

17 Computing

17.1 Introduction

The ATLAS Level-1 Trigger will require a large body of accompanying software at all phases of its design, development, installation, and running. We will need to build a well-documented and easily maintainable system that will last through the inevitable changes in personnel during the long running period of ATLAS. This chapter aims to give an overview of most aspects of our software and computing requirements, from module diagnostics to data acquisition and online monitoring, calibration and offline simulation and reconstruction. In many cases it summarizes experience and plans which have been presented in greater detail earlier in this document, in particular Sections 7.8, 8.6, and 11.2.

17.2 Physics simulation

Over the last few years we have performed wide-ranging and in-depth physics simulation of the performance of a number of proposed trigger designs. The methods used are described in the accompanying ATLAS Trigger Performance Status Report [17-1].

Initially, two levels of simulation for the calorimeter trigger were implemented: one level used smeared four-vectors with a only simplified detector model, but included the full time-history of each pulse. This was a fast simulation and was essential for studying effects of BCID, pile-up etc. The other level performed full GEANT tracking and digitization, but did not include the pulse history. This was more useful for understanding effects of resolution etc. We have also written some other stand-alone programs to examine certain requirements of the system, such as BCID, in greater detail. Recently, these approaches have been unified in a new program which has been used to produce many of the calorimeter simulation results presented in this document. This program will be developed further in the near future for continuing detailed design studies.

Similarly in the muon trigger, many early studies used simplified models of the detector and the physics. However, the acceptance of the muon trigger is much more complex, not least because of the highly nonuniform magnetic field, and so the motivation to use the full GEANT simulation was earlier and greater. A detailed description of the detector geometry and response of the muon trigger system has been simulated with the ATLAS Monte Carlo program DICE (GEANT based). The simulated data have been processed through a detailed simulation of the trigger logic, independently for the barrel and endcap subsystems.

Independent stand-alone simple simulation programs have been used to check the impact on physics performance of the more important physics effects of the muon trigger system. Also, the accidental trigger rate induced by the cavern background has been studied with dedicated numerical and Monte Carlo programs, to evaluate the robustness of the trigger system against possible severe environmental background conditions.

In the longer term, a suitable software model of the trigger will need to be implemented in the framework of the object-oriented offline software. This will be required both for future

simulations, for developing trigger menus, and in the reconstruction program for offline calibration and monitoring of the trigger performance.

17.3 Diagnostic, test and calibration software

During the prototyping phase, all components of the trigger will need to be tested. Each group will be providing its own test and diagnostic software for this purpose. There would be some benefits arising from a common approach, but the logistical difficulties involved may make this impossible. The experience of designing object-oriented software gained during the calorimeter trigger demonstrator programme may however be of use to the other groups.

The calibration procedures required by the various trigger components have been described elsewhere. It is envisaged that many of these will be implemented in software as test modes of the DAQ system.

17.4 DAQ software

We will need software running in a data acquisition framework on various timescales: from laboratory and beam tests of prototype modules to the full ATLAS environment. We are closely following the progress of the DAQ group 'prototype -1' project and hope also to learn from their experience in designing and implementing object-oriented software systems.

17.4.1 DAQ for prototype testing

Prototype modules will be tested in the laboratory and probably also in the ATLAS test beam. At this stage we would preferably use software integrated into the prototype ATLAS DAQ system to start gaining experience with that. However, it is possible that not all the DAQ components we require will be ready on the timescale we need, so we may have to keep evolving some of our existing DAQ software in the short term.

17.4.2 DAQ for final ATLAS system

Most of the basic DAQ infrastructure which we had to develop for the laboratory and test-beam environments will, in ATLAS, be provided by the DAQ group. We will still have to provide some software to run in or to control whatever processor(s) are chosen for the local crate controllers and RODs. We will also need to provide software, callable by the Run Control component of the ATLAS DAQ, to initialize the modules from their power-up state (e.g. load the trigger) and to prepare them for run start, etc.

For debugging and some calibrations, we will want to run the DAQ with the trigger as a stand-alone partition. Other calibrations will require both the trigger and either the muon or calorimeter detectors to be controlled together. We will need to provide the software to run in these modes, though the principle of partitioning the DAQ is envisaged as part of the ATLAS central DAQ system.

A vital feature of the online system will be real-time monitoring of the integrity of the trigger and the quality of the triggered data. We expect to monitor the system at various levels. Some monitoring may be provided in hardware, for example on the PPMs in the calorimeter trigger, which would give an untriggered 'level-0' view of what is happening in the detector. The next level would be implemented in the readout chain of each system to provide an immediate check of the performance of the electronics. An overall indication of the performance of the trigger system would be provided by online analysis of a fraction of complete events.

17.5 Detector Control System

We may need software to interface with the Detector Control System (DCS). We may want to report the status of our own crates on local displays and we will need to respond to certain DCS commands. We will also need to write the software for the local part of the DCS which is our responsibility. Details of the interaction with the DCS are still to be defined.

17.6 Platforms, languages, tools, and training

The trigger software will be run on a variety of platforms. DAQ, online monitoring, and some test and diagnostic will run in processors housed in VME (or other) crates. Parts of the diagnostic software may be run on PCs or workstations. Calibrations will be performed both online and offline on workstations or work-group servers. For this reason, it is desirable to avoid manufacturer-specific features in the DAQ code, and premature relationships with a single hardware manufacturer should be avoided. The choice of hardware platforms will be guided by ATLAS standards, if and when they emerge.

One of the supposed advantages of OO software is that it will allow for reuse of software components. There may be some merit in considering the design of all the trigger software as a package, with some classes used only online, others only offline, but some shared by online and offline software. If this is desirable, a high degree of portability will be needed. This constrains the choice of languages to (probably) just C++, but possibly including Java as an alternative.

The advantages of Java are that it is a cleaner, safer language than C++; it could allow a natural integration of some applications with Web-based documentation. This might be desirable especially in the ATLAS control room. However, Java can also be compiled and run as stand-alone programs with their own graphical user interface. Compiled Java is only moderately slower than C++.

There is now some OO experience within the level-1 calorimeter trigger group. However, given the ATLAS timescales, some turnover of personnel is inevitable. It is therefore necessary to consider the training of new people in both OO techniques and in the existing software. The latter requires some discipline in both design and documentation. Previous experience in HEP experiments suggests that this self-discipline is very hard to achieve. Even though we tried to approach the design of the calorimeter trigger diagnostics software 'properly', we did not in practice spend enough time in the analysis and design phase, and we had at least one serious redesign of the software part way through the implementation phase.

For the calorimeter trigger diagnostic software, we did not use any CASE tools. Use of such tools in the future is highly desirable, both as a design discipline and to provide design documentation to people joining the project later.

The offline component of our software will clearly be part of the 'ATLAS Software Process' (ASP), as described in the *ATLAS Software Technical Proposal* [17-2]. Since the more online aspects will also be developed by widely-scattered people, some similar approach to project management and quality assurance may be required.

17.7 Summary

We have a considerable software task ahead of us which will require various skills in online and offline software. With extra OO training and suitable CASE tools, we should be able to complete the project in the time available.

17.8 References

- 17-1 *ATLAS Trigger Performance Status Report*, CERN/LHCC/98-15, July 1998.
- 17-2 *ATLAS Software Technical Proposal*. CERN/LHCC/96-43, December 1996.
<http://atlasinfo.cern.ch/Atlas/GROUPS/SOFTWARE/TDR/TDR.bk.ps> or
<http://atlasinfo.cern.ch/Atlas/GROUPS/SOFTWARE/TDR/html/TDR-1.html>