Online Software for the ATLAS Test Beam Data Acquisition System

2003 IEEE Real Time Conference

I. Alexandrov[1], A. Amorim[2], E. Badescu[3], M. Barczyk[4], D.Burckhart-Chromek[4], M. Caprini[3], J. Da Silva Conceicao[4], J. Flammer[4,5], B. Di Girolamo[4], M. Dobson[4], R. Hart[6], R. Jones[4], A. Kazarov[7], S. Kolos[7], V. Kotov[1], D. Klose[2], D. Liko[4], J. Lima[2], L. Lucio[4], L. Mapelli[4], M. Mineev[1], L. Pedro[2], Y. Ryabov[7], I. Soloviev[7], H. Wolters[2]

***Abstract* — The Online Software is the global system software of the ATLAS Data Acquisition (DAQ) System, being responsible for the configuration, control and information sharing of the ATLAS DAQ System. A Test Beam facility offers the ATLAS detectors the possibility to study important performance aspects as well as to proceed on the way to the final ATLAS DAQ system. Last year, three sub-detectors of ATLAS as well as a combined setup of the three sub-detectors were using successfully the Online Software for the control of their data taking. In this paper, we describe the different components of the Online Software together with their usage at the ATLAS Test Beam.**

I. Introduction

The ATLAS experiment [1] is one of four experiments at the Large Hadron Collider (LHC) particle accelerator that is currently being built at CERN and is scheduled to start data taking in 2007. Its main goals are precision test of the Standard Model and the discovery of new physics signatures beyond the Standard Model.

The bunch crossing frequency of LHC will be ~40MHz with an average of ~25 events per collision. To cope with this high number of simultaneous events, the ATLAS detector has been designed as a high granularity spectrometer having ~10^8 detector channels. Both the high bunch crossing frequency, as well as the large amount of ATLAS detector data of ~1.5 MB per event, requires the design of a very efficient Data Acquisition (DAQ) with a three level trigger system. The first level trigger operates at the full rate of 40MHz, reducing the rate down to 100 kHz. The second level trigger will reduce the rate further down to a rate of ~1kHz before the third level trigger (Event Filter) reduces the rate to the final rate in the order of 200 Hz at which the data is put to permanent storage [2].

[1] Joint Institute for Nuclear Research, Dubna, Russia
[2] Science University of Lisbon (FCUL), Lisbon, Portugal
[3] Institute of Atomic Physics, Bucharest, Romania
[4] European Organization for Nuclear Research (CERN), Geneva, Switzerland
[5] Corresponding author
[6] National Institute for Nuclear Physics and High Energy Physics (NIKHEF), Amsterdam, Netherlands
[7] Petersburg Nuclear Physics Institute (PNPI), Gatchina, St. Petersburg, Russia

The Online Software is the global system software to configure and control the DAQ as well as to share information within the DAQ system, it excludes however the processing and transportation of physics data. It has interfaces to the DataFlow System (being responsible for the transportation of the data from the Readout Drivers to Mass Storage), to the different triggers as well as to the Detector Readout Crates controllers and the Detector Control System (DCS). Fig. 1 shows an overview of the DAQ system of ATLAS.

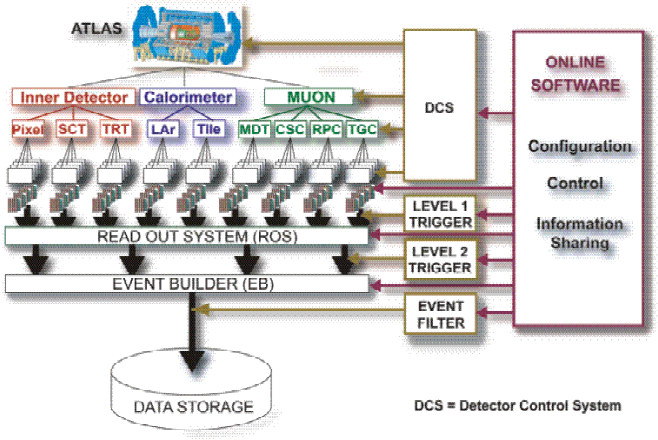

Fig. 1 Overview of the ATLAS Data Acquisition and Trigger system and the Online Software. The detector is split up in 33 sub-detector partitions that are read out by ~1000 Readout Drivers. The data is further processed by the Readout System (ROS) and the Event Builder (EB) before it is written to tape. The detector hardware is controlled by the Detector Control System (DCS) and the Online Software is responsible for the configuration, control and information sharing of the system.

II. ATLAS Test Beam

In order to study important performance aspects like alignment and calibration as well as to proceed on the way to the final ATLAS DAQ system, a Test Beam facility at the Super Proton Synchrotron (SPS) accelerator at CERN has been setup for the different detectors of ATLAS [3]. In summer 2002, three out of six sub-detectors were using the Online Software for their DAQ system to perform their data taking: the *Hadronic Tile Calorimeter (TileCal)*, the *Pixel Detector* and the *Muon Detector*. In addition, a combined data taking of these three sub-detectors has been done.

For the combined data taking, each detector was read out by a separate Readout System (ROS), who was sending the data – controlled by a common DataFlow Manager – via the SubFarm Input (SFI) to an Event Filter farm for processing and triggering. The accepted events were then send via the SubFarm Output (SFO) to permanent storage. The Online Software was used as the global baseline software of the Test Beam DAQ, providing all the necessary functionality for the configuration, control and information sharing and giving the detectors the possibility to incorporate their specific needs (e.g. specialized controllers). Fig. 2 shows the setup of the DAQ system at the Test Beam for the combined data taking.

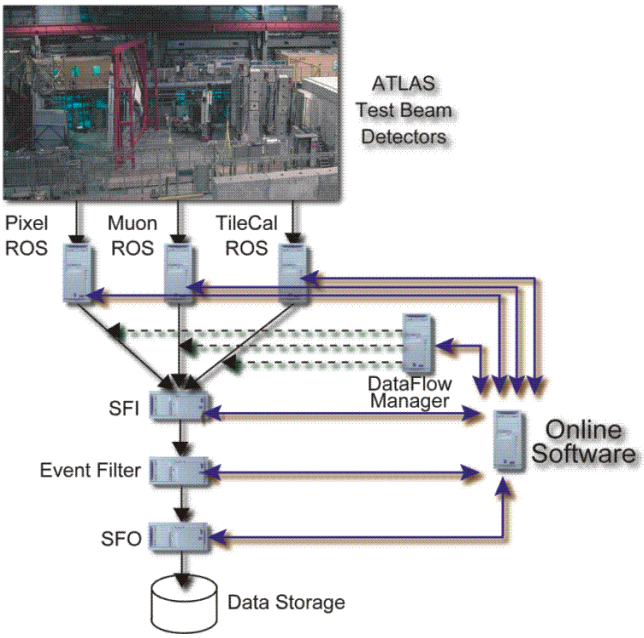

Fig. 2 Setup of the DAQ system at the Test Beam for the combined run of the Hadronic Tile Calorimeter, the Pixel and the Muon Detector in 2002.

III. Online Software

As the Online Software is the global system software, offering all the functionality to configure and control the DAQ as well as to share information in the DAQ, it has to be able to start, stop and synchronize in the order of 2000 programs on as many processors. It is build up as a customizable framework having no detector specific components in order to be able to interface to all the various sub-systems and to be used by all the various configurations of the DAQ [4].

The Online Software consists out of three main packages – *Configuration, Control* and *Information Sharing* – which in terms contain several components. Table 1 gives an overview of the different Online Software packages and their components. The full functionality of the Online Software was available for the Test Beam DAQ system and used by the three sub-detectors.

IV. Configuration

The DAQ needs a large number of parameters in order to describe its architecture, hardware and software components, running modes as well as its running status. Additionally it has to be as flexible as possible and externally parenthesized by databases describing its setup.

TABLE I
Online Software packages and their components

Configuration
Configuration Database
Online Bookkeeper
Control
Run Controller
DAQ Supervisor
Process Manager
Resource Manager
Graphical User Interface
Information Sharing
Information Service
Message Reporting Service
Online Histogramming Service
Event Monitoring Service

To store and access the configuration information of the data taking as well as the conditions during data taking, the Online Software offers two persistent services: For the configuration of the DAQ, a *Configuration Database*, for the parameters during the data taking an *Online Bookkeeper (OBK).*

The Configuration Database uses data object models to describe the configurations and is based on the OKS package. The database can be accessed by the user via a graphical user interface to define, view and modify configurations. Applications can access the database via a Data Access Library (DAL) that hides the databases implementation details [5].

Each Test Beam detector was using the database schema to describe their applications and hardware setup in a separate, independent Configuration Database. Using the DAL, they were able to access this information from their applications. In a second step, the different individual databases were combined into one common database to describe the common setup for the combined data taking of the three detectors. Fig. 3 shows a part of the database for the combined run in the graphical configuration database editor of the Online Software.

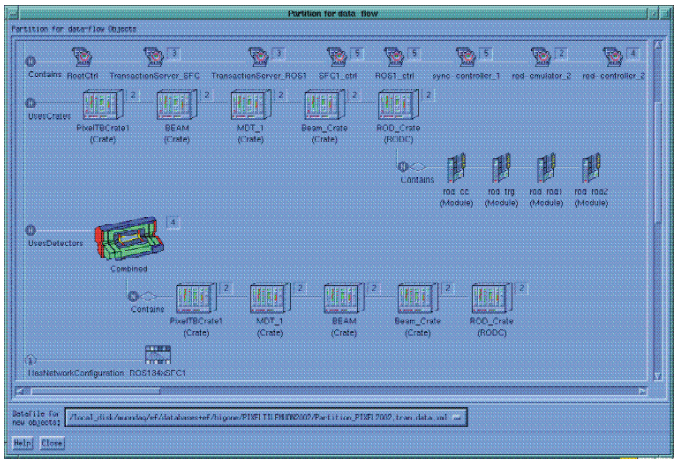

Fig. 3 The graphical database editor of the Online Software showing a part of the database for the combined data taking of the Test Beam.

The *Online Bookkeeper*, to store the relevant operational information and configuration description during data taking, organizes the data internally on a per-run basis and provides querying-APIs as well as a graphical user interface to browse and append information. The TileCal and Pixel detector were using the OBK to record their run information – both for the parameters during running as well as to add comments and histograms. Using the OBK, they were able to access the information immediately after the run via the web which proved to be very important for the analysis. Fig. 4 shows the graphical user interface of the mySQL OBK implementation [6].

Fig. 4 The Online Bookkeeper graphical user interface to browse and append operational information and configuration description.

V. Control

The control package of the Online Software supplies all the necessary control and supervision of the data taking by coordinating the different DAQ subsystem and detector operations from user interactions over initialization and shutdown, error handling, verification of the system status up to process, resource and access management [7].

To control the complete system in a coordinated way, a hierarchical tree structure of controllers is used with the top level *Root Controller* taking the role of the overall control and coordination of the system. Each controller level controls the next level controllers. The lowest level controllers are the *Local Controllers* being responsible for objects like readout crates. This flexible partitioning scheme also allows to exclude part of the system (e.g. a sub-detector) and to operate this part of the system in stand-alone mode for calibration or debugging. Fig. 5 shows an example of a controller tree.

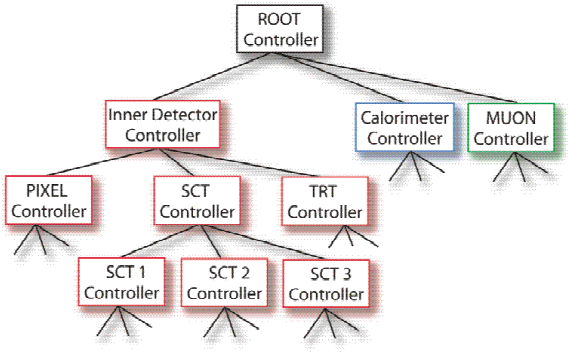

Fig. 5 Hierarchical controller tree of the ATLAS detector.

A controller in this tree is characterized by its state using a *finite state machine*. In any place of the hierarchy, a change of state is initiated from the root controller and then subsequently passed down from level to level. Four main states from initial to running are introduced. A further *Checkpoint* state allows to deal with changes in conditions, leading to a new run number without going to the stop/start sequence. Fig. 6 shows the finite state machine of the Online Software.

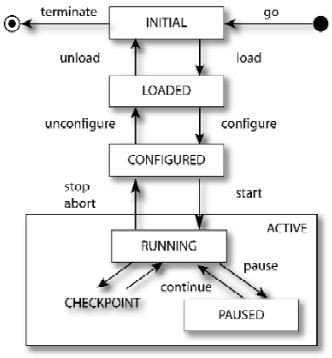

Fig. 6 The finite state machine of the Online Software Run Control.

A *run controller skeleton* is used as a basic template to provide all the necessary core functionality to control the system in a coherent way. This skeleton is adaptable to the individual needs of the different systems.

For the Test Beam, the sub-detectors were using the run controller skeleton to implement their specific controller needs. Various controller have been implemented in this way from Readout Crate controllers up to SFI controllers.

To start, stop and supervise the processes being under the control of the Online Software, the Online Software implements a *DAQ Supervisor* component. He is also able to re-initialize part of the DAQ partition when necessary.

The *Process Manager (PMG)* provides the basic process management functionality in a distributed environment. It starts, stops and monitors processes on the different DAQ hosts. To execute the requests, the PMG runs agents on all the machines and uses the information provided by the user and/or the configuration databases. The detectors at the Test Beam were using the PMG to control their specific processes running on the different hosts at the Test Beams.

A *Graphical User Interface (GUI)* allows the user to send commands to control as well as to monitor the system. To adapt for the needs of the different systems, it allows to add specialized panels. For the Test Beam, the sub-detectors were implementing several panels to be able to configure their setup (e.g. to define calibration files) as well as to receive and visualize information, their applications were publishing. Fig. 7 shows the Graphical User Interface of the Online Software.

VI. Information Sharing

There is a wide field of information sharing within the DAQ system like synchronization between processes, error reporting, operational or physics event monitoring. The Online Software offers several services for information sharing.

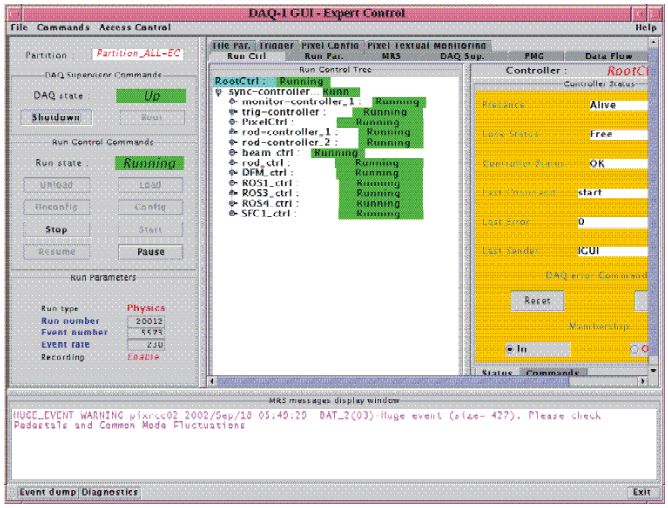

Fig. 7 Graphical User Interface of the Online Software.

The information sharing of the Online Software is based on a three layer architecture: The basic communication layer uses the *Common Object Request Broker Architecture (CORBA)*. On top of this, the middle layer, the *Inter Process Communication (IPC)* implements a communication abstraction layer to support partitioning and concurrent running of the Online Software. The top layer are the implementations of the different Online Software services. Fig. 8 shows the architecture of the Information Sharing Services.

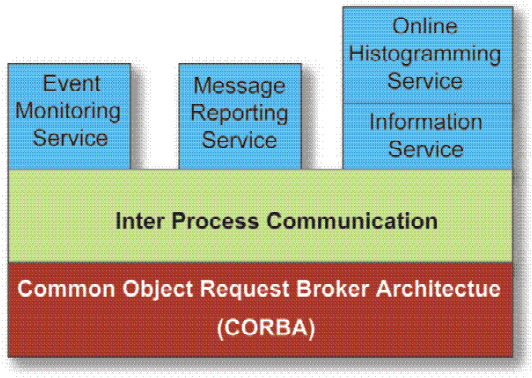

Figure 8 Information Sharing architecture.

The common principle of all Online Software services is an interaction between an information provider who provides data and receives commands and an information consumer who receives the data and issues commands [8]. To use the information sharing services, the user has to implement the corresponding interfaces of the different services.

In order to exchange user-defined information between software applications, the Online Software offers a service called *Information Service (IS)*. Any *Information Provider* can make its information publicly available via the *InfoDictionary* interface. An *Information Consumer* can subscribe via the *InfoReceiver* interface to this information and gets a notification from IS via the *InfoCallback* interface when an update of the information has been done. In addition an Information Consumer can get the information on request without subscribing for the information via the *InfoDocument* interface. The structure of the information is defined in XML and the description of the information is available at run-time.

At the Test Beam, the sub-detectors were using the IS to exchange information between their different applications as well as between the applications and the GUI for the shift users: Defining their information together with the description of their information in XML and implementing the different IS interfaces in their applications as well in specialized GUI panels, allowed them to exchange and publish detector and application specific values like the number of processed events, still pending events or buffer sizes which were then displayed on specialized panels in the GUI. In addition, the information were put from IS via the OBK to permanent storage so that they were accessible for further analysis of the data taking. Fig. 9 shows the interfaces provided by the IS.

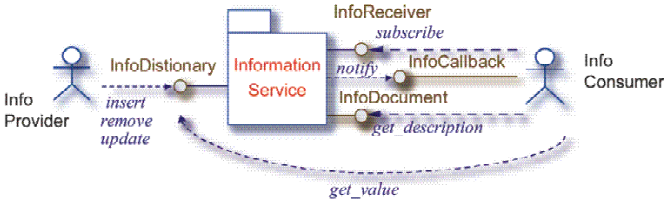

Fig. 9 Information Service (IS) to exchange used-defined information between applications.

The *Message Reporting Service (MRS)* provides the transportation of messages between different applications. Each message is identified by a unique ID, severity and text. A *Message Provider* can send messages via the *MRSStream* interface to the MRS. To automatically receive a message via the *MRSCallback* interface, a *Message Consumer* can implement the *MRSReceiver* interface and specify what kind of messages he wants to receive.

For the Test Beam, the different sub-detectors implemented the MRS interfaces in their applications to monitor the status of their applications and to inform the shift crew about upcoming errors via the GUI. In this way, the sub-detectors were able to detect problems early on and solve the problems in a fast way. Fig. 10 shows the interfaces provided by MRS.

Fig. 10 Message Reporting Service (MRS) to transport messages between different applications.

The *Event Monitoring Service (EMS)* allows the transportation of samples of physical events from points in the DataFlow chain to software application that want to analyze them. An application that is able to provide such samples, has to implement the *EventSampler* interface, applications interested in this samples can implement the *EventIterator* interface. If an Event Consumer requests information from a specific

Event Provider, the EMS system asks the Event Provider to start the sampling process. The Event Provider then starts to add the event samples via the *EventAccumulator* interface. If no more applications are interested in the samples of an Event Provider, the EMS system stops the sampling.

For the Test Beam, EMS was used to examine the raw data as well as to put them to histograms for monitoring purposes: The *Event Dump (ED)* application of the Online Software was used to look at the raw data from the Test Beam detectors. The ED uses the EMS to transport event fragments and allows the user to see the event fragments (e.g. for debugging purposes) in a raw format via a graphical user interface. In addition, the TileCal was using the EMS for a standalone monitoring application: Their application was using the EMS to transport event fragments from different points in the DataFlow chain to a standalone application that was producing ROOT histograms out of the data. Fig. 11 shows the interfaces of the EMS.

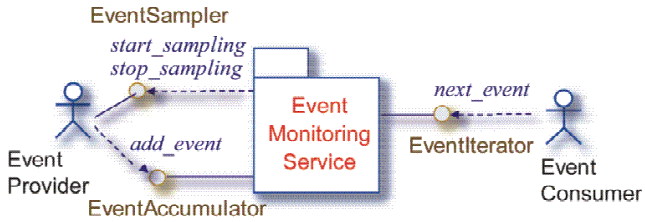

Fig. 11 Event Monitoring Service (EMS) to transport samples of physical events.

The *Online Histogramming Service (OHS)* allows applications to exchange histograms. It is based on the Online Software IS, described earlier in this paper. However, in contrast to IS, the transported information has a predefined format. The current implementation supports ROOT and raw histograms (vectors of data). Having an abstract interface layer, it also allows to add support for other types of histograms.

An *Online Histogramming* application of the Online Software based on ROOT and using the OHS offered the Test Beam detectors the possibility to look at online produced histograms for monitoring and debugging purposes. Fig. 12 shows the interface of the OHS.

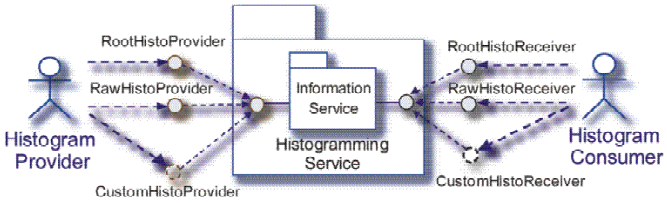

Fig. 12 Online Histogramming Service (OHS) to exchange histograms.

Finally, the *DAQ-DCS Connection (DDC)* allows the communication between the DAQ and the DCS system using the Online Software services in three ways: First the DCS can send alarms via DDC to the DAQ using MRS messages. Second, the DAQ can send commands to the DCS via DDC. This can be either done by using specialized controllers or, for commands outside of state transitions, using the IS. Third, data can be send via the DDC in between the DCS and the DAQ. On the Online Software side, this is realized via the IS service.

At the Test Beam, the TileCal was using the DDC to transport information from the DCS to the IS server of the Online Software, where the information was published and also recorded via the OBK. Using the OBK, the TileCal was able to reproduce and study the conditions of the data taking in more detail for their analysis. Fig. 13 shows the communication types of the DDC.

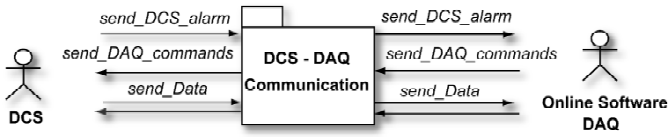

Fig. 13 DAQ-DCS connection (DDC).

VII. Support

The Online Software is based on a common software release scheme where one release was used for the full Test Beam period. This was giving the user the necessary stability, yet offering the Online Software to apply necessary patches. The Online Software was installed and supported on a separate server at the Test Beam.

In order to support the Test Beam users, the Online Software includes a training package which allows the user to learn about the different components of the Online Software and their usage. Furthermore detailed descriptions about the installation and the usage are available via the WWW. Finally specialized training lectures with hands-on examples were given for the Test Beam users before the start of the Test Beam [5]. In this way, the setup time for the software at the Test Beam has been minimized, maximizing the available time at the Test Beam.

During the Test Beam, the Online Software was offering on-call expert support to help with upcoming problems. Also, feed-back meetings with the Test Beam users were organized on a regular basis to discuss problems and experiences of the users.

VIII. Experiences from the Test Beam

The Online Software was successfully used at the Test Beam by the different sub-detectors. It not only offered the detectors the possibility to use a ready made, flexible and adaptable DAQ software, but also gave the Online Software group the possibility to test the software in a real test environment. Besides receiving a lot of positive feedback from the detectors, the Test Beam also helped to improve the Online Software: Problems of the software were identified and new requirements for the Online Software were brought up.

In this context, the Test Beam proved to be a very valuable test-bed for the Online Software on the way to the final system.

IX. SUMMARY

The Online Software is the global system software of the ATLAS Data Acquisition system to configure, control and share information. Different detectors were using the Online Software framework to implement their specific needs, minimizing the work to setup the DAQ system. Also, it is an important step to start commissioning the final ATLAS DAQ system. Furthermore it allowed in an easy way to combine different detector setups to perform a combined data taking of the three detectors. The training and support of the Online Software helped the detectors in preparing their setup and software in front of the Test Beam, minimizing the time for setup during the Test Beam and so maximizing the usable Test Beam time for data taking. Valuable feedback has been given by the different detectors that helped to improve the Online Software.

X. ACKNOWLEDGMENT

We'd like to thank all the detectors at the Test Beam that have used the Online Software for their feed-back.

XI. REFERENCES

[1] ATLAS Collaboration, *Technical Proposal for a General-Purpose pp Experiment at the Large Hadron Collider at CERN*, CERN/LHCC/94-43, 1994

[2] ATLAS Collaboration, *ATLAS High-Level Trigger, DAQ and DCS Technical Proposal*, CERN/LHCC/2000-17, 31 March 2000

[3] ATLAS Collaboration – ATLAS Technical Coordination Technical Design Report – CERN LHC 99/01, ATLAS TDR 13, January 1999.

[4] Online Software: http://atlas-onlsw.web.cern.ch/Atlas-onlsw/

[5] I. Alexandrov et al.., *ATLAS DAQ Configuration Database*, Proc. CHEP 2001, Beijing.

[6] I. Alexandrov et al.., *OBK – An Online High Energy Physics' Meta-Data Repository*, Proc. 28th VLDB Conference, Hong Kong (2002)

[7] Run Control User Guide, ATLAS TDAQ-1 Note 107

[8] I. Alexandrov et al., *Online Monitoring Software Framework in the ATLAS Experiment*, Proc. CHEP 2003, to be published