The ATLAS DAQ and Event Filter Prototype "-1" Project

G Ambrosini [1], D Burckhart [2], M Caprini [3], M Cobal [2], P-Y Duval [4], F Etienne [4], R Ferrari [5], D Francis [2], R Jones [2], M Joos [2], S Kolos [6], A Lacourt [2], A Le Van Suu [4], A Mailov [7], L Mapelli [2,8], M Michelotto [2], G Mornacchi [2], R Nacasch [4], M Niculescu [2,3], K Nurdan [7], C Ottavi [2], A Patel [2,9], F Pennerath [2], J Petersen [2], G Polesello [5], D Prigent [2], Z Qian [4], J Rochez [2], F Scuri [10], M Skiadelli [2,11], I Soloviev [2,6], R Spiwoks [2], F Touchard [4], L Tremblet [2], G Unel [7], V Vercesi [5], S Wheeler [12], A Wildish [2,13]

A project has been approved by the ATLAS Collaboration for the design and implementation of a Data Acquisition and Event Filter prototype, based on the functional architecture described in the ATLAS Technical Proposal. The prototype consists of a full "vertical" slice of the ATLAS Data Acquisition and Event Filter architecture, including all the hardware and software elements of the data flow, its control and monitoring as well as all the elements of a complete on-line system. This paper outlines the project, its goals, structure, schedule and current status and describes details of the system architecture and its components.

[1] Laboratory for High Energy Physics, University of Bern, Bern
[2] CERN, Geneva, Switzerland
[3] Institute of Atomic Physics, Bucharest, Romania
[4] Centre de Physique des Particules de Marseille, IN2P3, France
[5] Dipartimento di Fisica Nucleare e Teorica dell'Universita' di Pavia e INFN, Pavia, Italy
[6] Petersburg Nuclear Physics Institute, St. Petersburg, Russia
[7] Bogazici University, Istanbul, Turkey
[8] Spokesperson
[9] On leave from School of Computing and Information Systems, University of Sunderland, UK
[10] Dipartimento di Fisica dell'Universita' di Udine, Gruppo collegato di Udine INFN Trieste, Udine, Italy
[11] Now at SYBASE Relational Technology, Athens, Greece
[12] Sheffield University, Sheffield, UK
[13] Presently with Data Conversion Services, Geneva, Switzerland

Preprint submitted to Elsevier Preprint *27 May 1997*

1 Introduction

The final design of the Data Acquisition (DAQ) and Event Filter (EF) system for the ATLAS experiment at the LHC is not scheduled to start before 1999. Clear specification of the detector requirements, further investigations of hardware and software technologies and integration studies are still required before a mature design can be achieved. The ATLAS DAQ Group has chosen to approach such pre-design investigations via a structured prototype, supporting the evaluation of hardware and software technologies and their system integration aspects.
A project is ongoing in ATLAS to design and implement a full DAQ/EF prototype based on the Trigger/DAQ architecture described in the ATLAS Technical Proposal [1] and on supporting studies of the full system functionality. However, at this stage, the required final performance is obviously not yet taken into account. For this reason, it has been named Prototype "-1" (hereafter referred to as DAQ/EF-1 or simply DAQ-1) [2]. This project follows, and is in part based upon, previous investigations of DAQ related issues for the LHC experiments [3].
After an outline of the ATLAS Trigger/DAQ functional architecture (Section 2), this paper describes the DAQ/EF -1 project (Section 3). Details of its major system components are given in Sections 4 to 6. The project also covers issues related to the software development environment, from the Operating Systems [4] to Software Engineering and CASE tools [5].

2 The ATLAS Trigger/DAQ System

The Trigger and DAQ system for ATLAS must be capable of 1) reducing the event data produced at the LHC rate to a manageable data volume for permanent storage and 2) moving and distributing the corresponding huge volume of data from the detector front-end electronics to permanent storage. Rate and data reduction are organised in ATLAS in three trigger levels: the Level-1 trigger (LVL1), the Level-2 trigger (LVL2) and the Event Filter.

2.1 The ATLAS Trigger/DAQ functional scheme

At LVL1 (see Figure 1), special-purpose processors act on reduced-granularity data from a subset of the detectors (calorimeters and muon trigger chambers). The LVL1 trigger accepts data at the full LHC bunch-crossing rate of 40 MHz (every 25 ns). The latency (time taken to form and distribute the LVL1 trigger decision) is $\sim$2 microseconds, and the maximum output rate is expected not to exceed 100 kHz at the maximum LHC luminosity (10^{34} cm^2 sec^{-1}). At this luminosity, each bunch crossing produces an average of about 23 proton-proton collisions, which result in an aggregate data throughput of at least 100 GByte/s into the Data Acquisition system.

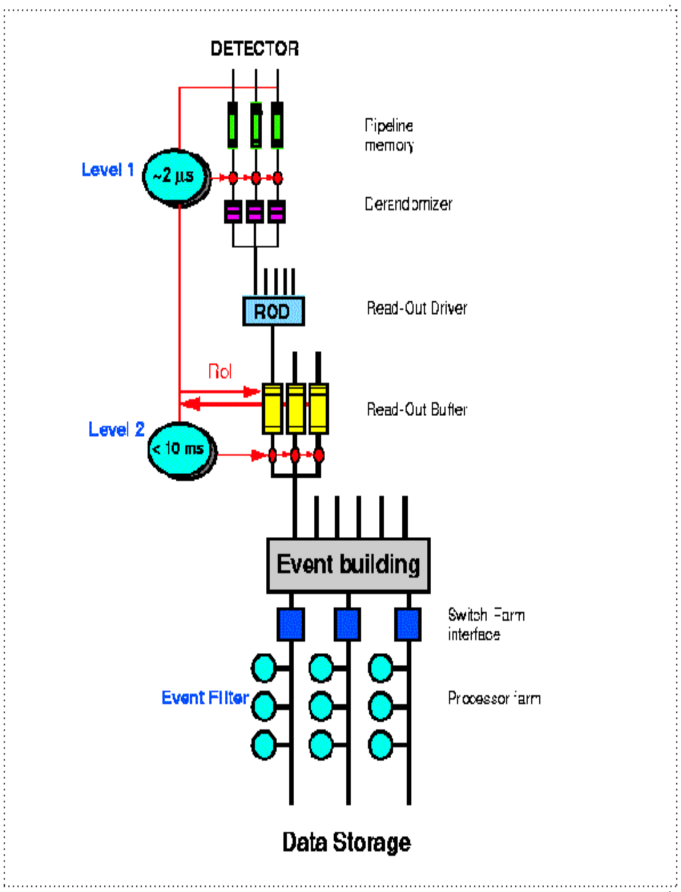

Fig. 1. ATLAS T/DAQ functional scheme

The LVL2 trigger must reduce the LVL1 rate to a level at which full event building is possible, so that the EF can perform final selection before permanent storage. The LVL2 trigger uses full-granularity, full-precision data from most of the detectors, but examines only regions of the detector identified by the LVL1 as containing interesting information (Regions of Interest or RoIs). Consequently, only a small fraction of the total detector data must be accessed and processed by such RoI based LVL2 architecture, with corresponding advantages in terms of the required processing power and bandwidth.
After an event is accepted by the LVL2 trigger, the full data are sent to the EF via the Event Builder (EB). At the level of the Event Filter, the full event data are used to make the final selection of events to be recorded for off-line analysis.

2.2 *One implementation option of the ATLAS Trigger/DAQ scheme*

Several LVL2 architectures [6] are presently under study in ATLAS, providing different rejection factors and, therefore, different requirements for the event building phase of the DAQ and the Event Filter. For the specification of the DAQ/EF prototype, we have adopted the Technical Proposal view as our working principle.
When an event is accepted by LVL1, the data from all the front-end electronics systems are transferred to off- detector read-out cards, containing read-out buffers (ROBs). Data from all the detectors must arrive at the ROBs in a specified, uniform format. For this purpose, Read-Out Drivers (RODs) interface the detector specific front- end electronics to the common ROB. The LVL2 system is separated from the main data flow and accesses the RoI data, the only data required by the LVL2 algorithms, from the ROBs via its own network, to provide a rejection of $\sim 10^2$. Most of the data are not accessed from the buffers during the LVL2 latency, which is variable and up to ∼10 ms on average. For the ∼1 kHz of LVL2 accepted events, the full detector data are transferred from the ROBs via the EB to the EF system for the final stage of selection before recording. Complete event reconstruction is possible in the EF farm, with average decision times of up to ∼1 s. The EF system must achieve a data-storage rate of ∼100 MB/s by reducing the event rate and/or the event size. For some triggers, for example Higgs boson candidates, the full event data will be recorded with an event size of ∼1 MB, allowing a maximum event rate of ∼100Hz.

3 The ATLAS DAQ/EF Prototype "-1" Project

The goal of the ATLAS DAQ/EF-1 project is to produce a fully functional prototype suitable for evaluating candidate technologies and architectures for the final DAQ/EF system of the ATLAS experiment at the CERN LHC accelerator. The prototype consists of a complete "vertical" slice of the ATLAS DAQ/EF architecture, including all the hardware and software elements of the data flow, its control and monitoring as well as all the other elements of a complete on-line system, from the detector read-out to data recording.
The phases of the project are as follows: 1) Define the requirements; 2) Perform pre-design evaluations of candidate techniques and technologies; 3) Make an architecture design based on the results of the pre-design evaluations; 4) Propose a detailed design and implementation; 5) Perform unit testing and system integration.
Work started on defining the requirements in Spring 1996 and it is expected that the prototype should be running in 1998. Each phase of the project is reviewed by the ATLAS Trigger/DAQ community.

The system consists of 4 major sub-systems, namely:

- The Data flow, the hardware and software complex responsible for moving, distributing and monitoring the event data from the detector electronics to the mass storage (Section 4)
- The Back-End DAQ, encompassing all the software for configuring, controlling and monitoring the DAQ (Section 5)
- The Event Filter, based on processor farms running an on-line implementation of the standard ATLAS reconstruction and physics analysis applications (Section 6)
- The Detector Interface, the collection of detector specific aspects of the DAQ system, which includes the detector read-out, monitoring and calibration, as well as the integration with the run control and database organisation. This part of the project is now in the User Requirement specification phase and will not be discussed further here.

4 The Data flow Component

The data flow component of the ATLAS DAQ-1 system is responsible for moving the event data from the detector read-out links to the final mass storage. It also provides event data for monitoring purposes and implements local control for the various data flow elements. A global view of the data flow component is shown in Figure 2.

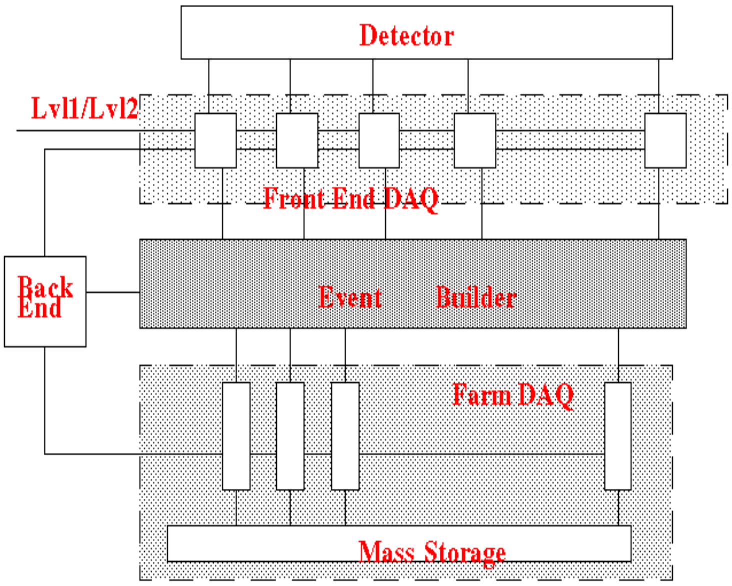

Fig. 2. The DAQ/EF-1 Data flow

Three main functions are provided by the data flow specifically: the collection and buffering of data from the detector (the front-end DAQ), the merging of fragments into full events (the EB) and the interaction with the EF (the farm DAQ). This leads to a decomposition of the system into the read-out crate, the EB and the sub-farm DAQ.

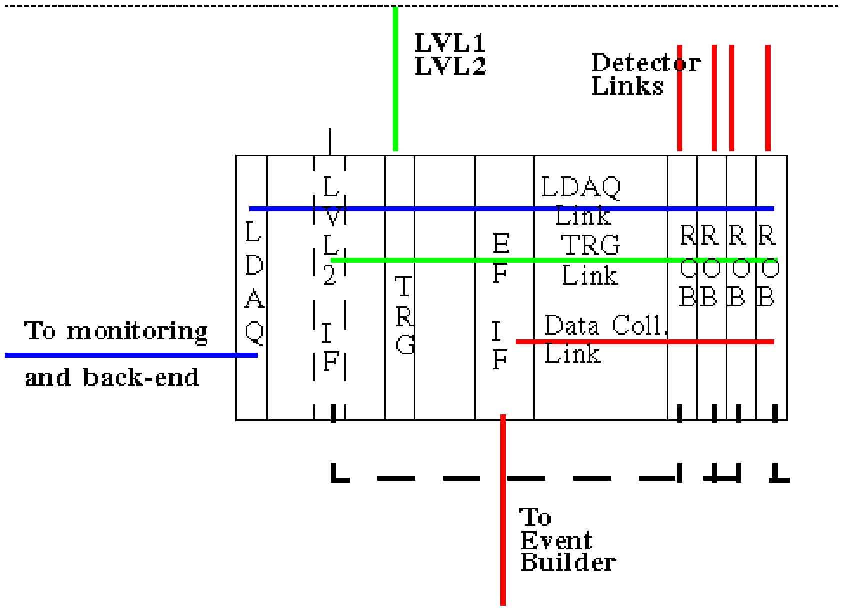

Fig. 3. The read-out crate

4.1 The read-out crate

The read-out crate is the modular element/footnoteBy modular we mean an element which is replicated in the system in many independent copies (i.e. can be developed and run independently). responsible for handling the movement of data, originating from a group of RODs, between the detector link and the event builder. It provides the following main functions:

- Detector read-out, buffering and data distribution to other data flow elements in the crate. This function is provided by the ROB (read-out buffer) module in Figure 3.
- The control of the flow of data within the crate. For example, event fragments buffered in the ROBs have to be discarded or moved to the EB according to the response provided by a data reduction device such as the LVL2 trigger system. The TRG (trigger) module provides this function.
- Fragments of accepted events are moved from the ROB memories and merged into a "crate fragment" (consisting of all the elementary fragments

from the individual ROBs). These are buffered and then sent to the event builder. This function is implemented by the EFIF (EF interface) module [14].
- Other ancillary functions must also be provided locally in the crate: the control of the crate, the handling of errors and support for event monitoring (the capability for external application programs to receive, upon request, a sample of the data flowing in the crate). An interface point to back-end DAQ functions is also needed. The Local DAQ (LDAQ) is the component which is assigned to these ancillary tasks.
- An important role is played by the intra-crate links (e.g. the one connecting the ROBs and the EFIF to support the data collection function) and the related communication protocols.

The above elements as well as the links, which allow the communications (transport of data and control commands), are sketched in Figure 3.

4.2 The Event Builder

The event builder merges the "crate fragments" belonging to the same bunch crossing into a complete event at a single destination. This will be implemented by means of a switching network which will allow events to be merged concurrently.
We are currently studying the possibility of using commercially available switching technologies. To this end we intend to integrate into the DAQ-1 at least two different communication technologies: ATM and Fibre Channel are our current choice. Such studies, however, cannot provide insights as to how the EB will scale to large configurations. We intend to address such scalability issues with the integration into the DAQ/EF-1 of the DS-Link based large switch configuration provided by the MACRAME project [7].

To fulfil the above objectives, the event builder itself is partitioned into two layers:

- A technology independent layer, which implements the event building algorithm (e.g. it determines which destination subfarm will receive a given event). This consists of the data flow manager and source/ destination processes depicted in Figure 4. The data flow manager implements the highlevel event building protocol while the source/destination processes are responsible to send/receive the event fragments (from/to the buffers delimiting the EB) via the technology dependent layer. As far as the flow of data is con-

[14] This is a functional description of the "data collection" process. One can also envisage the extreme situation where each ROB connects directly to an event builder port; this means collapsing the ROB with the output port of the EFIF.

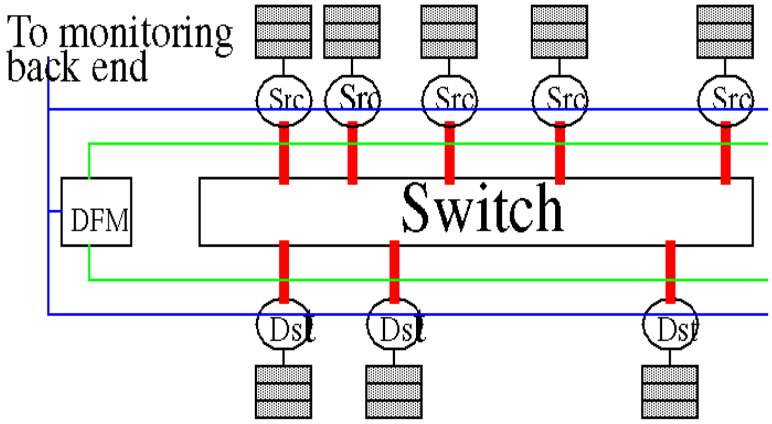

Fig. 4. The Event Builder

cerned, partitioning[15] is naturally implemented by the data flow manager (i.e. a subset of destinations can be assigned to a subset of sources).

- A technology dependent layer, which interfaces in a common way the technology independent component to the features of the switching hardware.

Ancillary functions, such as local control and monitoring of its behaviour as well as interfacing to the back-end part, are relevant to the EB too. To this end the data merging component is complemented by a local DAQ.

4.3 The sub-farm DAQ

The DAQ part of the sub-farm receives full events from the EB, buffers them in the Switch to Farm Interface (SFI) element, distributes the events to the EF and records or rejects events on the basis of the EF reply.
Although there are different performance requirements between the data flow in the read-out crate and in the sub-farm, they are functionally very close. One common aspect is that as far as the data flow is concerned, individual crates as well as individual sub-farms, are independent modular elements. Our objective, at the moment, is to have a joint approach to the data flow in the read-out crate and in the sub-farm. This allows us to exploit the common issues at the design and organisational level; we also expect to share implementation solutions where feasible.
Support for features such as local control and event monitoring, has to be provided by the sub-farm DAQ as well. This also calls for a joint solution for the LDAQ component.

[15] The capability of running multiple, concurrent and independent data acquisition sessions.

4.4 Status of the developments

The first phase of the project, one of pre-design analysis and early prototypes both in the area of the read-out crate and the event builder, was completed in 1996. Extensive documentation in terms of summary documents, workplans and technical notes is available in [8] and [9] to which reference should be made for detailed results. Here we discuss some relevant topics.
Pre-design analysis of the read-out crate and the sub-farm DAQ has resulted in the identification, of the concept of an I/O module: the combination of a memory, a processor and a number of I/O channels. For example, a ROB is an instance of an I/O module with one high-speed input channel (from the detector) and at least two output channels (one to the EFIF and the second to the LDAQ); the EFIF, TRG and SFI are other instances of an I/O module.

A number of initial selections of hardware elements has been made in the area of the read-out crate.

- VMEbus is used as the crate integration bus as well as the initial implementation of the intra-crate links; evaluations and studies to select other technologies for intra-crate links, such as the Raceway Interlink [10] and the PVIC [11] systems, are planned.
- PCI has been selected as the I/O integration bus within an I/O module, the PMC format is the preferred one for I/O interfaces.
- I/O modules are currently implemented by VMEbus, PowerPC based processors with 2 (or more) PMC sites.

Hardware libraries (e.g. interfaces to the module hardware features such as the PCI bus) and generic (hardware and operating system independent) packages (such as buffer management and I/O scheduling) have been developed to support the generic I/O module element.
Studies of operating systems for real-time applications including the possibility of using PCs and/or the WindowsNT operating system, at least in the area of the local DAQ, have also been performed and are reported in [4].
Intra-crate communication protocols have been defined and prototyped to support both the local DAQ (e.g. control transactions between LDAQ and a ROB) and the data flow (e.g. the data flow control messages exchanged between the TRG module and the EFIF).
A local control and a monitoring sub-system have also been prototyped. The event monitoring sub-system has been designed to de-couple the sampling of event data (from the data flow) from the request of events (from analysis programs). This de-coupling is achieved via a database which is periodically updated to contain a consistent snapshot of the data flowing in the crate. The same LDAQ functionality is applicable to the sub-farm DAQ.
Based on the above hardware choices and software basis, a read-out crate pro-

totype has been built. It covers the full internal functionality of the crate and is being used to test design ideas and to measure performance in certain areas. For example, the TRG module has been shown to be capable of performing part of its task (input from PMCs, some calculation on the input and then re-sending the result to one I/O module in the crate) at a rate compatible with the final ATLAS requirement of 100KHz (LVL1 rate).
In the event builder area we have thanksed some of the issues related to the general event builder process as well as performing technology evaluations [12].

5 The Back-End

The back-end DAQ encompasses the software to configure, control and monitor the DAQ but excludes the management, processing or transportation of physics data. A set of user requirements for the back-end software [13] has been defined that identifies a number of components as described below.

5.1 The back-end components

- Partition manager. The DAQ contains many resources (both hardware and software) which, at times (e.g. for detector debugging, calibration, etc.) must be shared to allow independent and simultaneous use. Therefore, their usage must be controlled to avoid conflicts. The purpose of the Partition Manager is to formalise the allocation of DAQ resources and allow groups to work in parallel without interference.
- Run control. The run control system controls the data taking activities by coordinating the operations of the DAQ sub-systems, back-end software components and external systems. It has user interfaces for the shift operators to control and supervise the data-taking session and software interfaces with the DAQ sub-systems and other back-end software components. Through these interfaces the run control can exchange commands, status and information used to control the DAQ activities.
- Configuration databases. A data acquisition system needs a large number of parameters to describe its system architecture, hardware and software components, running modes and the system running status. One of the major design issues of Atlas DAQ is to be as flexible as possible, externally parameterised by databases. The database management is a major technical issue for the system.
- Message reporting system. The aim of the Message Reporting System (MRS) is to provide a facility which allows all software components in the ATLAS DAQ system and related processes to report error messages to other components of the distributed DAQ system. The MRS shall provide for the transport, the filtering and the routing of the messages. It shall provide a

facility for users to define unique error messages which will be used in the application programs.

- Status display. The status display presents the status of the current data taking run to the user in terms of its main run parameters, detector configuration, trigger rate, buffer occupancy and state of the run controller.
- Monitoring. Monitoring in this document refers to the monitoring of physics data as it flows through the DAQ. The issue of monitoring is to provide a program skeleton with which users can build their own tasks for analysis. One of these tasks is the event dump which tests event integrity and is useful as an event structure debugging tool. It also acts as an example monitoring task and is developed by the DAQ group.
- Run bookkeeper. The purpose of the Run bookkeeper is to archive information about the data recorded to permanent storage by the DAQ system. It records information on a per-run basis and provides a number of interfaces for retrieving and updating the information
- Data viewing. The Data Viewing System (DVS) will allow users to have access to aggregate displays, such as histograms and trace plots, representing the performance of sub-systems of the DAQ system such as subdetectors and trigger.
- Event Viewing. The Event Viewing System (EVS) will provide, via a simple, flexible user interface, a facility for viewing event data in all detectors together with a description of the event classification.
- Process Manager. The purpose of the process manager is to perform basic job control of software components of the DAQ. It shall be capable of starting, stopping and monitoring the basic status (e.g. running or exited) of software components on the DAQ workstations and front-end processors. In this description the terms process and job are considered equivalent.
- Test Manager. The purpose of the test manager is to provide a means of organising individual tests for hardware and software components. The individual tests themselves are not the responsibility of the test manager which simply assures their execution and verifies their output. The individual "atomic" tests are intended to verify the functionality of a given component. They will not be used to modify the state of a component or to retrieve status information. Tests are not optimised for speed or use of resources and are not a suitable basis for other components such as monitoring or status display.
- Diagnostics Package. The diagnostics package uses the tests held in the test manager to diagnose problems with the DAQ and confirm its functionality. By grouping tests into logical sequences, the diagnostic framework can examine any single component of the system (hardware or software) at different levels of detail in order to determine as accurately as possible the functional state of components or the entire system. The diagnostic framework will report the state of the system at a level of abstraction appropriate to any required intervention.

5.2 Status of the back-end software

We have completed the high-level design of the essential components (i.e. run control, configuration databases, message reporting system, process manager as well as elements of the partition manager and status display) and evaluated suitable supporting technologies in the areas of data storage, inter-process communication, dynamic object behaviour, graphical user interfaces and software integration [14].

5.3 Software development environment

In parallel to the definition and design of the DAQ sub-systems, we have addressed the issue of the means by which the necessary software will be developed. We have adopted an object-oriented method supported by a commercial CASE tool and linked to a programming environment [5]. We are currently investigating suitable code management systems and distribution tools. This work is performed in conjunction with all software groups of ATLAS (on-line and off-line) based on a common set of requirements [15] and summarised in the experiment's Computing Technical Proposal [16].

6 The Event Filter

The Event Filter computing instrument will be provided by a processor farm. In the present view of the ATLAS DAQ, data coming via the Switch Farm Interface (SFI) will be distributed to several processors, working in parallel, each one analysing a full event. In its final configuration the farm must provide enough processing power to build the high-level information which will allow the Event Filter to take the decision of accepting or rejecting the event. Further requirements might come from the necessity of accepting and processing different trigger types, including physics events, calibration and monitoring data. In this context special care must be taken to avoid the creation of bottlenecks due to a lack of resources while retaining the highest possible flexibility.
It is recognised that the final performance of this system will be determined by the maximum manageable complexity and the available financial resources. Extrapolations based on the reconstruction of CDF/D0 events and of ATLAS Monte-Carlo events provide an estimate of approximately 25 SPECint95-sec per event. This brings the total amount of processing power required by the Event Filter process (excluding calibration and monitoring) to $2.5\text{x}10^4$ SPECint95.
The aim of the DAQ/EF -1 project is to address some of the fundamental

issues of Event Filtering in ATLAS and their impact on the data acquisition architecture on one side and the filtering code on the other. More specifically, the project will:

- ascertain the validity of the strategy of using the off-line reconstruction and physics analysis code as the filtering program running in the farm, with no development of specific Event Filter algorithms
- use the filter program as the basic tool to study the fundamental parameters of the EF, namely event distribution schemes, farm architectures and processor types and configurations
- via software instrumentation and its adaptation to the particular needs of the real-time DAQ environment, provide feedback on features required for an effective use as a filter program, in particular concerning reliability, robustness, program structure and execution performance
- address the issues of on-line calibration and monitoring and their impact on the architecture and partitioning of the EF farm.

The EF part of the project is now in the definition and pre-design phase.

7 Acknowledgements

We gratefully acknowledge the active support and efficient technical assistance of C. Parkman and his team. We also warmly thank D. Klein for her precious work in many aspects of the organisation of the DAQ/EF -1 project and C. Costaz for her contribution in maintaining our development environment. We are also grateful to the many ATLAS colleagues with whom we have valuable and constructive exchanges of ideas and results.

8 References

[1] ATLAS Technical Proposal, CERN/LHCC/94-43 (ISBN 92-9083-067-0)

[2] ATLAS DAQ/Event Filter Prototype "-1" Project, http://atddoc.cern.ch/Atlas/

[3] L.Mapelli et al., A scalable Data Taking System at a Test Beam for LHC, CERN/DRDC 90-64, CERN/DRDC 91-23, CERN/DRDC 92-1, CERN/DRDC 93-25, CERN/LHCC 95-47. See also the World Wide Web at URL http://rd13doc.cern.ch/

[4] Operating System Studies for DAQ Applications, G.Ambrosini et al. Published in these proceedings

[5] Use of a CASE tool for developing ATLAS DAQ software, A.Patel et al. Published in these proceedings

[6] A demonstrator program to study options for the ATLAS Level 2 Trigger, J.R.Hubbard et al. Published in these proceedings

[7] B.Martin et al., Realisation of a 1000 node High Speed packet Switching Network, Int. Symposium on Problems of Modular Information Computer Systems, 1995.

[8] Front-end DAQ Discussion Group. Summary document and workplan. April 1996. http://atddoc.cern.ch/ Atlas/FrontEnd/document/draft.ps

[9] http://atddoc.cern.ch/Atlas/Notes/Welcome.html

[10] ANSI/VITA S-1994, Raceway Interlink.

[11] PCI-to-PCI Inter-Crate Connection (PVIC), Catalogue 97-98, pp. 25-26, CES Geneva.

[12] Event Builder Working Group. Summary document and workplan. August 1996.http://atddoc.cern.ch/Atlas/ EventBuilder/document/summary.ps.

[13] ATLAS DAQ Back-end software User Requirements Document. 30 May 1996. http://atddoc.cern.ch/Atlas/ DaqSoft/document/draft_1.html

[14] Software Technologies For A Prototype ATLAS DAQ, R.Jones et al. Published in these proceedings

[15] ATLAS Software Development Environment User Requirements Document Issue: 1.0, ATLAS internal note SOFT-NO-03, December 1996, http://ptsun00.cern.ch/~ruggier/AtlasSE/URD/

[16] ATLAS Computing Technical Proposal, CERN/LHCC/96-43, Issue 1.0, 15 December 1996, pp 54, ISBN 92- 9083-092-1. http://atlasinfo.cern.ch/Atlas/GROUPS/SOFTWARE/ctp/ctp-work.html