROL IN THE ATLAS TDAQ SYSTEM

D. Liko, D. Burckhart-Chromek, J. Flammer, M. Dobson, R. Jones, L. Mapelli,
CERN, Geneva, Switzerland

I. Alexandrov, S. Korobov, V. Kotov, M. Mineev,
Joint Institute for Nuclear Research, Dubna, Russia

A. Amorim, N. Fiuza de Barros, D. Klose, L. Pedro,
Universidade de Lisboa, Faculdade de Ciencias (FCUL- CFNUL), Lisbon, Portugal

E. Badescu, M. Caprini,
National Institute for Nuclear Physics and Engineering, Bucharest, Romania

S. Kolos*,
University of California, Irvine, USA

A. Kazarov, Y. Ryabov, I. Soloviev,
Petersburg Nuclear Physics Institute (PNPI), Gatchina, St. Petersburg, Russia

ATL-DAQ-2004-013
05 October 2004

* On leave from Petersburg Nuclear Physics Institute (PNPI), Gatchina, St. Petersburg, Russia

Abstract

The unprecedented size and complexity of the ATLAS TDAQ system requires a comprehensive and flexible control system. Its role ranges from the so-called run-control, e.g. starting and stopping the data taking, to error handling and fault tolerance. It also includes initialization and verification of the overall system. Following the traditional approach a hierarchical system of customizable controllers has been proposed. For the final system all functionality will be therefore available in a distributed manner, with the possibility of local customization.

After a technology survey the open source expert system CLIPS has been chosen as a basis for the implementation of the supervision and the verification system. The CLIPS interpreter has been extended to provide a general control framework. Other ATLAS Online software components have been integrated as plug-ins and provide the mechanism for configuration and communication.

Several components have been implemented sharing this technology. The dynamic behavior of the individual component is fully described by the rules, while the framework is based on a common implementation. During this year these components have been the subject of scalability tests up to the full system size. Encouraging results are presented and validate the technology choice.

INTRODUCTION

The ATLAS Online Software [1] encompasses the software to configure, control, and monitor the TDAQ system but excludes the management, processing, and transportation of physics data. It is designed as an object-oriented framework based on industrial standards (CORBA, XML, etc.). It follows an iterative development cycle: a first prototype system has been developed and was validated in several test beams and scalability tests [2]. In the context of the preparation of the ATLAS TDAQ TDR [3] high level requirements and architectural choices have been reexamined [4,5].

The Online Software architecture is based on a component model and consists of three high level packages, Control, Databases and Information Sharing. The Control package contains in turn sub-packages for the control of the TDAQ system and detectors. Control sub-packages exist to support TDAQ system initialization and shutdown, to provide control command distribution, synchronization, error handling, and system verification.

To complement the high level design a TDAQ Wide Run Control has been set up. This group investigated practical aspects of the integration between the Online Software and the relevant TDAQ systems: High Level Trigger and Dataflow. The Controller is considered as the fundamental building block of the system and its envisaged functionality has been described in detail [6].

THE CONTROL SUBSYSTEM

The TDAQ system is a large and heterogeneous system composed of a large number of items to be controlled. Typically these items are clustered and range from readout modules in VME crates to workstations within HLT computer farms. Such clusters are the preferred places to interface with the Online Software Control system. The number of these clusters is estimated to be in the range of 500–1000. To control these units the TDAQ control system is built in a hierarchical and distributed manner.

The Control package is divided into a number of sub-packages as shown in Figure 1. Its functionality has been distributed between several distinct sub-packages:

- The *User Interface* for interaction with the operator;
- The *Supervision* for the control of the data-taking session including initialization and shutdown, and for error handling;
- The *Verification* for analysis of the system status;
- The *Process Management* for the handling of processes in the distributed computing environment;
- The *Resource Management* for coordination of the access to shared resources;
- The *Access Management* for providing authentication and authorization when necessary.

In the following the only the Supervision sub-package is discussed.

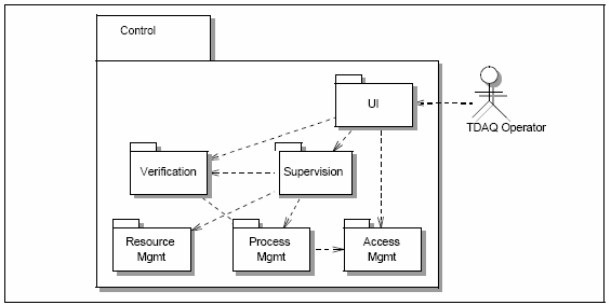

Figure 1: The organization of the control package

Supervision

The *Supervision* sub-package realizes the essential functionality of the Control package. A system will generally contain a number of controllers organized in a hierarchical tree, one controller being in control of a number of others in the system while being controlled itself from a higher level controller. One top level controller, called the root controller, will take the function of the overall control and coordination of the system. The User Interface sub-package provides all TDAQ control facilities to the Operator.

The *Initialization and Shutdown* is responsible for:

- initialization of TDAQ hardware and software components, bringing the TDAQ partition to the state in which it can accept Run commands;
- re-initialization of a part of the TDAQ partition;
- shutting the TDAQ partition down gracefully;
- TDAQ process supervision.

The *Run Control* is responsible for

- controlling the Run by accepting commands from the user and sending commands to TDAQ sub-systems;
- analyzing the status of controlled sub-systems and presenting the status to the Operator.

The *Error Handling* is concerned with

- analyzing run-time error messages coming from TDAQ sub-systems;
- diagnosing problems, proposing recovery actions to the operator, or performing automatic recovery if requested.

TECHNOLOGY CHOICE

Prototype evaluations have been performed for a number of technologies. The initial choice for our prototype had been based on experience in previous experiments. Products were chosen that fit well in the proposed object-oriented software environment.

- The *Run Control* implementation was based on a State Machine model and used the State Machine compiler, CHSM [7], as underlying technology.
- A Supervisor is mainly concerned with process management. It had been built using the Open Source expert system CLIPS [8].
- A verification system (DVS) performs tests and provides diagnosis. It was also based on CLIPS.

It has to be pointed out that verification of the prototype in scalability tests demonstrated successfully that the system was capable of controlling up to the expected system size [2]. Nevertheless several shortcomings of the current system were identified. An important one is the lack of flexibility in the state machine implementation CHSM.

Evaluations

Due to our positive experience our evaluations concentrated on the expert system CLIPS and related products. The general purpose nature of this product gives the possibility to use it for all control aspects. The knowledge base provides the basis for a customizable solution, which can be specialized for different parts of the system. Another advantage is the extensibility of CLIPS. It can be interfaced with other components of the Online Software system in a straight forward way. There are also alternative products that follow the CLIPS model as Jess [9], an implementation written in Java and a commercial alternative, Eclipse by Haley Inc. [10].

Further possibilities have been investigated: SMI++ is a system that models the controlled domain as a system of interacting state machines [11] and is in use at several HEP experiments. The architectural approach of this product does not fit well to our component model. Another possibility would be the use of general-purpose scripting languages, such as Python [12]. Its advanced capabilities make such a choice tempting, but some general expert system or event management capabilities would have to be implemented. While each of these approaches has its particular merits, our evaluation finally favored a CLIPS-based solution.

In conclusion the original choice the expert system CLIPS was confirmed. It was found to be an advantage to base the whole Control package on this technology.

DESIGN AND IMPLEMENTATION

In TDAQ Wide Run Control Working group a general consensus was reached to design a flexible Controller framework that unifies all control functionality in a hierarchical system. User customization would provide means to adapt the required functionality in accordance with the TDAQ sub-system in question.

In the iterative development model such a design should developed as an evolution of the prototype system. The component model allows exchanging individual parts without breaking the overall system. It was decided to implement a common framework based on CLIPS. In a first iteration the Run Control and the Supervision component would be implemented on top of this framework[†].

Framework

The CLIPS interpreter has been extended to provide a general control framework. It has been embedded in a common CORBA server. A thread-safe mechanism for the periodic evaluations of the knowledge base has been put in place. Other ATLAS Online software components have been integrated as plug-ins and provide the mechanism for configuration and communication.

The complex interfaces between CLIPS and C++ is confined to the plug-ins. All Online Software functionality is available to the expert system by the mechanism of associating function calls to external C++ functions. Also CORBA callbacks can be associated with CLIPS user functions or CLIPS object message handlers.

In a typical implementation such CLIPS objects would be used to implement proxies representing other elements of the TDAQ system. The member attributes of these objects are available to the knowledgebase as facts. As a typical production system rules are then used to model the interaction between these objects. These rules monitor the status of the objects and provide the dynamic behavior of a controller. The Rete algorithm [13], that is the basis of CLIPS, ensures efficient evaluation,

Components

The run controller and the supervisor have been implemented using the CLIPS scripting language[‡]. As significant advantage to the previous implementation the problem specific domain is fully implemented in the knowledge base. This clear separation of CLIPS and C++ improves the maintainability of the components.

This approach can be compared with an implementation of a control system following the state machine model. The CLIPS solution profits from powerful paradigm of an expert system and the versatility of the CLIPS general purpose scripting language. On the negative side one can identify the complexity of the language bindings and the mechanism of object orientation.

[†] It has to be pointed out that other Control components are based on CLIPS technology. A smooth migration path can be envisaged in the future.

[‡] In addition Run Controllers have been extended by a common multithreaded implementation that interacts with several controlled items.

TESTS AND VERIFICATION

Scalability test

The aim of the tests [14] was to study the interaction between the components, to identify critical areas and to investigate the variation and optimization of online system parameters. The timing values of the steps which lead through the various data acquisition phases were recorded and analyzed.

Up to 330 PCs of the CERN IT LXSHARE test bed [15] were used. They were equipped with 600 to 800 MHz to 2.4 GHZ Dual Pentium III processors, 256 to 512 MB of memory running the Linux RedHat 7.3 operating system with selected system parameters adjusted to the needs of the tests. As there was no dedicated common file system such as NFS available for the PC's, the Software was replicated on the local disk of each PC. The tests were performed using several configurations:

- *mon_standard*: all the controller segments were evenly distributed over all nodes,;
- mon_server: configuration a number of central servers was run on dedicated 2.4 GHz PCs;
- *mon_server_rdb*: the configuration information was accessed by a new implementation of a remote database sever;
- *mon_local* the load of servers was decreased by additional servers for the individual detector partitions

Figure 2 shows a typical result of the measurements. The *boot* transition shows the time necessary to start up the system from the idle state to the active state. All processes are started up and configured. For a configuration close to the size of the expected system (100 controllers) times between 40 to 100 seconds were measured. As such an initialization is only performed at the start of data taking periods, the result was considered satisfactory.

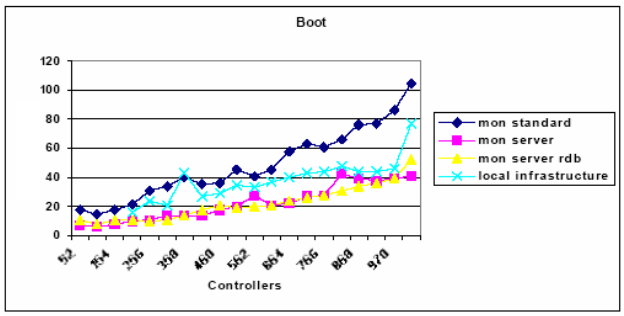

Figure 2: Boot transition for the different partition sets

Figure 3 shows the time necessary to perform the so-called *lukewarm stop* transition. This transition is typical for the run control operation necessary to start and stop of the data taking. The selected transition passes through 7 internal phases with synchronization of all controllers. In the measurements only the communications overhead introduces by the Online Software system has been measured. In a real life system the actual operations in the

Dataflow and High Level trigger system are expected to be an order of magnitude higher. Therefore the time measured to stop the system, between 2.5 to 6 seconds has also been considered satisfactory [§].

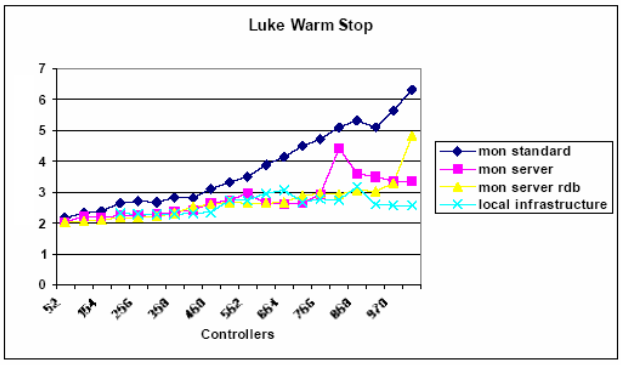

Figure 3: Luke Warm stop transition

In summary the new implementation of the Run Control and Supervision components based on a common CLIPS framework has proven to be of satisfactory quality and performance to control systems up to the size of the expected ATLAS TDAQ system.

Test beam

Soon afterwards the new Supervision framework was also used in this years combined test beam. For this test beam a true slice of the final ATLAS TDAQ system has been configured which includes all sub-systems. It has been reported in more detail at this conference. Due to the component nature of our software it was possible to exchange the related software components in a transparent way. The new components operated as expected.

CONCLUSIONS

The control package is an essential part of the ATLAS Online Software. Starting from the user requirements to the overall architecture the system has been iterated following the iterative software development cycle. The technology choice of the CLIPS expert system has been confirmed and its application extended to all aspects of the Control package. A new implementation of Control software components has been verified in a scalability test. The increased flexibility introduced with the implementation should provide an excellent basis for the development of an extended controller that will unify all control functionality in one component.

[§] The observed two seconds offset is due to an implementation artifact that has been identified and will be eliminated in a future release of the software.

ACKNOWLEDGEMENTS

The authors thank the other members of the TDAQ Wide Run Control group, Giovanna Lehman, Sarah Wheeler-Ellis and Haimo Zobernig for their engagement in the design of a better controller for the ATLAS TDAQ system. We would also express our gratitude to the ATLAS Dataflow and High Level Trigger systems for excellent collaboration.

REFERENCES

[1] *Atlas Online Software*, http://cern.ch/atlas-onlsw .
[2] D. Burckhart et al., *Large Scale and Performance Tests of the ATLAS Online Software*, Proceedings of CHEP 01, Beijing, (2001).
[3] *ATLAS High Level Trigger, Data Acquisition and Control TDR*, CERN/LHCC/2003-022, (2003), http://cern.ch/atlas-proj-hltdaqdcs-tdr .
[4] *Online Software Requirements*, Internal Note (2002) http://cern.ch/Atlas-onlsw/documents/doc/OnlSw_Req_05.pdf .
[5] *Online Software Architecture*, Internal Note, (2002) http://cern.ch/Atlas-onlsw/documents/doc/OnlSWArchitecture_03.pdf .
[6] *TDAQ Wide Run Control group, Controller Requirements*, ATL-DQ-ES-0054, (in preparation).
[7] P.J. Lucas, *An Object-Oriented language system for implementing concurrent hierarchical, finite state machines*, MS Thesis, University of Illinois, (1993).
[8] *CLIPS, A tool for building expert systems*, http://www.ghg.net/clips/CLIPS.html .
[9] *Jess, the Java Expert System Shell*, http://herzberg.ca.sandia.gov/jess .
[10] *Eclipse, Rule based programming language and inference engine*, see under 'products' on http://www.haley.com .
[11] *SMI++, State Management Interface*, http://cern.ch/smi .
[12] *Python, interpreted, interactive, object-oriented programming language* http://www.python.org .
[13] C.Forgy, *Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem*, Artificial Intelligence, 19, (1982).
[14] Large Scale Performance Tests March 2004, Internal Note, (in preparation).
[15] V. Bahyl et al., *Experience constructing and running large shared clusters at CERN*, Proceedings of CHEP 01, Beijing (2001).