THE RUN CONTROL IN THE ATLAS PROTOTYPE-1 DAQ/EF PROJECT

D. Schweiger, Institute f. Experimental Physics, Innsbruck, Austria / CERN, Switzerland
R. Jones, L. Mapelli, CERN, Switzerland
P.-Y. Duval, Centre de Physique des Particules, Marseille, France
A. Kazarov, S. Kolos, Y. Ryabov, PNPI, Gachina, St. Petersburg, Russia

Abstract:

The Run Control (RC) for the Atlas detector in the Atlas Data Aquisition/Event Filter (DAQ/EF) Prototype-1 Project is the component that coordinates the operation of the DAQ and provides the user interface to the shift operator. The current distributed object oriented RC implementation is presented. Recent work and future plans on integrating autonomous failure handling and diagnosis is discussed.

1 INTRODUCTION

The ATLAS data acquisition (DAQ) and Event Filter (EF) prototype-1 project [1] is intended to produce a prototype system representing a ‘‘full slice” of a DAQ suitable for evaluating candidate technologies and architectures for the final ATLAS DAQ system on the LHC accelerator at CERN. Within the prototype project, the back-end sub-system encompasses the software for configuring, controlling and monitoring the DAQ but specifically excludes the management, processing or transportation of physics data. The back-end software must coexist and co-operate with the other sub-systems. The RC is a central part of the backend software.

2 ARCHITECTURE

The RC is defined as the component that guides the DAQ through the startup and shutdown phase and takes care of keeping the system in the running status. It provides the interface between the shift operator and the components that make up the DAQ/EF system. The RC uses the following basic backend components as depicted in Figure 1:

- Configuration Database (CONFDB): Object oriented persistent storage management system with corresponding schemes to model hardware and software usage for coping with the required high configurability of the DAQ especially in respect to the testing activities of the detector. An access library layer provides transparence in respect to the used storage manager (OKS, Objectivity) [2]
- Inter Process Communication (IPC): Layer to provide independence of communication from the used CORBA ORB and supports the partitioning scheme needed to be able to run several DAQ sessions in parallel without interference (comparable to namespaces) [3].
- Information Service (IS): client server based means of storing and retrieving information based on IPC (e.g. controller states)
- Message Reporting System (MRS): client server based means of distributing messages based on IPC (comparable to the mailing list scheme)
- Process Manager (PMG): component for starting/ stopping monitoring processes corresponding to software applications found in the CONFDB on distributed hosts with defined environment and parameters.

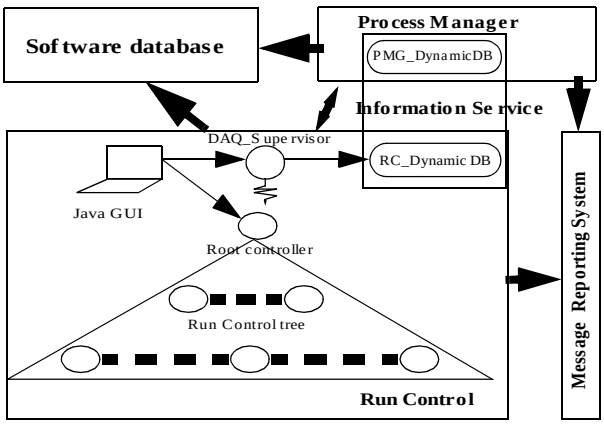

Figure 1: The RC in the context of Backend Components

The central component of the RC is the RC-controller, a customizable entity that inherits a Finite State Machine (FSM) with CORBA based communication abilities from a skeleton. The FSM, in C++, was generated using Harel State Charts with the CHSM package [4]. It provides hierarchical concurrent sets of states to model the status of any DAQ-component in a uniform way. For individual components a specific controller can be built using inheritance and overloading of the action methods corresponding to the state changes of the FSM.

The whole RC is made up by a hierarchical distributed tree of controllers via a parent/child relationship that models the logical decompositon of the DAQ system as defined in the CONFDB. The controller at the top of the hierarchy (root controller) is , in the general case, the point of interaction with the operator via a Java graphical user interface. Commands issued there are conveyed through

the tree towards the actual domain specific controllers. The performance of the communication was found to be acceptable and scalable even for a large setup of 250 controllers as expected for the final system [5]. A controller with TCP socket based communication has been added recently for components that can't use the CORBA ORB, as the case for the data flow component.

The second major element of the RC is the RC-Supervisor. Currently its task is to set up and shut down the DAQ i.e. the starting/termination and supervision of the processes as listed in the CONFDB taking into account dependencies between processes using the PMG.

The IS component is used by the RC to publish the FMS states of the controllers for synchronisation purposes. MRS messages are being sent mainly from the component controllers as a means of publishing domain specific events to be displayed to the operator or to be used for automated diagnosis procedures.

The system as described above has been successfully used to control the DAQ setup of the June99-Milestone that integrated the backend with the data flow components of the DAQ.

3 HANDLING FAILURES

One of the last remaining challenges in the scope of the backend software is the handling of unexpected situations. Historically in High Energy Physics (HEP) this problem was attacked after the actual start of data taking for most existing experiments. The aim of systems alike was mainly to reduce human expert presence during the shift, reduce knowledge loss due to staff turnover, and reduce down time by speeding up the recovery action by means of automation.

3.1 Simple Error Handling Scheme

The Atlas Run Control can work without such a component as there are no real time requirements in respect to fault recovery. A simple error handling scheme included in the skelleton controller takes care of errors:

- If a controller (or controlled component respectively) falls into the Fault state during data taking the whole tree will exit the data taking state. This minimizes the effect of failures to other components (failure propagation).
- A failure of a controller at any state will be reported up the tree to the root controller making it visible to the operator and locking certain transitions of the FSM. The manual recovery isolates the faulty parts of the tree (logical subsystems) and permits their recovery by repeating the initialisation or resetting the subsystem. After arriving in a good state the tree can be relinked and the system can be guided towards the running state. This is preferred as it is too expensive in terms of down time to reset the whole DAQ system for every failure occurence.

With this tree approach and via the graphical user interface the operator gets a comprehensive and detailed picture of the system status with uniform interaction points making it easy to control the system. MRS error messages give additional hints.

3.2 Intelligent Error Handling Schemes

The same applies to an automated approach that is eased considerably by this well defined uniform structure. A component for this task has been foreseen in the DAQ-1 project but the design was left open, so that it could emerge more recently.

The so called Diagnostics package comprises two components: The Verification Component (VC) deals with complex tests to aid failure diagnosis mainly when the DAQ is not taking data.

The Supervision Component(SC) is a intelligent extension for the Supervisor as described above. It acts either as an intelligent agent, autonomously performing reasoning in case of failures and adequate recovery actions or gives aid to the human operator. Both the VC and the SC use the C-Language Embedded Production System (CLIPS), a rule based language for building expert system (ES) [6]. Rule based systems have been successfully used in control and diagnosis in the domain of HEP data aquisition inside the DEXPERT system for the ALEPH experiment [7].

The main advantage for these domains is the relative ease of extending and integrating knowledge(about failure cases) compared to conventional programming languages. This is because the rules (basically if <expression> then <action> statements) are not arranged in a rigid control structure (i.e. decision tree) but rather that the rules act concurrently on the available data (facts). Rules and facts make up the knowledge base. But the unpredictable flow of control becomes a problem for a large number of rules (more than several hundreds) [8].

The current basic implementation of the SC maintains the functionality of the Supervisor, that is the FSM describing the DAQ status from the process view as described above. The state machine is implemented by means of the State pattern [9], using the object oriented programming extension of CLIPS. The rules are additionally grouped in several autonomous knowledge bases using the module structure of CLIPS to ease the flow control and making the rulesets more manageable. The current modules which correspond to the states of the FSM are being handed the control on the common data by the rules in an extra module acting as a controller within a Blackboard Architecture. The shared information that is defined and owned by the controlling module is:

- IS information (e.g. controller status)
- MRS messages (e.g. notification of faults from components)
- PMG messages i.e. process creation/death notifications
- Configuration information for the individual hard-

ware and software setup.

This design now provides the skeleton that can be extended as more knowledge is aquired by adding rules to the corresponding modules. Rules for the booting and shutdown phases of the DAQ are mainly concerned with processing the creation/termination of processes according to the chosen setup in the CONFDB and problems related to configuration inavailability of resources as network, filesystems, hardware and basic software components. The part of the knowledge base for when the RC is active is the most challenging. Here the subsystem specific failures have to be handled.

Our first simple approach to handling failures in subsystems is to follow the scheme depicted for the manual interaction above: perform a deep first search through the tree to find the erronous component controllers, isolate them and try to get them back into a good state by exersising statechanges. This generic approach lacks any knowledge of the functionallity of the controlled components in respect to diagnosing the fault propagation. A primary fault will often cause cascading faults in several components. This problem can be reduced by grouping dependent components accordingly in the tree so that failures stay mainly inside this group's boundaries. This way actions on the corresponding group controller can still resolve the problem and synchronisation of state changes is taken care of by the knowledge about its subsystems inherent in this controller.

4 FUTURE WORK

A more advanced scheme would take component specific knowledge into account. This can be done in a classical approach by using directly the rule based language to insert knowledge of known or expected problems as done in DEXPERT. The rules would have to be configuration independent which requires some additional information inside the CONFDB e.g. showing the relationship of a controller to its controlled component. But, as stated above, rule based ES become difficult to maintain for large systems. Model based reasoning became very comon in the field of diagnostics and maintenance field [8]. In the model based approaches, e.g. the fault digraph model the knowledge of system behaviour in respect to faults is represented as a network of nodes representing the functional components connected by propagation paths of errors. The main advantage of this approach is that knowledge presentation is deep compared with the rule based systems. This means that rules are rather an indirect or implicit presentation of the knowledge about a system. With the very intuitive (graphic) system knowledge representation in model based reasoning it is easier to maintain and keep the knowledge consistent and comprehensive. The OO character makes it easily configarable. Not every failure pattern has to be stated explicitely. The fault digraph analysis tool EDNA has been successfully implemented using the CLIPS rule based system [10]. Hence it seems to be an approach worth studying to enhance the capabilities of the SC in terms of maintainability and user acceptance.

5 SUMMARY

The Run Control in the DAQ/EF Prototype-1 project has shown that it can be successfully integrated with other DAQ components. The rule based approach has shown to be a suitable method for adding knowledge for a advanced failure handling. For the future we would like to investigate on model based approaches in order to improve knowledge manageability and consequently user friendliness.

REFERENCES

[1] G. Ambrosini et al., ``The ATLAS DAQ and Event Filter prototype ``-1" project", Computer Physics Communications, vol. 110, pp. 95-102, May 1998.

[2] R. Jones, I. Soloviev, ``Configuration Databases in the ATLAS Prototype DAQ", CHEP'98, Chicago USA, September 1998, http://www.hep.net/chep98/PDF/66.pdf

[3] S. Kolos et al., ``Applications of CORBA in the ATLAS prototype DAQ", Proceedings of the IEEE Real Time Conference, Santa Fe, USA, June 1999, http://atddoc.cern.ch/Atlas/DaqSoft/components/ipc/RT40.ps

[4] P. Croll et al., ``Use of Statecharts in the Modelling the Dynamic Behaviour of the ATLAS DAQ Prototype-1", IEEE Transactions on Nuclear Science, vol. 45, no. 4, pp. 1983-1988, August 1998

[5] I. Alexandrov et al., "Performance and Scalability of the Back-end sub-system in the ATLAS DAQ/EF Prototype", proceedings of the IEEE Real Time Conference, Santa Fe, USA, June 1999, http://atddoc.cern.ch/Atlas/Conferences/RT99/RT175.ps

[6] J. Giarratano, G. Riley, "Expert Systems: Principles and Programming", PWS-Kent, Boston, Mass., 1989

[7] P. Mato, "DEXPERT: an Expert System for Read-Out Error Recovery in the Aleph Data Acquisition System", pp. 537-542, Second International Workshop on Software Engineering, Artificial Intelligence and Expert Systems in High Energy and Nuclear Physics, L'Agelonde, France, January 13-18, 1992

[8] M. Ben-Bassat et al., "Different approaches to diagnostic modeling", pp. 175-186, IEEE AUTOTESTCON 98 Proceedings, IEEE Systems Readiness Technology Conference, Salt Lake City, UT, USA, 24-27 Aug. 1998

[9] E. Gamma et al., "Design Patterns", Addison Wesley, ISBN 0-201-63361-2

[10] V.V.Dixit, "EDNA: expert fault digraph analysis using CLIPS", pp. 118-124, Proceedings of the first CLIPS Conference, NASA Conference Publication 10049, Houston, Texas, 1990